

# Hybrid Equations (HyEQ) Toolbox v2.04

## *A Toolbox for Simulating Hybrid Systems in MATLAB/Simulink<sup>®</sup>*

Ricardo G. Sanfelice  
*University of California  
Santa Cruz, CA 95064  
USA*

David A. Copp  
*University of California  
Santa Barbara, CA 93109  
USA*

Pablo Nanez  
*Universidad de Los Andes  
Colombia*

May 17, 2017

### Abstract

This note describes the Hybrid Equations (HyEQ) Toolbox implemented in MATLAB/Simulink for the simulation of hybrid dynamical systems. This toolbox is capable of simulating individual and interconnected hybrid systems where multiple hybrid systems are connected and interact such as a bouncing ball on a moving platform, fireflies synchronizing their flashing, and more. The Simulink implementation includes four basic blocks that define the dynamics of a hybrid system. These include a flow map, flow set, jump map, and jump set. The flows and jumps of the system are computed by the integrator system which is comprised of blocks that compute the continuous dynamics of the hybrid system, trigger jumps, update the state of the system and simulation time at jumps, and stop the simulation. We also describe a “lite simulator” which allows for faster simulation.

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Installation</b>	<b>3</b>
<b>3</b>	<b>Lite HyEQ Simulator: A stand-alone MATLAB code for simulation of hybrid systems without inputs</b>	<b>3</b>
3.1	Solver Function . . . . .	7
3.1.1	Events Detection . . . . .	13
3.1.2	Jump Map . . . . .	14
3.1.3	Function Wrapper . . . . .	14
3.2	Software Requirements . . . . .	15
3.3	Configuration of Solver . . . . .	15
3.4	Initialization . . . . .	15
3.5	Postprocessing and Plotting solutions . . . . .	15
<b>4</b>	<b>HyEQ Simulator: A Simulink implementation for simulation of single and interconnected hybrid systems with or without inputs</b>	<b>17</b>
4.1	Specialized library for simulation of Cyber-Physical Systems . . . . .	18
4.1.1	Models of physical components . . . . .	19
4.1.2	Models of cyber components . . . . .	20
4.2	The Integrator System . . . . .	22
4.2.1	CT Dynamics . . . . .	22
4.2.2	Jump Logic . . . . .	23
4.2.3	Update Logic . . . . .	23
4.2.4	Stop Logic . . . . .	24
4.3	Software Requirements . . . . .	24
4.3.1	Configuration of HyEQ Simulator with embedded functions for Windows . . . . .	25
4.3.2	Configuration of HyEQ Simulator with embedded functions for Mac/Linux . . . . .	26
4.4	Configuration of Integration Scheme . . . . .	28
4.5	Initialization . . . . .	28
4.6	Postprocessing and Plotting solutions . . . . .	29

5	Examples	29
6	Further Reading	71
7	Acknowledgments	71

# 1 Introduction

To get started, a webinar introducing the HyEQ Toolbox is available at <http://www.mathworks.com/videos/hyeq-a-toolbox-for-simulation-of-hybrid-dynamical-systems-81992.html>  
A free two-step registration is required by Mathworks.

A hybrid system is a dynamical system with continuous and discrete dynamics. Several mathematical models for hybrid systems have appeared in literature. In this paper, we consider the framework for hybrid systems used in [1, 2, 3, 4], where a hybrid system  $\mathcal{H}$  on a state space  $\mathbb{R}^n$  with input space  $\mathbb{R}^m$  is defined by the following objects:

- A set  $C \subset \mathbb{R}^n \times \mathbb{R}^m$  called the *flow set*.
- A function  $f: \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$  called the *flow map*.
- A set  $D \subset \mathbb{R}^n \times \mathbb{R}^m$  called the *jump set*.
- A function  $g: \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$  called the *jump map*.

We consider the simulation in MATLAB/Simulink of hybrid systems  $\mathcal{H} = (C, f, D, g)$  written as

$$\mathcal{H}: \quad x, \quad u \in \mathbb{R}^m, \quad y \in \mathbb{R}^n \quad \begin{cases} x = f(x, u) & (x, u) \in C \\ x^+ = g(x, u) & (x, u) \in D \\ y = h(x) \end{cases} \quad (1)$$

The flow map  $f$  defines the continuous dynamics on the flow set  $C$ , while the jump map  $g$  defines the discrete dynamics on the jump set  $D$ . These objects are referred to as the *data* of the hybrid system  $\mathcal{H}$ , which at times is explicitly denoted as  $\mathcal{H} = (C, f, D, g)$ . We illustrate this framework in a simple, yet rich in behavior, hybrid system.

**Example 1.1** (bouncing ball system) Consider a model for a bouncing ball written as

$$f(x) := \begin{bmatrix} x_2 \\ -\gamma \end{bmatrix}, \quad C := \{x \in \mathbb{R}^2 \mid x_1 \geq 0\} \quad (2)$$

$$g(x) := \begin{bmatrix} 0 \\ -\lambda x_2 \end{bmatrix}, \quad D := \{x \in \mathbb{R}^2 \mid x_1 \leq 0, x_2 \leq 0\} \quad (3)$$

where  $\gamma > 0$  is the gravity constant and  $\lambda \in [0, 1)$  is the restitution coefficient. In this model, we consider the ball to be bouncing on a floor at a height of 0. This model is re-visited as an example in Section 3 and Section 5. □

The remainder of this note is organized as follows. In Section 2, we describe how to install the HyEQ Toolbox in MATLAB. In Section 3, we introduce the Lite HyEQ Simulator for solving hybrid systems without inputs. In Section 4, we introduce the HyEQ Simulator implemented in Simulink for solving single and interconnected hybrid systems with inputs. In Section 5, we work through several examples for the simulation of single and interconnected hybrid systems. In Section 6, we give directions to where the simulator files can be downloaded.

## 2 Installation

The following procedure describes how to install the Hybrid Equations (HyEQ) Toolbox in MATLAB. This installation adds useful `.m` files to the MATLAB library and several blocks to the Simulink block library.

Steps for installation:

1. Download the HyEQ Toolbox from MATLAB Central or the author's website at <https://hybrid.soe.ucsc.edu/software>.
2. Extract all files and save in any place (except the root folder).
3. Open MATLAB and change the current folder to the folder where the `install.m` is located.
4. Type `install` in the command window and hit enter to run the file `install.m`.
5. Follow the on-screen prompts. Must answer yes to the question:

```
Add toolbox permanently into your startup path (highly recommended)? Y/E/N [Y]: y
```

6. Once installation has finished, close and then reopen MATLAB.

Now the HyEQ Toolbox is ready for use.

If you wish to uninstall the HyEQ Toolbox from MATLAB, simply run the `tbclean.m` file inside the `HyEQ_Toolbox_V2_04` folder, and follow the on-screen prompts.

## 3 Lite HyEQ Simulator: A stand-alone MATLAB code for simulation of hybrid systems without inputs

One way to simulate hybrid systems is to use ODE function calls with events in MATLAB. Such an implementation gives fast simulation of a hybrid system.

In the lite HyEQ solver, four basic functions are used to define the *data* of the hybrid system  $\mathcal{H}$  as in (1) (without inputs):

- The flow map is defined in the MATLAB function `f.m`. The input to this function is a vector with components defining the state of the system  $x$ . Its output is the value of the flow map  $f$ .
- The flow set is defined in the MATLAB function `C.m`. The input to this function is a vector with components defining the state of the system  $x$ . Its output is equal to 1 if the state belongs to the set  $C$  or equal to 0 otherwise.
- The jump map is defined in the MATLAB function `g.m`. Its input is a vector with components defining the state of the system  $x$ . Its output is the value of the jump map  $g$ .
- The jump set is defined in the MATLAB function `D.m`. Its input is a vector with components defining the state of the system  $x$ . Its output is equal to 1 if the state belongs to  $D$  or equal to 0 otherwise.

Our Lite HyEQ Simulator uses a main function `run.m` to initialize, run, and plot solutions for the simulation, functions `f.m`, `C.m`, `g.m`, and `D.m` to implement the data of the hybrid system, and `HyEQsolver.m` which will solve the differential equations by integrating the continuous dynamics,  $\dot{x} = f(x)$ , and jumping by the update law  $x^+ = g(x)$ . The ODE solver called in `HyEQsolver.m` initially uses the initial or most recent step size, and after each integration, the algorithms in `HyEQsolver.m` check to see if the solution is in the set  $C$ ,  $D$ , or neither. Depending on which set the solution is in, the simulation is accordingly reset following the dynamics given in  $f$  or  $g$ , or the simulation is stopped. This implementation is fast because it also does not store variables to the workspace and only uses built-in ODE function calls.

Time and jump horizons are set for the simulation using `TSPAN = [TSTART TFINAL]` as the time interval of the simulation and `JSPAN = [JSTART JSTOP]` as the interval for the number of discrete jumps allowed. The simulation stops when either the time or jump horizon, i.e. the final value of either interval, is reached.

The example below shows how to use the HyEQ solver to simulate a bouncing ball.

**Example 3.1** (bouncing ball with Lite HyEQ Solver) Consider the hybrid system model for the bouncing ball with data given in Example 1.1.

For this example, we consider the ball to be bouncing on a floor at zero height. The constants for the bouncing ball system are  $\gamma = 9.81$  and  $\lambda = 0.8$ . The following procedure is used to simulate this example in the Lite HyEQ Solver:

- Inside the MATLAB script `run.m`, initial conditions, simulation horizons, a rule for jumps, ode solver options, and a step size coefficient are defined. The function `HyEQsolver.m` is called in order to run the simulation, and a script for plotting solutions is included.
- Then the MATLAB functions `f.m`, `C.m`, `g.m`, `D.m` are edited according to the data given above.
- Finally, the simulation is run by clicking the run button in `run.m` or by calling `run.m` in the MATLAB command window.

Example code for each of the MATLAB files `run.m`, `f.m`, `C.m`, `g.m`, and `D.m` is given below.

```

1  %-----
2  % Matlab M-file Project: HyEQ Toolbox @ Hybrid Systems Laboratory (HSL),
3  % https://hybrid.soe.ucsc.edu/software
4  % http://hybridsimulator.wordpress.com/
5  % Filename: run_ex1_2a.m
6  %-----
7  % Project: Simulation of a hybrid system (bouncing ball)
8  %-----
9  %-----
10 % See also HYEQSOLVER, PLOTARC, PLOTARC3, PLOTFLows, PLOTHARC,
11 % PLOTHARCCOLOR, PLOTHARCCOLOR3D, PLOTHYBRIDARC, PLOTJUMPS.
12 % Copyright @ Hybrid Systems Laboratory (HSL),
13 % Revision: 0.0.0.3 Date: 05/20/2015 3:42:00
14
15 function run_ex1_2a
16 % initial conditions
17 x1_0 = 1;
18 x2_0 = 0;
19 x0 = [x1_0;x2_0];
20
21 % physical variables
22 global gamma lambda
23 gamma = -9.81; % gravity constant
24 lambda = 0.8; % restitution coefficient
25
26 % simulation horizon
27 TSPAN=[0 10];
28 JSPAN = [0 20];
29
30 % rule for jumps
31 % rule = 1 -> priority for jumps
32 % rule = 2 -> priority for flows
33 rule = 1;
34
35 options = odeset('RelTol',1e-6,'MaxStep',.1);
36
37 % simulate
38 [t j x] = HyEQsolver( @f_ex1_2a,@g_ex1_2a,@C_ex1_2a,@D_ex1_2a,...
39 x0,TSPAN,JSPAN,rule,options);

```

```

40
41 % plot solution
42 figure(1) % position
43 clf
44 subplot(2,1,1), plotHarc(t,j,x(:,1));
45 grid on
46 ylabel('x_1 position')
47 subplot(2,1,2), plotHarc(t,j,x(:,2));
48 grid on
49 ylabel('x_2 velocity')
50
51 % plot phase plane
52 figure(2) % position
53 clf
54 plotHarcColor(x(:,1),j,x(:,2),t);
55 xlabel('x_1')
56 ylabel('x_2')
57 grid on
58
59 % plot hybrid arc
60 figure(3)
61 plotHybridArc(t,j,x)
62 xlabel('j')
63 ylabel('t')
64 zlabel('x1')
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905

```

```

2  %-----
3  % Matlab M-file Project: HyEQ Toolbox @ Hybrid Systems Laboratory (HSL),
4  % https://hybrid.soe.ucsc.edu/software
5  % http://hybridsimulator.wordpress.com/
6  % Filename: C_ex1_2a.m
7  %-----
8  % Description: Flow set
9  % Return 0 if outside of C, and 1 if inside C
10 %-----
11 %-----
12 % See also HYEQSOLVER, PLOTARC, PLOTARC3, PLOTFLows, PLOTHARC,
13 % PLOTHARCCOLOR, PLOTHARCCOLOR3D, PLOTHYBRIDARC, PLOTJUMPS.
14 % Copyright @ Hybrid Systems Laboratory (HSL),
15 % Revision: 0.0.0.3 Date: 05/20/2015 3:42:00
16
17
18 x1 = x(1);
19
20 if x1 >= 0
21     value = 1;
22 else
23     value = 0;
24 end
25 end

1 function xplus = g_ex1_2a(x)
2 %-----
3 % Matlab M-file Project: HyEQ Toolbox @ Hybrid Systems Laboratory (HSL),
4 % https://hybrid.soe.ucsc.edu/software
5 % http://hybridsimulator.wordpress.com/
6 % Filename: g_ex1_2a.m
7 %-----
8 % Project: Simulation of a hybrid system (bouncing ball)
9 % Description: Jump map
10 %-----
11 %-----
12 % See also HYEQSOLVER, PLOTARC, PLOTARC3, PLOTFLows, PLOTHARC,
13 % PLOTHARCCOLOR, PLOTHARCCOLOR3D, PLOTHYBRIDARC, PLOTJUMPS.
14 % Copyright @ Hybrid Systems Laboratory (HSL),
15 % Revision: 0.0.0.3 Date: 05/20/2015 3:42:00
16
17 % state
18 x1 = x(1);
19 x2 = x(2);
20
21 global lambda
22
23 xplus = [-x1 ; -lambda*x2];
24 end

1 function inside = D_ex1_2a(x)
2 %-----
3 % Matlab M-file Project: HyEQ Toolbox @ Hybrid Systems Laboratory (HSL),
4 % https://hybrid.soe.ucsc.edu/software

```

```

5 % http://hybridsimulator.wordpress.com/
6 % Filename: D_ex1_2a.m
7 %-----
8 % Description: Jump set
9 % Return 0 if outside of D, and 1 if inside D
10 %-----
11 %-----
12 % See also HYEQSOLVER, PLOTARC, PLOTARC3, PLOTFLAWS, PLOTHARC,
13 % PLOTHARCCOLOR, PLOTHARCCOLOR3D, PLOTHYBRIDARC, PLOTJUMPS.
14 % Copyright © Hybrid Systems Laboratory (HSL),
15 % Revision: 0.0.0.3 Date: 05/20/2015 3:42:00
16
17 x1 = x(1);
18 x2 = x(2);
19 if (x1 <= 0 && x2 <= 0)
20     inside = 1;
21 else
22     inside = 0;
23 end
24 end

```

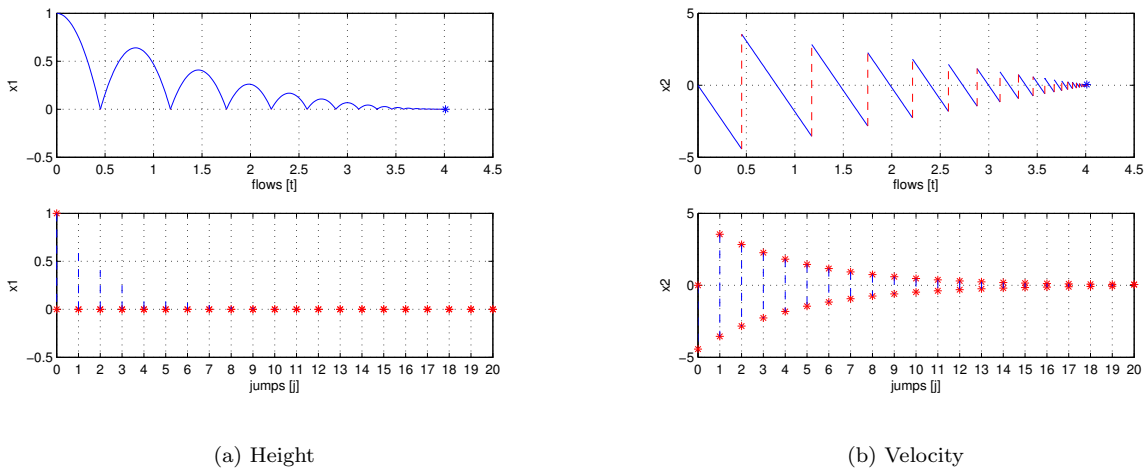


Figure 1: Solution of Example 3.1

A solution to the bouncing ball system from  $x(0,0) = [1,0]^T$  and with  $TSPAN = [0 \ 10]$ ,  $JSPAN = [0 \ 20]$ ,  $rule = 1$ , is depicted in Figure 1(a) (height) and Figure 1(b) (velocity). Both the projection onto  $t$  and  $j$  are shown. Figure 2 depicts the corresponding hybrid arc for the position state.

For MATLAB files of this example, see Examples/Example\_1.2.

□

### 3.1 Solver Function

The solver function `HyEqSolver` solves the hybrid system using three different functions as shown below. First, the flows are calculated using the built-in ODE solver function `ODE45` in MATLAB. If the solution leaves the flow set  $C$ , the discrete event is detected using the function `zeroevents` as shown in Section 3.1.1. When the state jumps, the next value of the state is calculated via the jump map  $g$  using the function `jump`

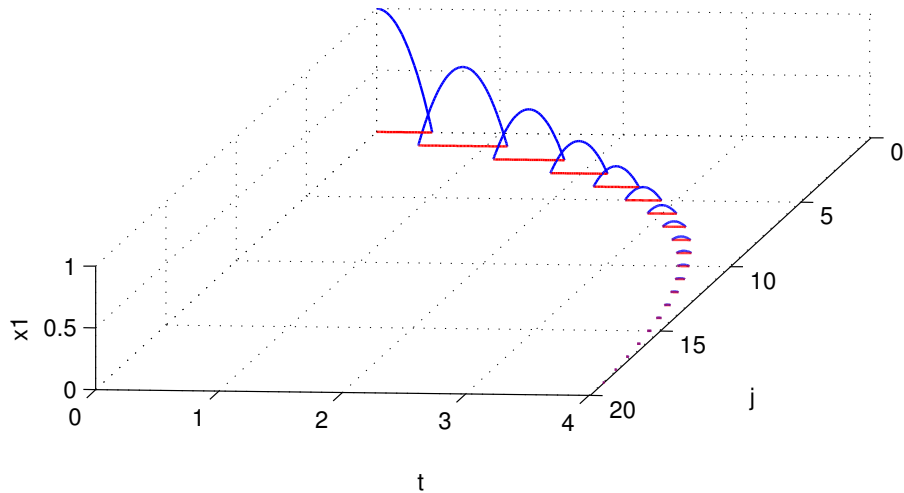


Figure 2: Hybrid arc corresponding to a solution of Example 3.1: height

as shown in Section 3.1.2.

```

1 function [t j x] = HyEQsolver(f,g,C,D,x0,TSPAN,JSPAN,rule,options,solver,E)
2 %HYEQSOLVER solves hybrid equations.
3 % Syntax: [t j x] = HYEQSOLVER(f,g,C,D,x0,TSPAN,JSPAN,rule,options,solver,E)
4 % computes solutions to the hybrid equations
5 %
6 %  $\dot{x} = f(x,t,j)$   $x \in C$   $x^+ = g(x,t,j)$   $x \in D$ 
7 %
8 % where  $x$  is the state,  $f$  is the flow map,  $g$  is the jump map,  $C$  is the
9 % flow set, and  $D$  is the jump set. It outputs the state trajectory  $(t,j)$ 
10 %  $\rightarrow x(t,j)$ , where  $t$  is the flow time parameter and  $j$  is the jump
11 % parameter.
12 %
13 %  $x_0$  defines the initial condition for the state.
14 %
15 % TSPAN = [TSTART TFINAL] is the time interval. JSPAN = [JSTART JSTOP] is
16 % the interval for discrete jumps. The algorithm stop when the first
17 % stop condition is reached.
18 %
19 % rule (optional parameter) - rule for jumps
20 % rule = 1 (default)  $\rightarrow$  priority for jumps rule = 2  $\rightarrow$  priority for
21 % flows
22 %
23 % options (optional parameter) - options for the solver see odeset f.ex.
24 % options = odeset('RelTol',1e-6);

```



```

25 %     options = odeset('InitialStep',eps);
26 %
27 % solver (optional parameter. String) - selection of the desired ode
28 % solver. All ode solvers are supported, except for odel5i. See help
29 % odeset for detailed information.
30 %
31 % E (optional parameter) - Mass matrix [constant matrix | function_handle]
32 % For problems:
33 %  $E \cdot \dot{x} = f(x) \quad x \in C$ 
34 %  $x^+ = g(x) \quad x \in D$ 
35 % set this property to the value of the constant mass matrix. For
36 % problems with time- or state-dependent mass matrices, set this
37 % property to a function that evaluates the mass matrix. See help
38 % odeset for detailed information.
39 %
40 % Example: Bouncing ball with Lite HyEQ Solver
41 %
42 % % Consider the hybrid system model for the bouncing ball with data given in
43 % % Example 1.2. For this example, we consider the ball to be bouncing on a
44 % % floor at zero height. The constants for the bouncing ball system are
45 % %  $\gamma=9.81$  and  $\lambda=0.8$ . The following procedure is used to
46 % % simulate this example in the Lite HyEQ Solver:
47 %
48 % % * Inside the MATLAB script run_ex1_2.m, initial conditions, simulation
49 % % horizons, a rule for jumps, ode solver options, and a step size
50 % % coefficient are defined. The function HYEQSOLVER.m is called in order to
51 % % run the simulation, and a script for plotting solutions is included.
52 % % * Then the MATLAB functions f_ex1_2.m, C_ex1_2.m, g_ex1_2.m, D_ex1_2.m
53 % % are edited according to the data given below.
54 % % * Finally, the simulation is run by clicking the run button in
55 % % run_ex1_2.m or by calling run_ex1_2.m in the MATLAB command window.
56 %
57 % % For further information, type in the command window:
58 % web(['Example_1_2.html']);
59 %
60 % % Define initial conditions
61 % x1_0 = 1;
62 % x2_0 = 0;
63 % x0 = [x1_0; x2_0];
64 %
65 % % Set simulation horizon
66 % TSPAN = [0 10];
67 % JSPAN = [0 20];
68 %
69 % % Set rule for jumps and ODE solver options
70 %
71 % % rule = 1 -> priority for jumps
72 %
73 % % rule = 2 -> priority for flows
74 %
75 % % set the maximum step length. At each run of the
76 % % integrator the option 'MaxStep' is set to
77 % % (time length of last integration)*maxStepCoefficient.
78 % % Default value = 0.1

```

```

79 %
80 %     rule                = 1;
81 %
82 %     options            = odeset('RelTol',1e-6,'MaxStep',.1);
83 %
84 %     % Simulate using the HYEQSOLVER script
85 %     % Given the matlab functions that models the flow map, jump map,
86 %     % flow set and jump set (f_ex1_2, g_ex1_2, C_ex1_2, and D_ex1_2
87 %     % respectively)
88 %
89 %     [t j x] = HYEQSOLVER( @f_ex1_2,@g_ex1_2,@C_ex1_2,@D_ex1_2,...
90 %         x0,TSPAN,JSPAN,rule,options,'ode45');
91 %
92 %     % plot solution
93 %
94 %     figure(1) % position
95 %     clf
96 %     subplot(2,1,1),plotflows(t,j,x(:,1))
97 %     grid on
98 %     ylabel('x1')
99 %
100 %     subplot(2,1,2),plotjumps(t,j,x(:,1))
101 %     grid on
102 %     ylabel('x1')
103 %
104 %     figure(2) % velocity
105 %     clf
106 %     subplot(2,1,1),plotflows(t,j,x(:,2))
107 %     grid on
108 %     ylabel('x2')
109 %
110 %     subplot(2,1,2),plotjumps(t,j,x(:,2))
111 %     grid on
112 %     ylabel('x2')
113 %
114 %     % plot hybrid arc
115 %
116 %     figure(3)
117 %     plotHybridArc(t,j,x)
118 %     xlabel('j')
119 %     ylabel('t')
120 %     zlabel('x1')
121 %
122 %     % plot solution using plotHarc and plotHarcColor
123 %
124 %     figure(4) % position
125 %     clf
126 %     subplot(2,1,1), plotHarc(t,j,x(:,1));
127 %     grid on
128 %     ylabel('x_1 position')
129 %     subplot(2,1,2), plotHarc(t,j,x(:,2));
130 %     grid on
131 %     ylabel('x_2 velocity')
132 %

```

```

133 %
134 %     % plot a phase plane
135 %     figure(5) % position
136 %     clf
137 %     plotHarcColor(x(:,1),j,x(:,2),t);
138 %     xlabel('x_1')
139 %     ylabel('x_2')
140 %     grid on
141 %
142 %-----
143 % Matlab M-file Project: HyEQ Toolbox @ Hybrid Systems Laboratory (HSL),
144 % https://hybrid.soe.ucsc.edu/software
145 % http://hybridsimulator.wordpress.com/
146 % Filename: HYEQSOLVER.m
147 %-----
148 % See also HYEQSOLVER, PLOTARC, PLOTARC3, PLOTFLAWS, PLOTHARC,
149 % PLOTHARCCOLOR, PLOTHARCCOLOR3D, PLOTHYBRIDARC, PLOTJUMPS.
150 % Copyright @ Hybrid Systems Laboratory (HSL),
151 % Revision: 0.0.0.4 Date: 04/6/2017 16:26:00
152
153
154 if ~exist('rule','var')
155     rule = 1;
156 end
157
158 if ~exist('options','var')
159     options = odeset();
160 end
161 if exist('E','var') && ~exist('solver','var')
162     solver = 'ode15s';
163 end
164 if ~exist('solver','var')
165     solver = 'ode45';
166 end
167 if ~exist('E','var')
168     E = [];
169 end
170 % mass matrix (if existent)
171 isDAE = false;
172 if ~isempty(E)
173     isDAE = true;
174     switch isa(E,'function_handle')
175     case true % Function E(x)
176         M = E;
177         options = odeset(options,'Mass',M,'Stats','off',...
178             'MassSingular','maybe','MStateDependence','strong',...
179             'InitialSlope',f_hdae(x0,TSPAN(1)));
180     case false % Constant double matrix
181         M = double(E);
182         options = odeset(options,'Mass',M,'Stats','off',...
183             'MassSingular','maybe','MStateDependence','none');
184     end
185 end
186

```

```

187 odeX = str2func(solver);
188 nargf = nargin(f);
189 nargg = nargin(g);
190 nargC = nargin(C);
191 nargD = nargin(D);
192
193
194
195 % simulation horizon
196 tstart = TSPAN(1);
197 tfinal = TSPAN(end);
198 jout = JSPAN(1);
199 j = jout(end);
200
201 % simulate
202 tout = tstart;
203 [rx,cx] = size(x0);
204 if rx == 1
205     xout = x0;
206 elseif cx == 1
207     xout = x0.';
208 else
209     error('Error, x0 does not have the proper size')
210 end
211
212 % Jump if jump is prioritized:
213 if rule == 1
214     while (j<JSPAN(end))
215         % Check if value it is possible to jump current position
216         insideD = fun_wrap(xout(end,:)','tout(end),j,D,nargD);
217         if insideD == 1
218             [j tout jout xout] = jump(g,j,tout,jout,xout,nargg);
219         else
220             break;
221         end
222     end
223 end
224 fprintf('Completed: %3.0f%%',0);
225 while (j < JSPAN(end) && tout(end) < TSPAN(end))
226     options = odeset(options,'Events',@(t,x) zeroevents(x,t,j,C,D,...
227         rule,nargC,nargD));
228     % Check if it is possible to flow from current position
229     insideC = fun_wrap(xout(end,:)','tout(end),j,C,nargC);
230     if insideC == 1
231         if isDAE
232             options = odeset(options,'InitialSlope',f(xout(end,:)','tout(end)));
233         end
234         [t,x] = odeX(@(t,x) fun_wrap(x,t,j,f,nargf),[tout(end) tfinal],...
235             xout(end,:)',' options);
236         nt = length(t);
237         tout = [tout; t];
238         xout = [xout; x];
239         jout = [jout; j*ones(1,nt)'];
240     end

```

```

241
242 %Check if it is possible to jump
243 insideD = fun_wrap(xout(end,:),tout(end),j,D,nargD);
244 if insideD == 0
245     break;
246 else
247     if rule == 1
248         while (j<JSPAN(end))
249             % Check if it is possible to jump from current position
250             insideD = fun_wrap(xout(end,:),tout(end),j,D,nargD);
251             if insideD == 1
252                 [j tout jout xout] = jump(g,j,tout,jout,xout,nargg);
253             else
254                 break;
255             end
256         end
257     else
258         [j tout jout xout] = jump(g,j,tout,jout,xout,nargg);
259     end
260 end
261 fprintf('\b\b\b\b\b%3.0f%%',max(100*j/JSPAN(end),100*tout(end)/TSPAN(end)));
262 end
263 t = tout;
264 x = xout;
265 j = jout;
266 fprintf('\nDone\n');
267 end
268

```

### 3.1.1 Events Detection

```

1 function [value,isterminal,direction] = zeroevents(x,t,j,C,D,rule,nargC,nargD)
2 switch rule
3     case 1 % -> priority for jumps
4         isterminal(1) = 1; % InsideC
5         isterminal(2) = 1; % Inside(C \cap D)
6         isterminal(3) = 1; % OutsideC
7         direction(1) = -1; % InsideC
8         direction(2) = -1; % Inside(C \cap D)
9         direction(3) = 1; % OutsideC
10    case 2 %(default) -> priority for flows
11        isterminal(1) = 1; % InsideC
12        isterminal(2) = 0; % Inside(C \cap D)
13        isterminal(3) = 1; % OutsideC
14        direction(1) = -1; % InsideC
15        direction(2) = -1; % Inside(C \cap D)
16        direction(3) = 1; % OutsideC
17 end
18
19 insideC = fun_wrap(x,t,j,C,nargC);
20 insideD = fun_wrap(x,t,j,D,nargD);
21 outsideC = -fun_wrap(x,t,j,C,nargC);
22

```

```

23
24 value(1) = 2*insideC;
25 value(2) = 2-insideC - insideD;
26 value(3) = 2*outsideC;
27
28 end
29

```

### 3.1.2 Jump Map

```

1 function [j tout jout xout] = jump(g, j, tout, jout, xout, nargsfun)
2 % Jump
3 j = j+1;
4 y = fun_wrap(xout(end, :).', tout(end), jout(end), g, nargsfun);
5 % Save results
6 tout = [tout; tout(end)];
7 xout = [xout; y.'];
8 jout = [jout; j];
9 end
10

```

### 3.1.3 Function Wrapper

```

1 function xdelta = fun_wrap(x,t,j,h,nargsfun)
2 %fun_wrap Variable input arguments function (easy use for users).
3 % fun_wrap(x,t,j,h,nargsfun) depending on the function h written by the
4 % user, this script selects how the HyEQ solver should call that
5 % function.
6 % x: state
7 % t: time
8 % j: discrete time
9 % h: function handle
10 % nargsfun: number of input arguments of function h
11 %-----
12 % Matlab M-file Project: HyEQ Toolbox @ Hybrid Systems Laboratory (HSL),
13 % https://hybrid.soe.ucsc.edu/software
14 % http://hybridsimulator.wordpress.com/
15 % Filename: fun_wrap.m
16 %-----
17 % See also HYEQSOLVER, PLOTARC, PLOTARC3, PLOTFLows, PLOTHARC,
18 % PLOTHARCCOLOR, PLOTHARCCOLOR3D, PLOTHYBRIDARC, PLOTJUMPS.
19 % Copyright @ Hybrid Systems Laboratory (HSL),
20 % Revision: 0.0.0.3 Date: 01/28/2016 5:12:00
21
22
23 switch nargsfun
24 case 1
25     xdelta = h(x);
26 case 2
27     xdelta = h(x,t);
28 case 3
29     xdelta = h(x,t,j);
30 end

```

## 3.2 Software Requirements

In order to run simulations using the Lite HyEQ Simulator, MATLAB R13 or newer is required.

## 3.3 Configuration of Solver

Before a simulation is started, it is important to determine the needed integrator scheme, zero-cross detection settings, precision, and other tolerances. Using the default settings does not always give the most efficient or most accurate simulations. In the Lite HyEQ Simulator, these parameters are edited in the `run.m` file using

```
options = odeset(RelTol,1e-6,MaxStep ,.1);
```

## 3.4 Initialization

The Lite HyEQ Simulator is initialized and run by calling the function `run.m`. Inside `run.m`, the initial conditions, simulation horizons `TSPAN` and `JSPAN`, a rule for jumps, and simulation tolerances are defined. After all of the parameters are defined, the function `HyEQsolver` is called, and the simulation runs. See below for sample code to initialize and run the bouncing ball example, Example 3.1.

```
1 % initial conditions
2 x1_0 = 1;
3 x2_0 = 0;
4 x0 = [x1_0;x2_0];
5 % simulation horizon
6 TSPAN=[0,10];
7 JSPAN = [0,20];
8 % rule for jumps
9 % rule = 1 -> priority for jumps
10 % rule = 2 -> priority for flows
11 rule = 1;
12 options = odeset('RelTol',1e-6,'MaxStep',.1);
13 % simulate
14 [t,j,x] = HyEQsolver(@f,@g,@C,@D,x0,TSPAN,JSPAN,rule,options);
```

## 3.5 Postprocessing and Plotting solutions

The function `run.m` is also used to plot solutions after the simulations is complete. See below for sample code to plot solutions to the bouncing ball example, Example 3.1.

```
1 % plot solution
2 figure(1) % position
3 clf
4 subplot(2,1,1),plotflows(t,j,x(:,1))
5 grid on
6 ylabel('x1')
7 subplot(2,1,2),plotjumps(t,j,x(:,1))
8 grid on
9 ylabel('x1')
10 figure(2) % velocity
11 clf
12 subplot(2,1,1),plotflows(t,j,x(:,2))
13 grid on
```

```

14 ylabel('x2')
15 subplot(2,1,2),plotjumps(t,j,x(:,2))
16 grid on
17 ylabel('x2')
18 % plot hybrid arc
19 figure(2)
20 plotHybridArc(t,j,x)
21 xlabel('j')
22 ylabel('t')
23 zlabel('x1')
24 grid on
25 view(37.5,30)

```

The following functions are used to generate the plots:

- `plotarc(t,j,x,L,jstar,modifierF,modifierJ,resolution,DDD,true3D)`: plots the hybrid time domain (matrix)  $(t, j)$  versus the state  $x$  (matrix) taking into account jumps  $j$ . If  $x$  is a matrix ( $n$  states), then the hybrid time is plotted versus the rows or columns of the matrix, whichever line up. If  $t$  and  $j$  are matrices, then each column of  $x$  will be plotted according to the hybrid time domain composed for each column of  $t$  and  $j$ . Depending on the input data, this function is capable of plotting several types of figures, e.g., 2D, 3D, hybrid arcs with color, etc. Next, we list several functions that specialize different types of plotting styles from `plotarc`. For more information, please type `>> help plotarc` or `>> helpwin plotarc` in the command window.
- `plotarc3(t,j,x,L,jstar,modifierF,modifierJ,resolution,true3D)` is a version of `plotarc` that specializes in figures in 3D. For more information, please type `>> help plotarc3` or `>> helpwin plotarc3` in the command window.
- `plotflows(t,j,x,jstar,resolution)`: plots (in blue) the projection of the trajectory  $x$  onto the flow time axis  $t$ . The value of the trajectory for intervals  $[t_j, t_{j+1}]$  with empty interior is marked with `*` (in blue). Dashed lines (in red) connect the value of the trajectory before and after the jump. Figure 11(a) shows a plot created with this function.
  - `plotflows(t,j,x,jstar)`: The plot is cut regarding the  $jstar$  interval ( $jstar = [j - initialj - final]$ ).
  - `plotflows(t,j,x,jstar,resolution)`: Also, a maximum resolution in between jumps is given by the input variable resolution
- `plotjumps(t,j,x,jstar,resolution)`: plots (in red) the projection of the trajectory  $x$  onto the jump time  $j$ . The initial and final value of the trajectory on each interval  $[t_j, t_{j+1}]$  is denoted by `*` (in red) and the continuous evolution of the trajectory on each interval is depicted with a dashed line (in blue). Figure 11(a) shows a plot created with this function.
  - `plotjumps(t,j,x,jstar)`: The plot is cut regarding the  $jstar$  interval ( $jstar = [j - initialj - final]$ ).
  - `plotjumps(t,j,x,jstar,resolution)`: Also, a maximum resolution in between jumps is given by the input variable resolution
- `plotHybridArc(t,j,x,jstar,resolution)`: plots (in blue and red) the trajectory  $x$  on hybrid time domains. The intervals  $[t_j, t_{j+1}]$  indexed by the corresponding  $j$  are depicted in the  $t-j$  plane (in red). Figure 12 shows a plot created with this function.
  - `plotHybridArc(t,j,x,jstar)`: The plot is cut regarding the  $jstar$  interval ( $jstar = [j - initialj - final]$ ).
  - `plotHybridArc(t,j,x,jstar,resolution)`: Also, a maximum resolution in between jumps is given by the input variable resolution
- `plotHarc(t,j,x,jstar,modifierF,modifierJ,resolution)` is a function for plotting hybrid arcs ( $n$  states).



- plotHarc(t,j,x): plots the trajectory  $x$  versus the hybrid time domain  $(t, j)$ . If  $x$  is a matrix, then the time vector is plotted versus the rows or columns of the matrix, whichever line up.
- plotHarc(t,j,x,jstar): plots the trajectory  $x$  versus the hybrid time domain  $(t, j)$ , and the plot is cut regarding the  $jstar$  interval ( $jstar = [j_{initial}, j_{final}]$ ).
- plotHarc(t,j,x,jstar,modifierF,modifierJ): ModifierF and ModifierJ are cell arrays that contains the standard matlab plotting modifiers (type >> help plotHarc or >> helpwin plotHarc in the command window for more information).
- plotHarcColor(t,j,x,L,jstar,resolution) plots the trajectory  $x$  on hybrid time domain with color.
  - plotHarcColor(t,j,x,L): plots the trajectory  $x$  (vector) versus the hybrid time domain  $(t, j)$ . The hybrid arc is plotted with  $L$  data as color. The input vectors  $t, j, x, L$  must have the same length.
  - plotHarcColor(t,j,x,L,jstar): If a specific interval in  $j$  is required,  $jstar = [j_{initial}, j_{final}]$  must be provided. (type >> help plotHarcColor or >> helpwin plotHarcColor in the command window for more information)
- plotHarcColor3D(t,j,x,L,jstar,modifier,resolution) plots an  $3D$  hybrid arc with color.
  - plotHarcColor3D(t,j,x,L) plots the trajectory  $x$  (3 states) taking into account the hybrid time domain  $(t, j)$ . The hybrid arc is plotted with  $L$  data as color. The input vectors  $t, j, x, L$  must have the same length and  $x$  must have three columns.
  - plotHarcColor3D(t,j,x,L,jstar) If a specific interval in  $j$  is required,  $jstar = [j_{initial}, j_{final}]$  must be provided.
  - plotHarcColor3D(t,j,x,L,jstar,modifier) Modifier is a cell array that contains the standard matlab plotting modifiers (type >> help plotHarcColor3D or >> helpwin plotHarcColor3D in the command window for more information).

## 4 HyEQ Simulator: A Simulink implementation for simulation of single and interconnected hybrid systems with or without inputs

The HyEQ Toolbox includes three main Simulink library blocks that allow for simulation of a hybrid system  $\mathcal{H} = (C, f, D, g)$  using either externally defined functions or embedded MATLAB functions, and a single hybrid system or interconnected hybrid systems with inputs using embedded MATLAB functions. Figure 3 shows these blocks in the Simulink Library Browser.

Figure 4 shows a Simulink implementation for simulating a hybrid system with inputs using embedded MATLAB functions. In this implementation, four basic blocks are used to define the *data* of the hybrid system  $\mathcal{H}$ :

- The flow map is implemented in an *Embedded MATLAB function block* executing the function `f.m`. Its input is a vector with components defining the state of the system  $x$ , and the input  $u$ . Its output is the value of the flow map  $f$  which is connected to the input of an integrator.
- The flow set is implemented in an *Embedded MATLAB function block* executing the function `C.m`. Its input is a vector with components  $x^-$  and input  $u$  of the *Integrator system*. Its output is equal to 1 if the state belongs to the set  $C$  or equal to 0 otherwise. The minus notation denotes the previous value of the variables (before integration). The value  $x^-$  is obtained from the state port of the integrator.
- The jump map is implemented in an *Embedded MATLAB function block* executing the function `g.m`. Its input is a vector with components  $x^-$  and input  $u$  of the *Integrator system*. Its output is the value of the jump map  $g$ .

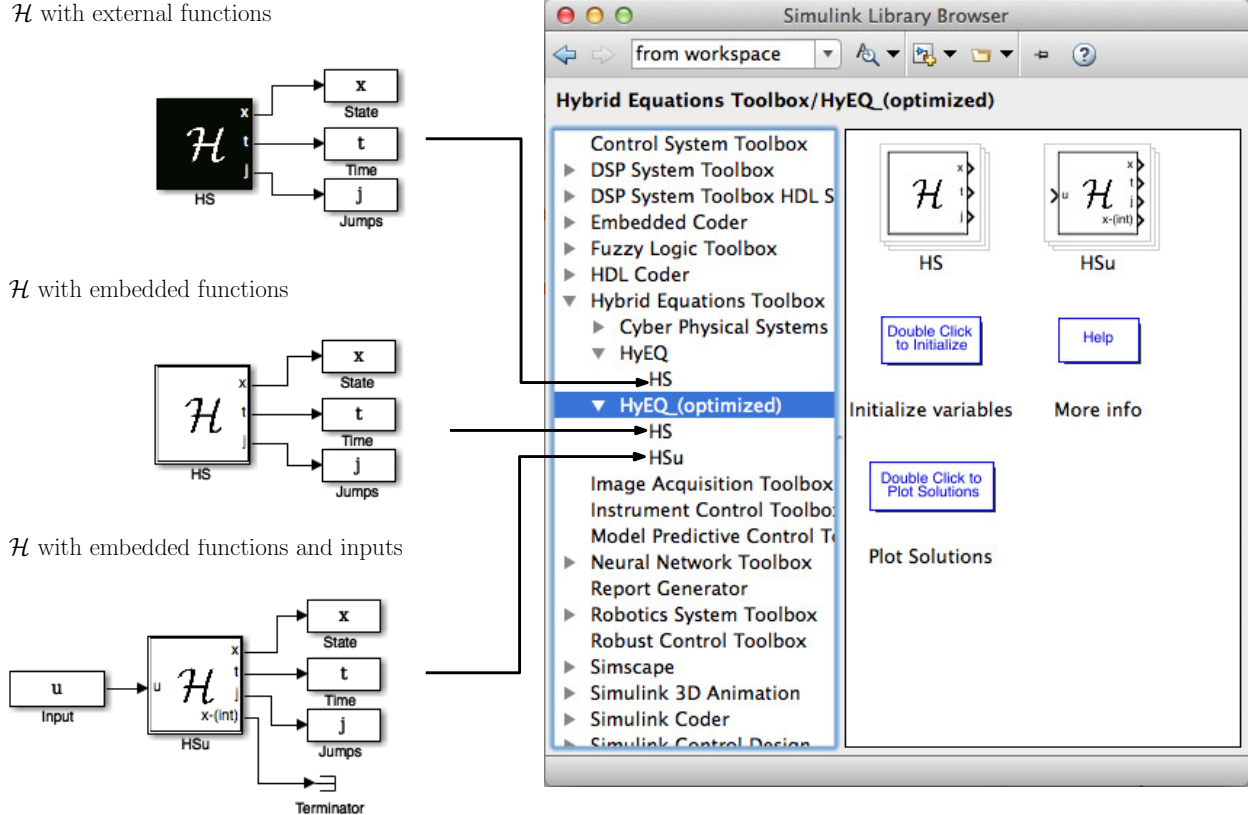


Figure 3: MATLAB/Simulink library blocks for Simulink implementation.

- The jump set is implemented in an *Embedded MATLAB function block* executing the function `D.m`. Its input is a vector with components  $x^-$  and input  $u$  of the *Integrator system*. Its output is equal to 1 if the state belongs to  $D$  or equal to 0 otherwise.

In our implementation, MATLAB `.m` files are used. The file `initialization.m` is used to define initial variables before simulation. The file `postprocessing.m` is used to plot the solutions after a simulation is complete. These two `.m` files are called by double-clicking the *Double Click to...* blocks at the top of the Simulink Model (see Section 4.6 for more information on these `.m` files and their use).

#### 4.1 Specialized library for simulation of Cyber-Physical Systems

The HyEQ Toolbox includes a series of blocks that model elements of a cyber-physical system (CPS). Those models are special instances of the hybrid systems blocks described above, particularly the blocks that use embedded MATLAB functions. A cyber-physical system is given by the interconnection between a physical process, *the plant*, a computer algorithm used for control, *the controller*; and the subsystems needed to interconnect the plant and the controller, i.e., *the interfaces, converters, and signal conditioners*. Most of the elements described in this section are presented in an extended form in [5].

In these notes, the temporal evolution of the variables of a cyber-physical system are captured using dynamical models. In this document, we advocate that hybrid dynamical system models can be employed to capture the behavior of cyber-physical systems. More precisely, the evolution of the continuous variables is captured by *differential equations* while the evolution of the discrete variables is captured by *difference equations*. These equations are typically nonlinear due to the complexity of the dynamics of those variables. Furthermore, conditions determining the change of the continuous and discrete variables according to the

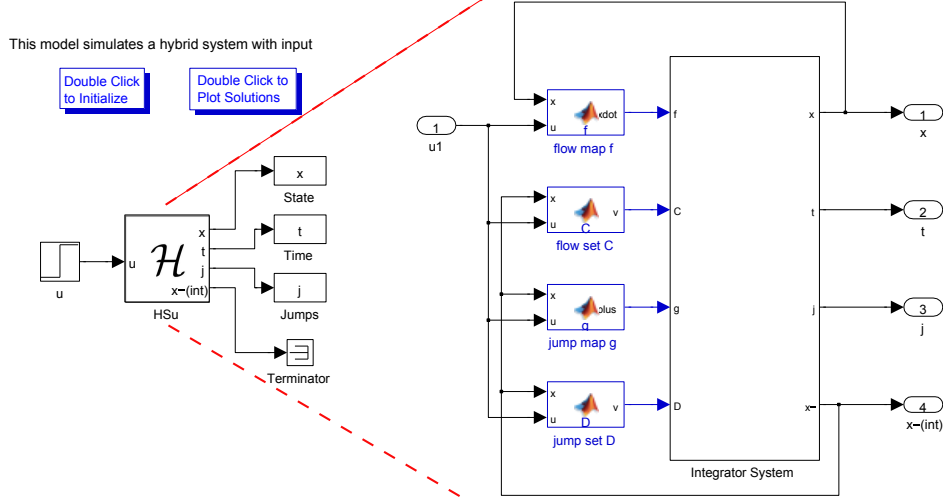


Figure 4: MATLAB/Simulink implementation of a hybrid system  $\mathcal{H} = (C, f, D, g)$  with inputs.

said equations/inclusions can be conveniently captured by functions of the variables, inputs, and outputs. In Figure 5, cyber and physical blocks are implemented within the HyEQ simulator.

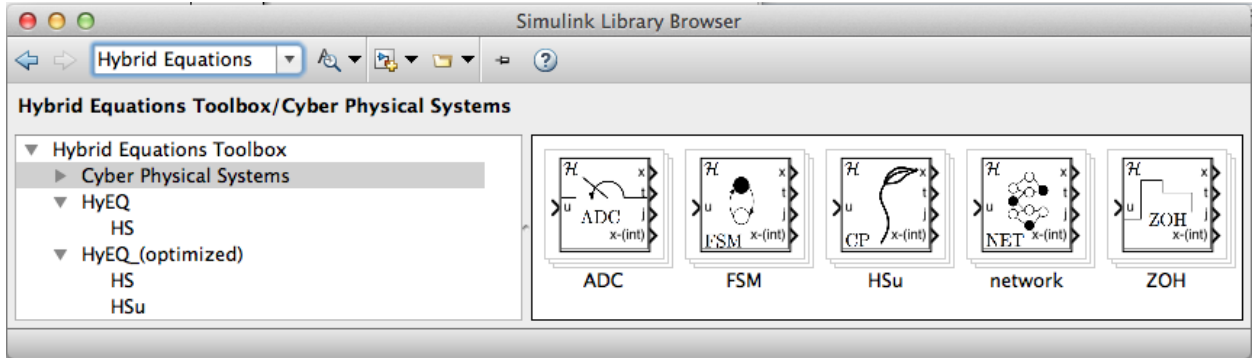


Figure 5: MATLAB/Simulink library blocks for Cyber-Physical Systems implementation.

#### 4.1.1 Models of physical components

The physical components of a cyber-physical system include the analog elements, physical systems, and the environment. Consider a model of a physical system given by a differential equation  $\dot{x} = f(x, u)$  and a output map  $y = h(x, u)$ . Among the many possible models available, we capture the dynamics of the physical components by using the hybrid systems framework by

$$\dot{x} = f(x, u), \quad C := \{x \mid x \in \mathbb{R}^n\}, \quad (4a)$$

$$x^+ = g(x) = \emptyset, \quad D := \emptyset, \quad (4b)$$

$$y = h(x, u) \quad (4c)$$

Particularly, the model in (4) can be implemented in the simulink block *HSu* in Figures 4) and 5).

### 4.1.2 Models of cyber components

The cyber components of a cyber-physical system include those in charge of performing computations, implementing algorithms, and transmitting digital data over networks. The tasks performed by the code (at the software level) and the logic-based mechanisms (at the circuit level) involve variables that only change at discrete events, not necessarily periodically.

We denote the state variable of the cyber components by  $x_c \in \Upsilon$ , where  $\Upsilon \subset \mathbb{R}^{n_c}$  is the state space. The dynamics of  $\eta$  are defined by a difference equation with right-hand side defined by the map  $G_c$ . We let  $v \in \mathcal{V} \subset \mathbb{R}^{m_c}$  denote the input signals affecting the cyber components and  $\zeta \in \mathbb{R}^{r_c}$  to be the output defined by the output function  $\kappa$ , which is a function of the state  $x_c$  and of the input  $v$ . With these definitions, the general mathematical description of the cyber component is

$$x_c^+ = G_c(x_c, v), \quad \zeta = \kappa(x_c, v) \quad (5)$$

In certain cases, it would be needed to impose restrictions on the state and inputs to the cyber component. Such conditions can be modeled imposing that  $x_c$  and  $v$  belong to a subset of their state space, namely,

$$(x_c, v) \in D_c \subset \Upsilon \times \mathcal{V} \quad (6)$$

The model of the cyber components is given by (5)-(6). Next, we provide specific constructions of models of cyber components.

- 1) *Pure Finite State Machines.* A finite state machine (FSM) or deterministic finite automaton (DFA) is a system with inputs, states, and outputs taking values from discrete sets that are updated at discrete transitions (or jumps) triggered by its inputs. Then, given a FSM and an initial state  $q_0 \in Q$ , a transition to a state  $q_1 = \delta(q_0, v)$  is performed when an input  $v \in \Sigma$  is applied to it. After the transition, the output of the FSM is updated to  $\kappa(q_1)$ . This mechanism can be captured by the difference equation

$$q^+ = \delta(q, v) \quad \zeta = \kappa(q) \quad (q, v) \in Q \times \Sigma \quad (7)$$

This model captures the dynamics the model of the cyber components in (5)-(6) with

$$x_c = q, \quad \Upsilon = Q, \quad \mathcal{V} = \Sigma, \quad G_c = \delta, \quad D_c = \Upsilon \times \mathcal{V}$$

Note that there is no notion of time associated with the FSM model above. An example of this model is presented in Example 5.8. In addition, the block FSM in Figure 5 can be used to model these type of systems.

- 2) *Analog-to-Digital Converters.* Analog-to-digital converters (ADCs), or simply sampling devices, are commonly used to provide measurements of the physical systems to the cyber components. Their main function is to sample their input, which is usually the output of the sensors measuring the output  $y$ , at a given periodic rate  $T_s^*$  and to make these samples available to the embedded computer. A basic model for a sampling device consists of a timer state and a sample state. When the timer reaches the value of the sampling time  $T_s^*$ , the timer is reset to zero and the sample state is updated with the inputs to the sampling device.

The model for the sampling device we propose has both continuous and discrete dynamics. If the timer state has not reached  $T_s^*$ , then the dynamics are such that the timer state increases continuously with a constant, unitary rate. When  $T_s^*$  is reached, the timer state is reset to zero and the sample state is mapped to the inputs of the sampling device. To implement this mechanism, we employ a timer state  $\tau_s \in \mathbb{R}_{\geq 0}$  and a sample state  $m_s \in \mathbb{R}^{r_p}$ . The input to the sampling device is denoted by  $v_s \in \mathbb{R}^{r_p}$ . The model of the sampling devices is

$$\dot{\tau}_s = 1, \quad \dot{m}_s = 0 \quad \text{when} \quad \tau_s \in [0, T_s^*] \quad (8)$$

$$\tau_s^+ = 0, \quad m_s^+ = v_s \quad \text{when} \quad \tau_s \geq T_s^* \quad (9)$$

In practice, there exists a time, usually called *the ADC acquisition time*, between the triggering of the ADC with the sampling device and the update of its output. Such a delay limits the number of samples per second that the ADC can provide. Additionally, an ADC can store and process finite-length digital words, which causes quantization. The model above omits effects such as acquisition delays and quantization effects, but those can be incorporated if needed. In particular, quantization effects can be added to the model in (8)-(9) by replacing the update law for  $m_s$  to  $m_s^+ = \text{round}(v_s)$ , where the function  $\text{round}$  is such that  $\text{round}(v_s)$  is the closest number to  $v_s$  that the machine precision can represent.

- 3 *Digital-to-Analog Converters.* The digital signals in the cyber components need to be converted to analog signals for their use in the physical world. Digital-to-analog converters (DACs) perform such a task by converging digital signals into analog equivalents. One of the most common models for a DAC is the zero-order hold model (ZOH). In simple terms, a ZOH converts a digital signal at its input into an analog signal at its output. Its output is updated at discrete time instants, typically periodically, and held constant in between updates, until new information is available at the next sampling time. We will model DACs as ZOH devices with dynamics similar to (8)-(9). Let  $\tau_h \in \mathbb{R}_{\geq 0}$  be the timer state,  $m_h \in \mathbb{R}^{r_c}$  be the sample state (note that the value of  $h$  indicates the number of DACs in the interface), and  $v_h \in \mathbb{R}^{r_c}$  be the inputs of the DAC. Its operation is as follows. When  $\tau_h \leq 0$ , the timer state is reset to  $\tau_r$  and the sample state is updated with  $v_h$  (usually the output of the embedded computer), where  $\tau_r \in [T^{\min}, T^{\max}]$  is a random variable that models the time in-between communication instants and  $T^{\min} \leq T^{\max}$ . A model that captures this mechanism is given by

$$\begin{aligned} \dot{\tau}_h &= -1, \quad \dot{m}_h = 0 & \text{when } \tau_h \in [T^{\min}, T^{\max}] & \quad (10) \\ \tau_h^+ &= \tau_r, \quad m_h^+ = v_h & \text{when } \tau_h \leq T^{\min} & \quad (11) \end{aligned}$$

- 4 *Digital Networks.* The information transfer between the physical and cyber components, or between subsystems within the cyber components, might occur over a digital communication network. The communication links bridging each of these components are not capable of continuously transmitting information, but rather, they can only transmit sampled (and quantized) information at discrete time instants. Combining the ideas in the models of the converters in the previous items, we propose a model of a digital network link that has a variable that triggers the transfer of information provided at its input, and that stores that information until new information arrives. We assume that the transmission of information occurs at instants  $\{t_i\}_{i=1}^{i^*}$ ,  $i^* \in \mathbb{N} \cup \{\infty\}$ , satisfying

$$T_N^{*\min} \leq t_{i+1} - t_i \leq T_N^{*\max} \quad \forall i \in \{1, 2, \dots, i^* - 1\}$$

where  $T_N^{*\min}$  and  $T_N^{*\max}$  are constants satisfying

$$T_N^{*\min}, T_N^{*\max} \in [0, \infty]$$

and

$$T_N^{*\min} \leq T_N^{*\max}$$

and  $i^*$  is the number of transmission events, which might be finite or infinite. The constant  $T_N^{*\min}$  determines the minimum possible time in between transmissions while the constant  $T_N^{*\max}$  defines the maximum amount of time elapsed between transmissions. In this way, a communication channel that allows transmission events at a high rate would have  $T_N^{*\min}$  small (zero for infinitely fast transmissions), while one with slow data rate would have  $T_N^{*\min}$  large. The constant  $T_N^{*\max}$  determines how often transmissions may take place. Note that the constants  $T_N^{*\min}$  and  $T_N^{*\max}$  can be generalized to functions so as to change according to other states or inputs.

At every  $t_i$ , the information at the input  $v_N$  of the communication link is used to update the internal variable  $m_N$ , which is accessible at the output end of the network and remains constant between

communication events. This internal variable acts as an information buffer, which can contain not only the latest piece of information transmitted but also previously transmitted information. A record of the number of communication events is logged in the internal variable  $j_N$ . At communication events, the value of  $j_N$  gets updated and remains constant in-between events. A mathematical model capturing the said mechanism is given by

$$\dot{\tau}_N = -1, \quad \dot{m}_N = 0, \quad \dot{j}_N = 0 \quad \text{when} \quad \tau_N \in [0, T_N^{*\max}] \quad (12)$$

$$\tau_N^+ \in [T_N^{*\min}, T_N^{*\max}], \quad m_N^+ = v_N, \quad j_N^+ = j_N + 1 \quad \text{when} \quad \tau_N \leq 0 \quad (13)$$

Note that the update law for  $\tau_N$  at jumps is given in terms of a difference inclusion, which implies that the new value of  $\tau_N$  is taken from the set  $[T_N^{*\min}, T_N^{*\max}]$ . The dimension of the states and the input would depend on the type of components that connect to and from it, and also the size of data transmitted and buffered. Similar to the models proposed for conversion, the model of the digital link (12)-(13) does not include delays nor quantization, but such effects can be incorporated if needed.

## 4.2 The Integrator System

In this section we discuss the internals of the *Integrator System* shown in Figure 6.

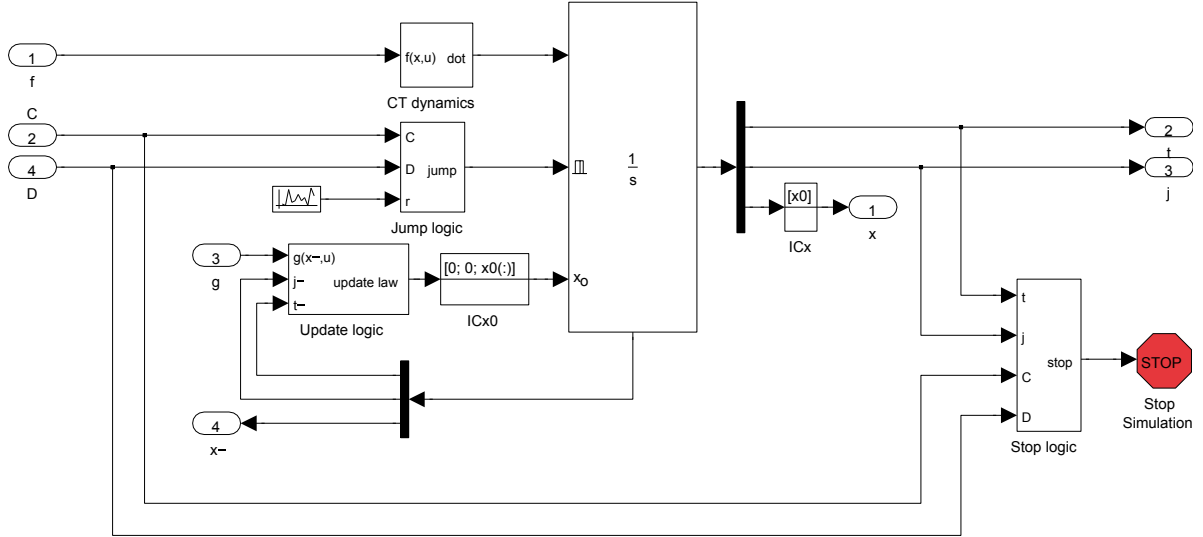


Figure 6: Integrator System

### 4.2.1 CT Dynamics

This block is shown in Figure 7. It defines the continuous-time (CT) dynamics by assembling the time derivative of the state  $[t \ j \ x^T]^T$ . States  $t$  and  $j$  are considered states of the system because they need to be updated throughout the simulation in order to keep track of the time and number of jumps. Without  $t$  and  $j$ , solutions could not be plotted accurately. This is given by

$$\dot{t} = 1, \quad \dot{j} = 0, \quad \dot{x} = f(x, u).$$

Note that input port 1 takes the value of  $f(x, u)$  through the output of the *Embedded MATLAB function block*  $f$  in Figure 4.

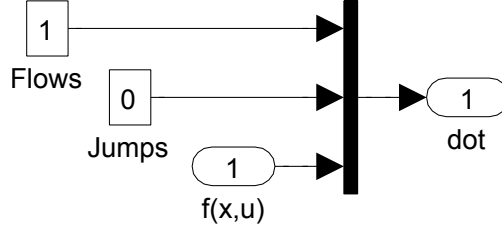


Figure 7: CT dynamics

### 4.2.2 Jump Logic

This block is shown in Figure 8. The inputs to the jump logic block are the output of the blocks  $C$  and  $D$  indicating whether the state is in those sets or not, and a random signal with uniform distribution in  $[0, 1]$ . Figure 8 shows the Simulink blocks used to implement the Jump Logic. The variable  $rule$  defines whether the simulator gives priority to jumps, priority to flows, or no priority. It is initialized in `initialization.m`.

The output of the Jump Logic is equal to one when:

- the output of the  $D$  block is equal to one and  $rule = 1$ ,
- the output of the  $C$  block is equal to zero, the output of the  $D$  block is equal to one, and  $rule = 2$ ,
- the output of the  $C$  block is equal to zero, the output of the  $D$  block is equal to one, and  $rule = 3$ ,
- or the output of the  $C$  block is equal to one, the output of the  $D$  block is equal to one,  $rule = 3$ , and the random signal  $r$  is larger or equal than 0.5.

Under these events, the output of this block, which is connected to the integrator external reset input, triggers a reset of the integrator, that is, a jump of  $\mathcal{H}$ . The reset or jump is activated since the configuration of the reset input is set to “level hold”, which executes resets when this external input is equal to one (if the next input remains set to one, multiple resets would be triggered). Otherwise, the output is equal to zero.

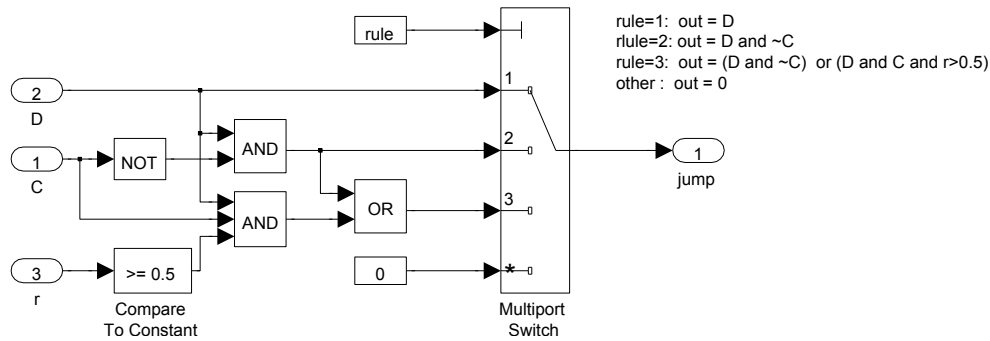


Figure 8: Jump Logic

### 4.2.3 Update Logic

This block is shown in Figure 9. The update logic uses the *state port* information of the integrator. This port reports the value of the state of the integrator,  $[t \ j \ x^T]^T$ , at the exact instant that the reset condition becomes true. Notice that  $x^-$  differs from  $x$  since at a jump,  $x^-$  indicates the value of the state that triggers the jump, but it is never assigned as the output of the integrator. In other words, “ $x \in D$ ” is checked using  $x^-$  and if true,  $x$  is reset to  $g(x^-, u)$ . Notice, however, that  $u$  is the same because at a jump,  $u$  indicates the

next evaluated value of the input, and it is assigned as the output of the integrator. The flow time  $t$  is kept constant at jumps and  $j$  is incremented by one. More precisely

$$t^+ = t^-, \quad j^+ = j^- + 1, \quad x^+ = g(x^-, u)$$

where  $[t^- \ j^- \ x^{-\top}]^\top$  is the state that triggers the jump.

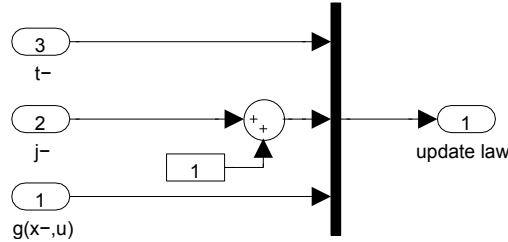


Figure 9: Update Logic

#### 4.2.4 Stop Logic

This block is shown in Figure 10. It stops the simulation under any of the following events:

- The flow time is larger than or equal to the maximum flow time specified by  $T$ .
- The jump time is larger than or equal to the maximum number of jumps specified by  $J$ .
- The state of the hybrid system  $x$  is neither in  $C$  nor in  $D$ .

Under any of these events, the output of the logic operator connected to the *Stop block* becomes one, stopping the simulation. Note that the inputs  $C$  and  $D$  are routed from the output of the blocks computing whether the state is in  $C$  or  $D$  and use the value of  $x^-$ .

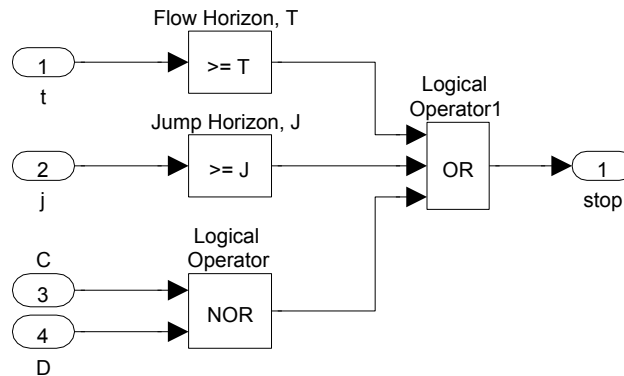


Figure 10: Stop Logic

### 4.3 Software Requirements

In order to run simulations of single hybrid systems using externally defined functions, MATLAB with Simulink is required.

In order to run simulations using the HyEQ Simulator with embedded MATLAB functions, MATLAB/Simulink and a supported ANSI, C, or C++ 32-bit compiler must be installed. We now briefly describe how to install necessary compilers for Windows and Mac/Linux. For more information on supported compilers, please visit <http://www.mathworks.com/support/compilers/R2012b/win64.html>.



### 4.3.1 Configuration of HyEQ Simulator with embedded functions for Windows

For 32-bit Windows, the LCC compiler is included with MATLAB. First, open MATLAB and then locate and choose a compiler for building MEX-files by typing

```
>> mex -setup
```

into the MATLAB command window. Then, follow the prompts as shown below.

```
>> mex -setup
```

```
Welcome to mex -setup. This utility will help you set up
a default compiler. For a list of supported compilers, see
http://www.mathworks.com/support/compilers/R2012a/win32.html
```

```
Please choose your compiler for building MEX-files:
```

```
Would you like mex to locate installed compilers [y]/n? y
```

```
Select a compiler:
```

```
[1] Lcc-win32 C 2.4.1
```

```
[0] None
```

```
Compiler: 1
```

```
Please verify your choices:
```

```
Compiler: Lcc-win32 C 2.4.1
```

```
Are these correct [y]/n? y
```

```
Done . . .
```

For 64-bit Windows, a C-compiler is not supplied with MATLAB. Before running the HyEQ Toolbox in MATLAB/Simulink, please follow the following steps:

1. If you don't have *Microsoft .NET Framework 4* on your computer, download and install it from <http://www.microsoft.com/en-us/download/details.aspx?id=17851>.
2. Then download and install *Microsoft Windows SDK* from <http://www.microsoft.com/en-us/download/details.aspx?id=8279>.
3. Then perform the steps outlined above for 32-bit Windows to setup and install the compiler.

As of October 10, 2013, when installing the toolbox in Windows 8, please follow the next steps.

1. If you don't have *Microsoft .NET Framework 4* on your computer, download and install it from <http://www.microsoft.com/en-us/download/details.aspx?id=8279>.
2. Then download and install *Microsoft Windows SDK*
  - If you don't have *Visual C++ 2010 SP1* installed on your computer:
    - Download and install *Microsoft Windows SDK 7.1* from <http://www.microsoft.com/en-us/download/details.aspx?displaylang=en&id=4422>
    - Apply the following patch from *Microsoft* onto the *SDK 7.1* installation: <http://www.microsoft.com/en-us/download/details.aspx?displaylang=en&id=4422>

- If you have Visual *Visual C++ 2010 SP1* or its redistributable packages installed on your computer:
  - Uninstall the *Visual C++ 2010* redistributable packages, both x64 and x86 versions. This can be done from *Control Panel / Uninstall Programs Menu*.
  - Download and install *Microsoft Windows SDK 7.1* from <http://www.microsoft.com/en-us/download/details.aspx?displaylang=en&id=4422>
  - Apply the following patch from *Microsoft* onto the *SDK 7.1* installation: <http://www.microsoft.com/en-us/download/details.aspx?displaylang=en&id=4422>
  - Reinstall the *Visual C++ 2010* redistributable packages:
    - x86 version: <http://www.microsoft.com/en-us/download/details.aspx?id=5555>
    - x64 version: <http://www.microsoft.com/en-us/download/details.aspx?id=14632>

3. Then perform the steps outlined above for 32-bit Windows to setup and install the compiler.

#### 4.3.2 Configuration of HyEQ Simulator with embedded functions for Mac/Linux

From a terminal window, check that the file `gcc` is in the folder `/usr/bin`. If it is not there, make a symbolic link. You might require to install the latest version of Xcode first. In order to generate a symbolic link for `gcc`, that MATLAB can find to compile the simulation files (see [http://www.mathworks.com/support/sysreq/previous\\_releases.html](http://www.mathworks.com/support/sysreq/previous_releases.html)), change folder to `/usr/bin` and then

```
sudo ln -s gcc gcc-4.2
```

Then, it should be possible to setup the `gcc` compiler in matlab as follows:

```
>> mex -setup
Options files control which compiler to use, the compiler and link command
options, and the runtime libraries to link against.
```

```
Using the 'mexsh -setup' command selects an options file that is
placed in ~/.matlab/R2013b and used by default for 'mexsh'. An options
file in the current working directory or specified on the command line
overrides the default options file in ~/.matlab/R2013b.
```

```
To override the default options file, use the 'mexsh -f' command
(see 'mexsh -help' for more information).
```

The options files available for MEX are:

The options files available for mexsh are:

```
1: /Applications/MATLAB_R2013b.app/bin/mexopts.sh :
   Template Options file for building MEX-files
```

```
0: Exit with no changes
```

```
Enter the number of the compiler (0-1): 1
```

```
Overwrite ~/.matlab/R2013b/mexopts.sh ([y]/n)? : Y
```

```
/Applications/MATLAB_R2013b.app/bin/mexopts.sh is being copied to
/SOME_FOLDER/mexopts.sh
```

At this point, it is possible to check if the *gcc* is properly setup by testing any of the Simulink examples with embedded functions (see Figure 3) (e.g., Examples 5.1, 5.2, 5.3, 5.4, 5.5 or 5.6).

If an error regarding “*gmake*” similar to “No supported compiler or SDK was found” and/or a warning “no such sysroot directory: ‘Developer/SDKs/MacOSX10.X.sdk’” is shown when compiling, please consider the following procedure.

A. For Matlab 2013b and previous:

The compiler can not find the appropriate path. It is necessary to change some lines in the file “*mexopts.sh*” (copied previously in the folder “SOME\_FOLDER”).

First, locate the Xcode-SDK in your hard drive. Open a terminal window and execute the following command

```
xcodebuild -version -sdk macosx10.9 Path
```

which returns the location of MacOSX10.9.sdk, denoted here as SDK\_FOLDER. Now, in the MATLAB command window locate the file “*mexopts.sh*” by typing

```
cd /SOME_FOLDER/
```

Then, open the file

```
edit mexopts.sh
```

and edit the lines

```
- SDKROOT='/Developer/SDKs/MacOSX10.X.sdk'  
to  
SDKROOT='SDK_FOLDER'  
and  
- MACOSX_DEPLOYMENT_TARGET='10.X'  
to  
MACOSX_DEPLOYMENT_TARGET='10.9'
```

Now, it is possible to test if the compiler works. If the following error appears “unknown type name ‘char16\_t’,” some flags must be changed to avoid this problem. It is required to add `-Dchar16_t=UINT16_T` and `-std=c++11` to the flags `CFLAGS` and `CXXFLAGS` respectively, e.g., change

```
- CFLAGS="-fno-common -no-cpp-precomp -arch $ARCHS -isysroot $SDKROOT  
-mmacosx-version-min=$MACOSX_DEPLOYMENT_TARGET"  
to  
CFLAGS="-fno-common -no-cpp-precomp -arch $ARCHS -isysroot $SDKROOT  
-mmacosx-version-min=$MACOSX_DEPLOYMENT_TARGET -Dchar16_t=UINT16_T"  
and  
- CXXFLAGS="-fno-common -no-cpp-precomp -fexceptions -arch $ARCHS -isysroot $SDKROOT  
-mmacosx-version-min=$MACOSX_DEPLOYMENT_TARGET"  
to  
CXXFLAGS="-fno-common -no-cpp-precomp -fexceptions -arch $ARCHS -isysroot $SDKROOT  
-mmacosx-version-min=$MACOSX_DEPLOYMENT_TARGET -std=c++11"
```

Finally, restart matlab and test any of the aforementioned Simulink examples. for more information visit <http://www.mathworks.com/matlabcentral/answers/121315-how-to-set-the-c-compiler-of-matlab2013a-in-osx-10-9>

B. For Matlab 2014b and newer.

First, locate the Xcode-SDK in your hard drive. It may be SDK 10.9, 10.10, 10.11, here we are going to denote it as **10.X**. Open a terminal window and execute the following command

```
xcodebuild -version -sdk macosx10.X Path
```

which returns the location of MacOSX10.X.sdk. We are interested in the last portion of the path, specifically after `/Applications/Xcode.app/Contents/Developer/`, denoted here as **SDK\_FOLDER**. Now, locate and edit the following files

```
/Applications/MATLAB_R201?? .app/bin/maci64/mexopts/clang_maci64.xml
```

```
/Applications/MATLAB_R201?? .app/bin/maci64/mexopts/clang++_maci64.xml
```

Inside those files there are the lines

```
<dirExists name="$$/Platforms/MacOSX.platform/Developer/SDKs/MacOSX10.10.sdk" />  
<cmdReturns name="find $$ -name MacOSX10.10.sdk" />
```

edit (or add below those lines) the lines

```
- <dirExists name="$$SDK\_FOLDER" />  
  and  
- <cmdReturns name="find $$ -name MacOSX10.X.sdk" />
```

Finally, restart matlab and test any of the aforementioned Simulink examples. for more information visit <https://bitbucket.org/d2d-development/d2d-software/issues/46/xcode-7-on-osx-with-matlab-r2015a-b>

## 4.4 Configuration of Integration Scheme

Before a simulation is started, it is important to determine the needed integrator scheme, zero-cross detection settings, precision, and other tolerances. Using the default settings does not always give the most efficient or most accurate simulations. One way to edit these settings is to open the Simulink Model, select **Simulation>Configuration Parameters>Solver**, and change the settings there. We have made this simple by defining variables for configuration parameters in the `initialization.m` file. The last few lines of the `initialization.m` file look like that given below.

```
1 %configuration of solver  
2 RelTol = 1e-8;  
3 MaxStep = .001;
```

In these lines, “RelTol = 1e-8” and “MaxStep = .001” define the relative tolerance and maximum step size of the ODE solver, respectively. These parameters greatly affect the speed and accuracy of solutions.

## 4.5 Initialization

When the block labeled *Double Click to Initialize* at the top of the Simulink Model is double-clicked, the simulation variables are initialized by calling the script `initialization.m`. The script `initialization.m` defines the initial conditions by defining the initial values of the state components, any necessary parameters, the maximum flow time specified by  $T$ , the maximum number of jumps specified by  $J$ , and tolerances used when simulating. These can be changed by editing the script file `initialization.m`. See below for sample code to initialize the bouncing ball example, Example 5.1.

```
1 % initialization for bouncing ball example  
2 clear all
```

```

3 % initial conditions
4 x0 = [1;0];
5 % simulation horizon
6 T = 10;
7 J = 20;
8 % rule for jumps
9 % rule = 1 -> priority for jumps
10 % rule = 2 -> priority for flows
11 % rule = 3 -> no priority, random selection when simultaneous conditions
12 rule = 1;
13 %configuration of solver
14 RelTol = 1e-8;

```

It is important to note that variables called in the *Embedded MATLAB function blocks* must be added as inputs and labeled as “parameters”. This can be done by opening the *Embedded MATLAB function block* selecting **Tools>Edit Data/Ports** and setting the scope to **Parameter**.

After the block labeled *Double Click to Initialize* is double-clicked and the variables initialized, the simulation is run by clicking the run button or selecting **Simulation>Start**.

## 4.6 Postprocessing and Plotting solutions

A similar procedure is used to define the plots of solutions after the simulation is run. The solutions can be plotted by double-clicking on the block at the top of the Simulink Model labeled *Double Click to Plot Solutions* which calls the script `postprocessing.m`. The script `postprocessing.m` may be edited to include the desired postprocessing and solution plots. See below for sample code to plot solutions to the bouncing ball example, Example 5.1. The functions used to generate the plots are described in Section 3.5.

```

1 %postprocessing for the bouncing ball example
2 % plot solution
3 figure(1)
4 clf
5 subplot(2,1,1),plotflows(t,j,x)
6 grid on
7 ylabel('x')
8 subplot(2,1,2),plotjumps(t,j,x)
9 grid on
10 ylabel('x')
11 % plot hybrid arc
12 figure(2)
13 plotHybridArc(t,j,x)
14 xlabel('j')
15 ylabel('t')
16 zlabel('x')
17 grid on
18 view(37.5,30)

```

## 5 Examples

The examples below illustrate the use of the Simulink implementation above.

**Example 5.1** (bouncing ball with input) For the simulation of the bouncing ball system with a constant input and regular data given by

$$f(x, u) := \begin{bmatrix} x_2 \\ -\gamma \end{bmatrix}, C := \{(x, u) \in \mathbb{R}^2 \times \mathbb{R} \mid x_1 \geq u\} \quad (14)$$

$$g(x, u) := \begin{bmatrix} u \\ -\lambda x_2 \end{bmatrix}, D := \{(x, u) \in \mathbb{R}^2 \times \mathbb{R} \mid x_1 \leq u, x_2 \leq 0\} \quad (15)$$

where  $\gamma > 0$  is the gravity constant,  $u$  is the input constant, and  $\lambda \in [0, 1)$  is the restitution coefficient. The MATLAB scripts in each of the function blocks of the implementation above are given as follows. An input was chosen to be  $u(t, j) = 0.2$  for all  $(t, j)$ . The constants for the bouncing ball system are  $\gamma = 9.81$  and  $\lambda = 0.8$ .

The following procedure is used to simulate this example using the model in the file `Example_1_2a.slx`:

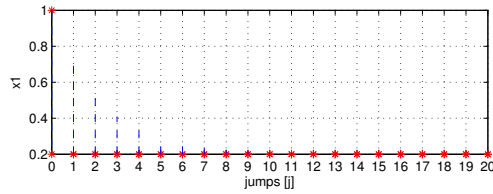
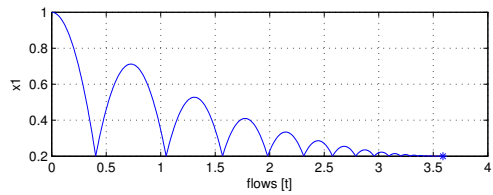
- `Example_1_2a.slx` is opened in MATLAB/Simulink.
- The *Embedded MATLAB function blocks*  $f$ ,  $C$ ,  $g$ ,  $D$  are edited by double-clicking on the block and editing the script. In each embedded function block, parameters must be added as inputs and defined as parameters by selecting `Tools>Edit Data/Ports`, and setting the scope to `Parameter`. For this example,  $\gamma$  and  $\lambda$  are defined in this way.
- The initialization script `initialization.m` is edited by opening the file and editing the script. The flow time and jump horizons,  $T$  and  $J$  are defined as well as the initial conditions for the state vector,  $x_0$ , and input vector,  $u_0$ , and a rule for jumps,  $rule$ .
- The postprocessing script `postprocessing.m` is edited by opening the file and editing the script. Flows and jumps may be plotted by calling the functions `plotflows` and `plotjumps`, respectively. The hybrid arc may be plotted by calling the function `plotHybridArc`.
- The simulation stop time and other simulation parameters are set to the values defined in `initialization.m` by selecting `Simulation>Configuration Parameters>Solver` and inputting  $T$ ,  $RelTol$ ,  $MaxStep$ , etc..
- The masked integrator system is double-clicked and the simulation horizons and initial conditions are set as desired.
- The block labeled *Double Click to Initialize* is double-clicked to initialize variables.
- The simulation is run by clicking the run button or selecting `Simulation>Start`.
- The block labeled *Double Click to Plot Solutions* is double-clicked to plot the desired solutions.

Flow map

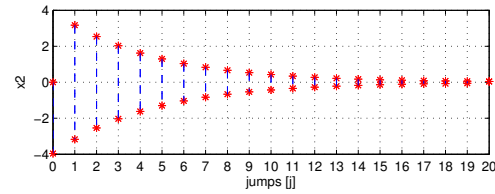
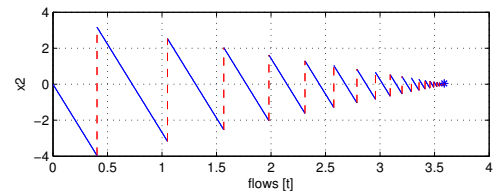
```

1 function xdot = f_ex1_2a(x)
2 %-----
3 % Matlab M-file Project: HyEQ Toolbox @ Hybrid Systems Laboratory (HSL),
4 % https://hybrid.soe.ucsc.edu/software
5 % http://hybridsimulator.wordpress.com/
6 % Filename: f_ex1_2a.m
7 %-----
8 % Project: Simulation of a hybrid system (bouncing ball)
9 % Description: Flow map
10 %-----
11 %-----
12 % See also HYESOLVER, PLOTARC, PLOTARC3, PLOTFLows, PLOTHARC,
13 % PLOTHARCCOLOR, PLOTHARCCOLOR3D, PLOTHYBRIDARC, PLOTJUMPS.
14 % Copyright @ Hybrid Systems Laboratory (HSL),
15 % Revision: 0.0.0.3 Date: 05/20/2015 3:42:00

```



(a) Height



(b) Velocity

Figure 11: Solution of Example 5.1

```

16
17 % state
18 x1 = x(1);
19 x2 = x(2);
20
21 global gamma
22
23 % differential equations
24 xdot = [x2 ; gamma];
25 end

Flow set
1 function [value] = C_ex1_2a(x)
2 %-----
3 % Matlab M-file Project: HyEQ Toolbox @ Hybrid Systems Laboratory (HSL),
4 % https://hybrid.soe.ucsc.edu/software
5 % http://hybridsimulator.wordpress.com/
6 % Filename: C_ex1_2a.m
7 %-----
8 % Description: Flow set
9 % Return 0 if outside of C, and 1 if inside C
10 %-----
11 %-----
12 % See also HYEQSOLVER, PLOTARC, PLOTARC3, PLOTFLAWS, PLOTHARC,
13 % PLOTHARCCOLOR, PLOTHARCCOLOR3D, PLOTHYBRIDARC, PLOTJUMPS.
14 % Copyright @ Hybrid Systems Laboratory (HSL),
15 % Revision: 0.0.0.3 Date: 05/20/2015 3:42:00
16
17
18 x1 = x(1);
19
20 if x1 >= 0
21     value = 1;
22 else

```

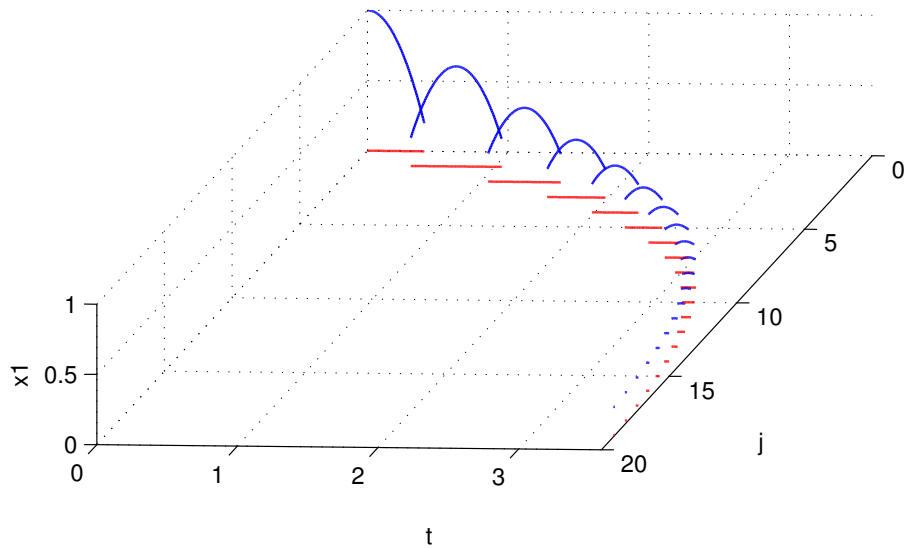


Figure 12: Hybrid arc corresponding to a solution of Example 5.1: height

```

23     value = 0;
24 end
25 end

    Jump map
1 function xplus = g_ex1_2a(x)
2 %-----
3 % Matlab M-file Project: HyEQ Toolbox @ Hybrid Systems Laboratory (HSL),
4 % https://hybrid.soe.ucsc.edu/software
5 % http://hybridsimulator.wordpress.com/
6 % Filename: g_ex1_2a.m
7 %-----
8 % Project: Simulation of a hybrid system (bouncing ball)
9 % Description: Jump map
10 %-----
11 %-----
12 % See also HYEQSOLVER, PLOTARC, PLOTARC3, PLOTFLAWS, PLOTHARC,
13 % PLOTHARCCOLOR, PLOTHARCCOLOR3D, PLOTHYBRIDARC, PLOTJUMPS.
14 % Copyright @ Hybrid Systems Laboratory (HSL),
15 % Revision: 0.0.0.3 Date: 05/20/2015 3:42:00
16
17 % state
18 x1 = x(1);
19 x2 = x(2);
20
21 global lambda

```



```

22
23 xplus = [-x1 ; -lambda*x2];
24 end

    Jump set
1 function inside = D_ex1_2a(x)
2 %-----
3 % Matlab M-file Project: HyEQ Toolbox @ Hybrid Systems Laboratory (HSL),
4 % https://hybrid.soe.ucsc.edu/software
5 % http://hybridsimulator.wordpress.com/
6 % Filename: D_ex1_2a.m
7 %-----
8 % Description: Jump set
9 % Return 0 if outside of D, and 1 if inside D
10 %-----
11 %-----
12 % See also HYEQSOLVER, PLOTARC, PLOTARC3, PLOTFLAWS, PLOTHARC,
13 % PLOTHARCCOLOR, PLOTHARCCOLOR3D, PLOTHYBRIDARC, PLOTJUMPS.
14 % Copyright @ Hybrid Systems Laboratory (HSL),
15 % Revision: 0.0.0.3 Date: 05/20/2015 3:42:00
16
17 x1 = x(1);
18 x2 = x(2);
19 if (x1 <= 0 && x2 <= 0)
20     inside = 1;
21 else
22     inside = 0;
23 end
24 end

```

A solution to the bouncing ball system from  $x(0,0) = [1,0]^T$  and with  $T = 10, J = 20, rule = 1$ , is depicted in Figure 11(a) (height) and Figure 11(b) (velocity). Both the projection onto  $t$  and  $j$  are shown. Figure 12 depicts the corresponding hybrid arc for the position state.

These simulations reflect the expected behavior of the bouncing ball model. Note the only difference between this example and the example of a bouncing ball without a constant input is that, in this example, the ball bounces on a platform at a height of the chosen input value 0.2 rather than the ground at a value of 0.

For MATLAB/Simulink files of this example, see Examples/Example.1.2a. □

**Example 5.2** (alternate way to simulate the bouncing ball)

Consider the bouncing ball system with a constant input and regular data as given in Example 5.1. This example shows that a MATLAB function block, such as the jump set  $D$ , can be replaced with operational blocks in Simulink. Figure 13 shows this implementation. The other functions and solutions are the same as in Example 5.1.

For MATLAB/Simulink files of this example, see Examples/Example.1.3 and Examples/Example.1.4. □

**Example 5.3** (vehicle following a track with boundaries) Consider a vehicle modeled by a Dubins vehicle model traveling along a given track with state vector  $x = [\xi_1, \xi_2, \xi_3]^T$  with dynamics given by  $\dot{\xi}_1 = u \cos \xi_3$ ,  $\dot{\xi}_2 = u \sin \xi_3$ , and  $\dot{\xi}_3 = -\xi_3 + r(q)$ . The input  $u$  is the tangential velocity of the vehicle,  $\xi_1$  and  $\xi_2$  describe the vehicle's position on the plane, and  $\xi_3$  is the vehicle's orientation angle. Also consider a switching controller attempting to keep the vehicle inside the boundaries of a track given by  $\{(\xi_1, \xi_2) : -1 \leq \xi_1 \leq 1\}$ . A state  $q \in \{1, 2\}$  is used to define the modes of operation of the controller. When  $q = 1$ , the vehicle is traveling to

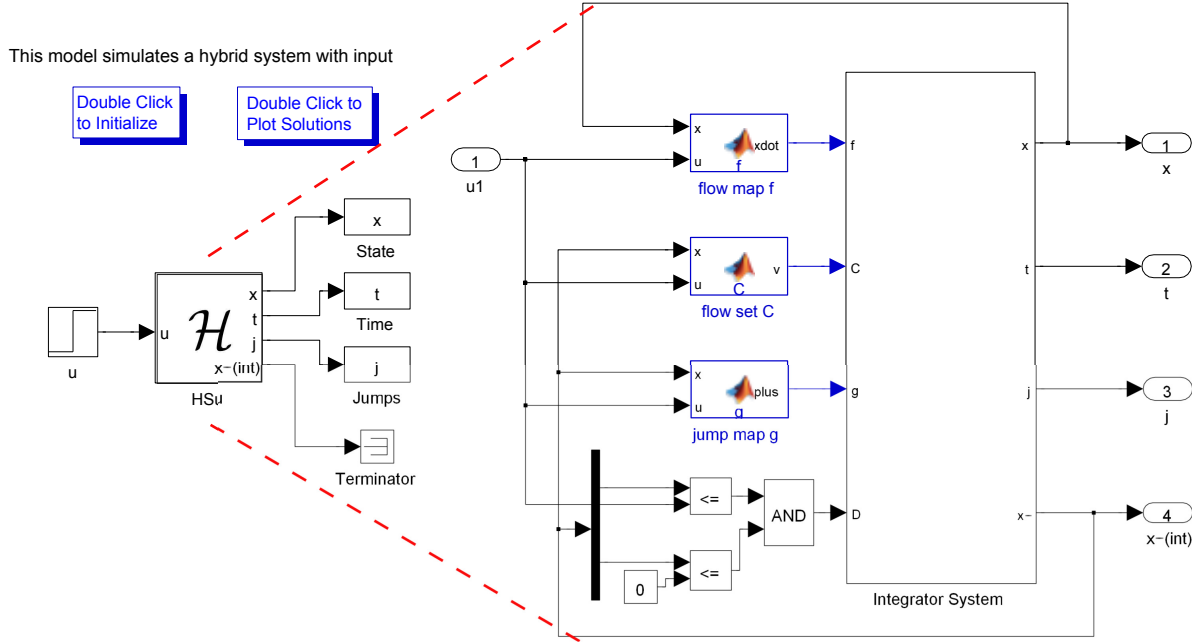


Figure 13: Simulink implementation of bouncing ball example with operator blocks

the left, and when  $q = 2$ , the vehicle is traveling to the right. A logic variable  $r$  is defined in order to steer the vehicle back inside the boundary. The state of the closed-loop system is given by  $x := [\xi^\top q]^\top$ . A model of such a closed-loop system is given by

$$f(x, u) := \begin{bmatrix} \begin{bmatrix} u \cos(\xi_3) \\ u \sin(\xi_3) \\ -\xi_3 + r(q) \end{bmatrix} \\ u \end{bmatrix}, \quad r(q) := \begin{cases} \frac{3\pi}{4} & \text{if } q = 1 \\ \frac{\pi}{4} & \text{if } q = 2 \end{cases} \quad (16)$$

$$C := \{(\xi, u) \in \mathbb{R}^3 \times \{1, 2\} \times \mathbb{R} \mid (\xi_1 \leq 1, q = 2) \text{ or } (\xi_1 \geq -1, q = 1)\}, \quad (17)$$

$$g(\xi, u) := \begin{cases} \begin{bmatrix} \xi \\ 2 \end{bmatrix} & \text{if } \xi_1 \leq -1, \quad q = 1 \\ \begin{bmatrix} \xi \\ 1 \end{bmatrix} & \text{if } \xi_1 \geq 1, \quad q = 2 \end{cases}, \quad (18)$$

$$D := \{(\xi, u) \in \mathbb{R}^3 \times \{1, 2\} \times \mathbb{R} \mid (\xi_1 \geq 1, q = 2) \text{ or } (\xi_1 \leq -1, q = 1)\} \quad (19)$$

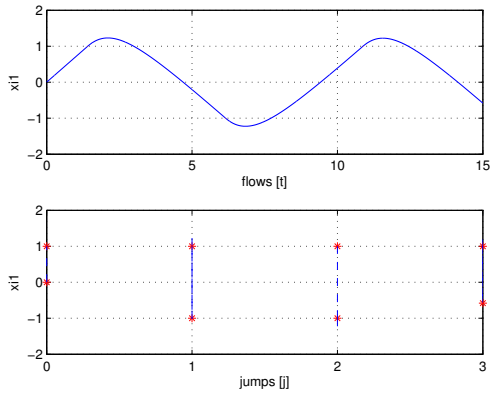
The MATLAB scripts in each of the function blocks of the implementation above are given as follows. The tangential velocity of the vehicle is chosen to be  $u = 1$ , the initial position on the plane is chosen to be  $(\xi_1, \xi_2) = (0, 0)$ , and the initial orientation angle is chosen to be  $\xi_3 = \frac{\pi}{4}$  radians.

Flow map

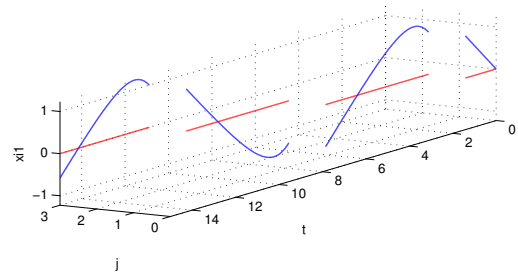
```

1 function xdot = f(x, u)
2
3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4 % Matlab Function Author: Ricardo Sanfelice
5 %
6 % Project: Simulation of a hybrid system (Vehicle Traveling on a Track)
7 %
8 % Name: f.m
9 %
10 % Description: Flow map

```



(a) Trajectory



(b) Hybrid arc

Figure 14: Solution of Example 5.3

```

11 %
12 % Version: 1.0
13 % Required files: -
14 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
15
16 xtemp = zeros(4,1);
17 xtemp = x;
18 x = xtemp;
19
20 % state
21 xi1 = x(1);      %x-position
22 xi2 = x(2);      %y-position
23 xi3 = x(3);      %orientation angle
24 q = x(4);
25
26 % q = 1 --> going left
27 % q = 2 --> going right
28
29 if q == 1
30     r = 3*pi/4;
31 elseif q == 2
32     r = pi/4;
33 else
34     r = 0;
35 end
36
37 % flow map: xidot=f(xi,u);
38 xildot = u*cos(xi3); %tangential velocity in x-direction
39 xi2dot = u*sin(xi3); %tangential velocity in y-direction
40 xi3dot = -xi3 + r;   %angular velocity
41 qdot = 0;
42
43 xdot = [xildot;xi2dot;xi3dot;qdot];

```

```

    Flow set
1  function v = C(x, u)
2
3  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4  % Matlab Function Author: Ricardo Sanfelice
5  %
6  % Project: Simulation of a hybrid system (Vehicle Traveling on a Track)
7  %
8  % Name: C.m
9  %
10 % Description: Flow set
11 %
12 % Version: 1.0
13 % Required files: -
14 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
15
16 xtemp = zeros(4,1);
17 xtemp = x;
18 x = xtemp;
19
20 % state
21 xi1 = x(1);      %x-position
22 xi2 = x(2);      %y-position
23 xi3 = x(3);      %orientation angle
24 q = x(4);
25
26 % q = 1 --> going left
27 % q = 2 --> going right
28
29 if ((xi1 < 1) && (q == 2)) || ((xi1 > -1) && (q == 1)) % flow condition
30     v = 1; % report flow
31 else
32     v = 0; % do not report flow
33 end

    Jump map
1  function xplus = g(x, u)
2
3  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4  % Matlab Function Author: Ricardo Sanfelice
5  %
6  % Project: Simulation of a hybrid system (Vehicle Traveling on a Track)
7  %
8  % Name: g.m
9  %
10 % Description: Jump map
11 %
12 % Version: 1.0
13 % Required files: -
14 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
15
16 xtemp = zeros(4,1);
17 xtemp = x;
18 x = xtemp;

```

```

19 % state
20 xi1 = x(1);      %x-position
21 xi2 = x(2);      %y-position
22 xi3 = x(3);      %orientation angle
23 q = x(4);
24
25 % q = 1 --> going left
26 % q = 2 --> going right
27 xilplus=xi1;
28 xi2plus=xi2;
29 xi3plus=xi3;
30 qplus=q;
31
32 % jump map
33 if ((xi1 >= 1) && (q == 2)) || ((xi1 <= -1) && (q == 1))
34     qplus = 3-q;
35 else
36     qplus = q;
37 end
38
39 xplus = [xilplus;xi2plus;xi3plus;qplus];

    Jump set
1  function v = D(x, u)
2
3  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4  % Matlab Function Author: Ricardo Sanfelice
5  %
6  % Project: Simulation of a hybrid system (Vehicle Traveling on a Track)
7  %
8  % Name: D.m
9  %
10 % Description: Jump set
11 %
12 % Version: 1.0
13 % Required files: -
14 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
15
16 xtemp = zeros(4,1);
17 xtemp = x;
18 x = xtemp;
19
20 % state
21 xi1 = x(1);      %x-position
22 xi2 = x(2);      %y-position
23 xi3 = x(3);      %orientation angle
24 q = x(4);
25
26 % q = 1 --> going left
27 % q = 2 --> going right
28
29 if ((xi1 >= 1) && (q == 2)) || ((xi1 <= -1) && (q == 1)) % jump condition
30     v = 1; % report jump
31 else

```

```

32     v = 0; % do not report jump
33 end

```

A solution to the system of a vehicle following a track in  $\{(\xi_1, \xi_2) : -1 \leq \xi_1 \leq 1\}$ , and with  $T = 15, J = 10, rule = 1$ , is depicted in Figure 14(a) (trajectory). Both the projection onto  $t$  and  $j$  are shown. Figure 14(b) depicts the corresponding hybrid arc.

For MATLAB/Simulink files of this example, see Examples/Example\_1.5. □

**Example 5.4** (interconnection of hybrid systems  $\mathcal{H}_1$  (bouncing ball) and  $\mathcal{H}_2$  (moving platform)) Consider a bouncing ball ( $\mathcal{H}_1$ ) bouncing on a platform ( $\mathcal{H}_2$ ) at some initial height and converging to the ground at zero height. This is an interconnection problem because the current states of each system affect the behavior of the other system. In this interconnection, the bouncing ball will contact the platform, bounce back up, and cause a jump in height of the platform so that it gets closer to the ground. After some time, both the ball and the platform will converge to the ground. In order to model this system, the output of the bouncing ball becomes the input of the moving platform, and vice versa. For the simulation of the described system with regular data where  $\mathcal{H}_1$  is given by

$$f_1(\xi, u_1, v_1) := \begin{bmatrix} \xi_2 \\ -\gamma - b\xi_2 + v_{11} \end{bmatrix}, C_1 := \{(\xi, u_1) \mid \xi_1 \geq u_1, u_1 \geq 0\} \quad (20)$$

$$g_1(\xi, u_1, v_1) := \begin{bmatrix} \xi_1 + \alpha_1 \xi_2^2 \\ e_1 |\xi_2| + v_{12} \end{bmatrix}, D_1 := \{(\xi, u_1) \mid \xi_1 = u_1, u_1 \geq 0\}, y_1 = h_1(\xi) := \xi_1 \quad (21)$$

where  $\gamma, b, \alpha_1 > 0, e_1 \in [0, 1)$ ,  $\xi = [\xi_1, \xi_2]^T$  is the state,  $y_1 \in \mathbb{R}$  is the output,  $u_1 \in \mathbb{R}$  and  $v_1 = [v_{11}, v_{12}]^T \in \mathbb{R}^2$  are the inputs, and the hybrid system  $\mathcal{H}_2$  is given by

$$f_2(\eta, u_2, v_2) := \begin{bmatrix} \eta_2 \\ -\eta_1 - 2\eta_2 + v_{21} \end{bmatrix}, C_2 := \{(\eta, u_2) \mid \eta_1 \leq u_2, \eta_1 \geq 0\} \quad (22)$$

$$g_2(\eta, u_2, v_2) := \begin{bmatrix} \eta_1 - \alpha_2 |\eta_2| \\ -e_2 |\eta_2| + v_{22} \end{bmatrix}, D_2 := \{(\eta, u_2) \mid \eta_1 = u_2, \eta_1 \geq 0\}, y_2 = h_2(\eta) := \eta_1 \quad (23)$$

where  $\alpha_2 > 0, e_2 \in [0, 1)$ ,  $\eta = [\eta_1, \eta_2]^T \in \mathbb{R}^2$  is the state,  $y_2 \in \mathbb{R}$  is the output, and  $u_2 \in \mathbb{R}$  and  $v_2 = [v_{21}, v_{22}]^T \in \mathbb{R}^2$  are the inputs.

Therefore, the interconnection may be defined by the input assignment

$$u_1 = y_2, \quad u_2 = y_1. \quad (24)$$

The signals  $v_1$  and  $v_2$  are included as external inputs in the model in order to simulate the effects of environmental perturbations, such as a wind gust, on the system.

The MATLAB scripts in each of the function blocks of the implementation above are given as follows. The constants for the interconnected system are  $\gamma = 0.8, b = 0.1$ , and  $\alpha_1, \alpha_2 = 0.1$ .

- For hybrid system  $\mathcal{H}_1$ :

Flow map

```

1 function xdot = f(x, u)
2
3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4 % Matlab Function Author: Ricardo Sanfelice
5 %
6 % Project: Simulation of a hybrid system (interconnection)
7 %
8 % Name: f.m

```

This model simulates the interconnection of two hybrid systems; a bouncing ball and a moving platform.

Double Click to Initialize

Double Click to Plot Solutions

More Info

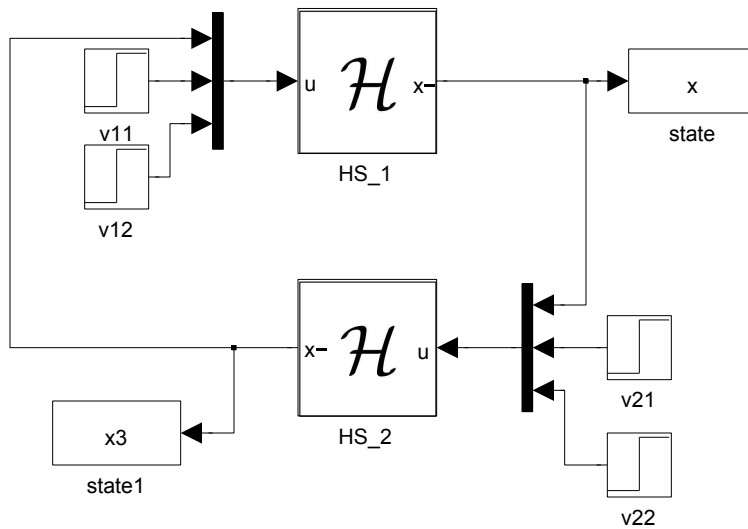


Figure 15: MATLAB/Simulink implementation of interconnected hybrid systems  $\mathcal{H}_1$  and  $\mathcal{H}_2$

```

9 %
10 % Description: Flow map
11 %
12 % Version: 1.0
13 % Required files: -
14 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
15
16 % state
17 xi1 = x(1);
18 xi2 = x(2);
19
20 %input
21 y2 = u(1);
22 v11 = u(2);
23 v12 = u(3);
24
25 % flow map
26 %xdot=f(x,u);
27 xildot = xi2;
28 xi2dot = -0.8-0.1*xi2+v11;
29
30 xdot = [xildot;xi2dot];

```

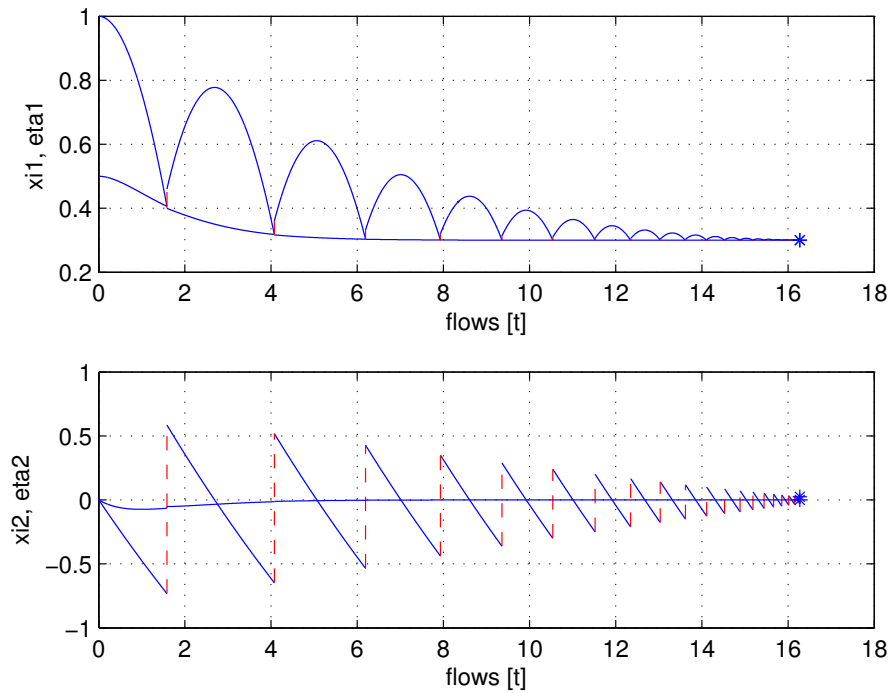


Figure 16: Solution of Example 5.4: height and velocity

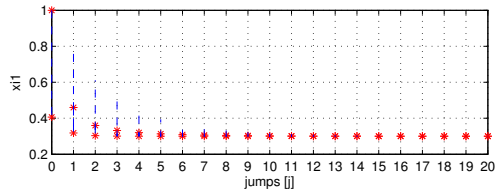
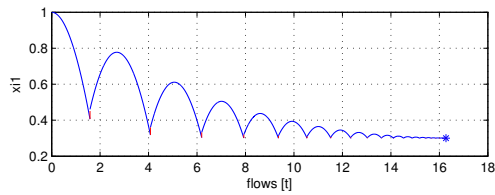
Flow set

```

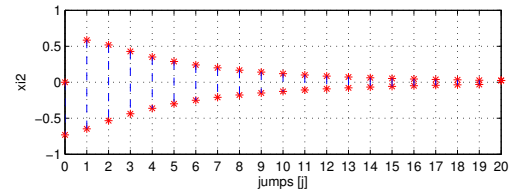
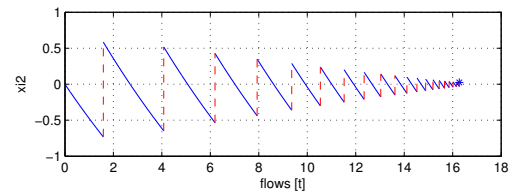
1 function v = C(x, u)
2
3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4 % Matlab Function Author: Ricardo Sanfelice
5 %
6 % Project: Simulation of a hybrid system (interconnection)
7 %
8 % Name: C.m
9 %
10 % Description: Flow set
11 %
12 % Version: 1.0
13 % Required files: -
14 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
15
16 % state
17 xi1 = x(1);
18 xi2 = x(2);
19
20 %input
21 y2 = u(1);
22 v11 = u(2);
23 v12 = u(3);
24
25 if (xi1 >= y2) % flow condition

```





(a) Height



(b) Velocity

Figure 17: Solution of Example 5.4 for system  $\mathcal{H}_1$

```

26     v = 1; % report flow
27 else
28     v = 0; % do not report flow
29 end

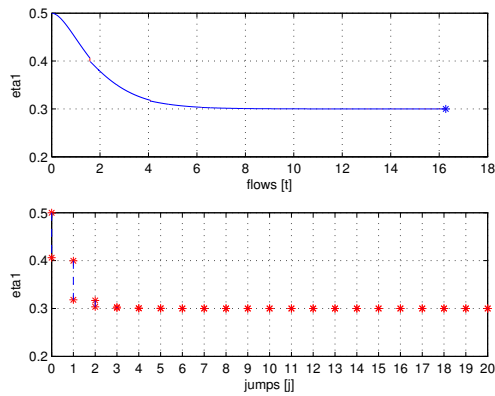
```

Jump map

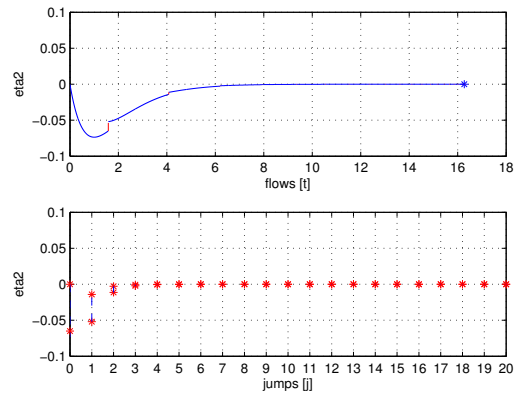
```

1 function xplus = g(x, u)
2
3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4 % Matlab Function Author: Ricardo Sanfelice
5 %
6 % Project: Simulation of a hybrid system (interconnection)
7 %
8 % Name: g.m
9 %
10 % Description: Jump map
11 %
12 % Version: 1.0
13 % Required files: -
14 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
15
16 % state
17 xi1 = x(1);
18 xi2 = x(2);
19
20 %input
21 y2 = u(1);
22 v11 = u(2);
23 v12 = u(3);
24
25 xi1plus=y2+0.1*xi2^2;
26 xi2plus=0.8*abs(xi2)+v12;
27

```



(a) Height



(b) Velocity

Figure 18: Solution of Example 5.4 for system  $\mathcal{H}_2$

```
28 xplus = [xi1plus;xi2plus];
```

Jump set

```
1 function v = D(x, u)
2
3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4 % Matlab Function Author: Ricardo Sanfelice
5 %
6 % Project: Simulation of a hybrid system (interconnection)
7 %
8 % Name: D.m
9 %
10 % Description: Jump set
11 %
12 % Version: 1.0
13 % Required files: -
14 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
15
16 % state
17 xi1 = x(1);
18 xi2 = x(2);
19
20 %input
21 y2 = u(1);
22 v11 = u(2);
23 v12 = u(3);
24
25
26 if (xi1 <= y2) % jump condition
27     v = 1; % report jump
28 else
29     v = 0; % do not report jump
30 end
```

- For hybrid system  $\mathcal{H}_2$ :

Flow map

```

1 function xdot = f(x, u)
2
3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4 % Matlab Function Author: Ricardo Sanfelice
5 %
6 % Project: Simulation of a hybrid system (interconnection)
7 %
8 % Name: f.m
9 %
10 % Description: Flow map
11 %
12 % Version: 1.0
13 % Required files: -
14 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
15
16
17 % state
18 eta1 = x(1);
19 eta2 = x(2);
20
21 %input
22 y1 = u(1);
23 v21 = u(2);
24 v22 = u(3);
25
26 % flow map
27 etaldot = eta2;
28 eta2dot = -eta1-2*eta2+v21;
29
30 xdot = [etaldot;eta2dot];

```

Flow set

```

1 function v = C(x, u)
2
3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4 % Matlab Function Author: Ricardo Sanfelice
5 %
6 % Project: Simulation of a hybrid system (interconnection)
7 %
8 % Name: C.m
9 %
10 % Description: Flow set
11 %
12 % Version: 1.0
13 % Required files: -
14 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
15
16 % state
17 eta1 = x(1);

```

```

18 eta2 = x(2);
19
20 %input
21 y1 = u(1);
22 v21 = u(2);
23 v22 = u(3);
24
25 if (eta1 <= y1) % flow condition
26     v = 1; % report flow
27 else
28     v = 0; % do not report flow
29 end

```

#### Jump map

```

1 function xplus = g(x, u)
2
3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4 % Matlab Function Author: Ricardo Sanfelice
5 %
6 % Project: Simulation of a hybrid system (interconnection)
7 %
8 % Name: g.m
9 %
10 % Description: Jump map
11 %
12 % Version: 1.0
13 % Required files: -
14 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
15
16 % state
17 eta1 = x(1);
18 eta2 = x(2);
19
20 %input
21 y1 = u(1);
22 v21 = u(2);
23 v22 = u(3);
24
25 % jump map
26 eta1plus = y1-0.1*abs(eta2);
27 eta2plus = -0.8*abs(eta2)+v22;
28
29 xplus = [eta1plus;eta2plus];

```

#### Jump set

```

1 function v = D(x, u)
2
3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4 % Matlab Function Author: Ricardo Sanfelice
5 %
6 % Project: Simulation of a hybrid system
7 %
8 % Name: D.m

```

```

9 %
10 % Description: Jump set
11 %
12 % Version: 1.0
13 % Required files: -
14 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
15
16 % state
17 eta1 = x(1);
18 eta2 = x(2);
19
20 %input
21 y1 = u(1);
22 v21 = u(2);
23 v22 = u(3);
24
25
26 if (eta1 >= y1) % jump condition
27     v = 1; % report jump
28 else
29     v = 0; % do not report jump
30 end

```

A solution to the interconnection of hybrid systems  $\mathcal{H}_1$  and  $\mathcal{H}_2$  with  $T = 18, J = 20, rule = 1$ , is depicted in Figure 16. Both the projection onto  $t$  and  $j$  are shown. A solution to the hybrid system  $\mathcal{H}_1$  is depicted in Figure 17(a) (height) and Figure 17(b) (velocity). A solution to the hybrid system  $\mathcal{H}_2$  is depicted in Figure 18(a) (height) and Figure 18(b) (velocity).

These simulations reflect the expected behavior of the interconnected hybrid systems.

For MATLAB/Simulink files of this example, see Examples/Example\_1.6.

□

**Example 5.5** (biological example: synchronization of two fireflies) Consider a biological example of the synchronization of two fireflies flashing. The fireflies can be modeled mathematically as periodic oscillators which tend to synchronize their flashing until they are flashing in phase with each other. A state value of  $\tau_i = 1$  corresponds to a flash, and after each flash, the firefly automatically resets its internal timer (periodic cycle) to  $\tau_i = 0$ . The synchronization of the fireflies can be modeled as an interconnection of two hybrid systems because every time one firefly flashes, the other firefly notices and jumps ahead in its internal timer  $\tau$  by  $(1 + \varepsilon)\tau$ , where  $\varepsilon$  is a biologically determined coefficient. This happens until eventually both fireflies synchronize their internal timers and are flashing simultaneously. Each firefly can be modeled as a hybrid system given by

$$f_i(\tau_i, u_i) := 1, \tag{25}$$

$$C_i := \{(\tau_i, u_i) \in \mathbb{R}^2 \mid 0 \leq \tau_i \leq 1\} \cap \{(\tau_i, u_i) \in \mathbb{R}^2 \mid 0 \leq u_i \leq 1\} \tag{26}$$

$$g_i(\tau_i, u_i) := \begin{cases} (1 + \varepsilon)\tau_i & (1 + \varepsilon)\tau_i < 1 \\ 0 & (1 + \varepsilon)\tau_i \geq 1 \end{cases} \tag{27}$$

$$D_i := \{(\tau_i, u_i) \in \mathbb{R}^2 \mid \tau_i = 1\} \cup \{(\tau_i, u_i) \in \mathbb{R}^2 \mid u_i = 1\}. \tag{28}$$

The interconnection diagram for this example is simpler than in the previous example because now no external inputs are being considered. The only event that affects the flashing of a firefly is the flashing of the other firefly. The interconnection diagram can be seen in Figure 19.

For hybrid system  $\mathcal{H}_i, i = 1, 2$ :

Flow map

```

1 function taudot = f(tau, u)
2

```

This model simulates interconnected hybrid systems.

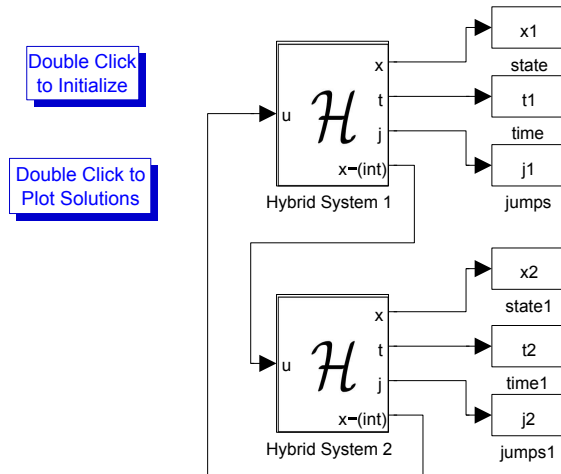


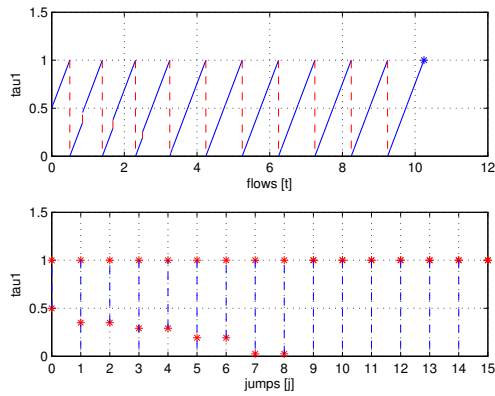
Figure 19: Interconnection Diagram for Example 5.5

```

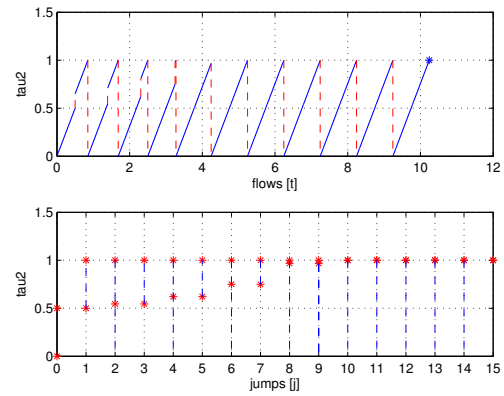
3  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4  % Matlab Function Author: Ricardo Sanfelice
5  %
6  % Project: Simulation of a hybrid system (interconnection)
7  %
8  % Name: f.m
9  %
10 % Description: Flow map
11 %
12 % Version: 1.0
13 % Required files: -
14 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
15
16 % flow map
17 taudot = 1;

    Flow set
1  function v = C(tau, u)
2
3  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4  % Matlab Function Author: Ricardo Sanfelice
5  %
6  % Project: Simulation of a hybrid system (interconnection)
7  %
8  % Name: C.m
9  %
10 % Description: Flow set
11 %
12 % Version: 1.0
13 % Required files: -
14 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
15
16 %flow set

```



(a) Solution for system  $\mathcal{H}_1$



(b) Solution for system  $\mathcal{H}_2$

Figure 20: Solution of Example 5.5

```

17 if ((tau > 0) && (tau < 1)) || ((u > 0) && (u <= 1)) % flow condition
18     v = 1; % report flow
19 else
20     v = 0; % do not report flow
21 end

    Jump map
1 function tauplus = g(tau, u, e)
2
3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4 % Matlab Function Author: Ricardo Sanfelice
5 %
6 % Project: Simulation of a hybrid system (interconnection)
7 %
8 % Name: g.m
9 %
10 % Description: Jump map
11 %
12 % Version: 1.0
13 % Required files: -
14 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
15
16 % jump map
17 if (1+e)*tau < 1
18     tauplus = (1+e)*tau;
19 elseif (1+e)*tau >= 1
20     tauplus = 0;
21 else
22     tauplus = tau;
23 end

    Jump set
1 function v = D(tau, u)
2

```

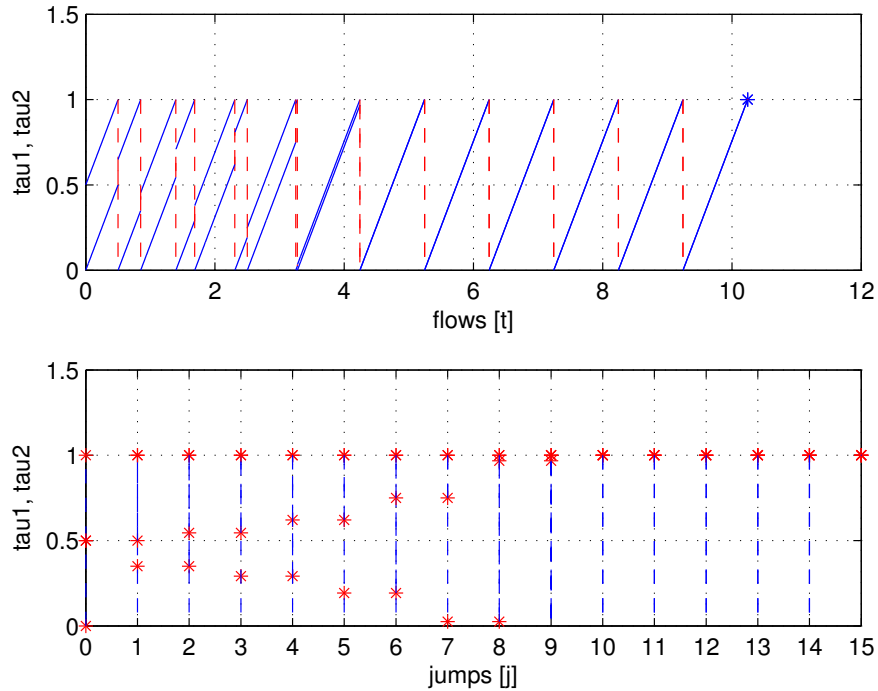


Figure 21: Solution of Example 5.5 for interconnection of  $\mathcal{H}_1$  and  $\mathcal{H}_2$

```

3  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4  % Matlab Function Author: Ricardo Sanfelice
5  %
6  % Project: Simulation of a hybrid system (interconnection)
7  %
8  % Name: D.m
9  %
10 % Description: Jump set
11 %
12 % Version: 1.0
13 % Required files: -
14 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
15
16 if (u >= 1) || (tau >= 1) % jump condition
17     v = 1; % report jump
18 else
19     v = 0; % do not report jump
20 end

```

A solution to the interconnection of hybrid systems  $\mathcal{H}_1$  and  $\mathcal{H}_2$  with  $T = 15, J = 15, rule = 1, \varepsilon = 0.3$  is depicted in Figure 21. Both the projection onto  $t$  and  $j$  are shown. A solution to the hybrid system  $\mathcal{H}_1$  is depicted in Figure 20(a). A solution to the hybrid system  $\mathcal{H}_2$  is depicted in Figure 20(b).

These simulations reflect the expected behavior of the interconnected hybrid systems. The fireflies initially flash out of phase with one another and then synchronize to flash in the same phase.

For MATLAB/Simulink files of this example, see Examples/Example\_1.7.

□



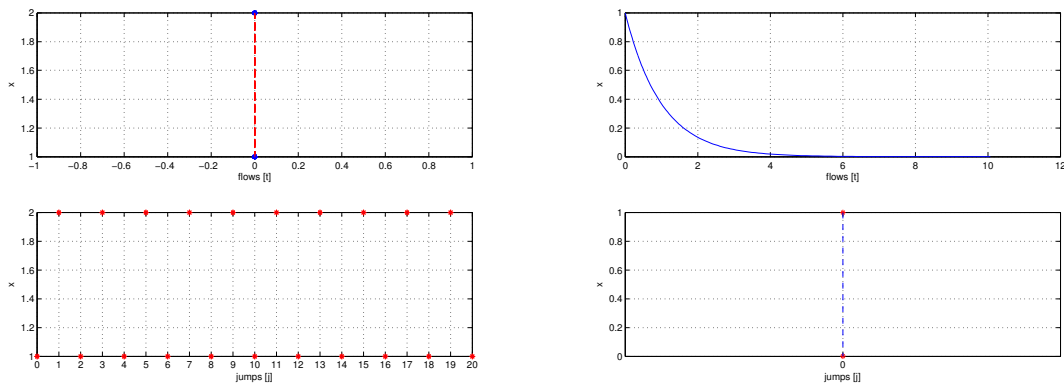
**Example 5.6** (a simple mathematical example to show different type of simulation results) Consider the hybrid system with data

$$f(x) := -x, \quad C := [0, 1], \quad g(x) := 1 + \text{mod}(x, 2), \quad D := \{1\} \cup \{2\}.$$

Note that solutions from  $\xi = 1$  and  $\xi = 2$  are nonunique. The following simulations show the use of the variable *rule* in the *Jump Logic block*.

**Jumps enforced:** A solution from  $x_0 = 1$  with  $T = 10, J = 20, rule = 1$  is depicted in Figure 22(a). The solution jumps from 1 to 2, and from 2 to 1 repetitively.

**Flows enforced:** A solution from  $x_0 = 1$  with  $T = 10, J = 20, rule = 2$  is depicted in Figure 22(b). The solution flows for all time and converges exponentially to zero.



(a) Forced jumps logic.

(b) Forced flows logic.

Figure 22: Solution of Example 5.6

**Random rule:** A solution from  $x_0 = 1$  with  $T = 10, J = 20, rule = 3$  is depicted in Figure 23(a). The solution jumps to 2, then jumps to 1 and flows for the rest of the time converging to zero exponentially. Enlarging  $D$  to

$$D := [1/50, 1] \cup \{2\}$$

causes the overlap between  $C$  and  $D$  to be “thicker”. The simulation result is depicted in Figure 23(b) with the same parameters used in the simulation in Figure 23(a). The plot suggests that the solution jumps several times until  $x < 1/50$  from where it flows to zero. However, Figure 23(c), a zoomed version of Figure 23(b), shows that initially the solution flows and that at  $(t, j) = (0.2e - 3, 0)$  it jumps. After the jump, it continues flowing, then it jumps a few times, then it flows, etc. The combination of flowing and jumping occurs while the solution is in the intersection of  $C$  and  $D$ , where the selection of whether flowing or jumping is done randomly due to using  $rule = 3$ .

This simulation also reveals that this implementation does not precisely generate hybrid arcs. The maximum step size was set to  $0.1e - 3$ . The solution flows during the first two steps of the integration of the flows with maximum step size. The value at  $t = 0.1e - 3$  is very close to 1. At  $t = 0.2e - 3$ , instead of assuming a value given by the flow map, the value of the solution is about 0.5, which is the result of the jump occurring at  $(0.2e - 3, 0)$ . This is the value stored in  $x$  at such time by the integrator. Note that the value of  $x'$  at  $(0.2e - 3, 0)$  is the one given by the flow map that triggers the jump, and if available for recording, it should be stored in  $(0.2e - 3, 0)$ . This is a limitation of the current implementation.

The following simulations show the *Stop Logic block* stopping the simulation at different events.

**Solution outside  $C \cup D$ :** Taking  $D = \{1\}$ , a simulation starting from  $x_0 = 1$  with  $T = 10, J = 20, rule = 1$  stops since the solution leaves  $C \cup D$ . Figure 24(a) shows this.

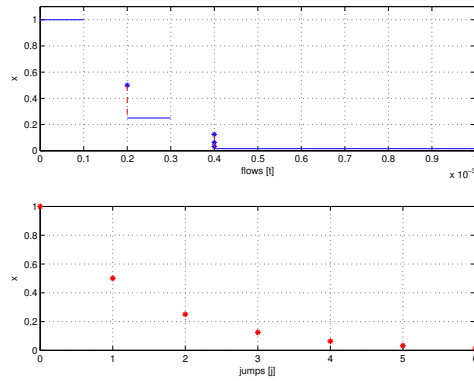
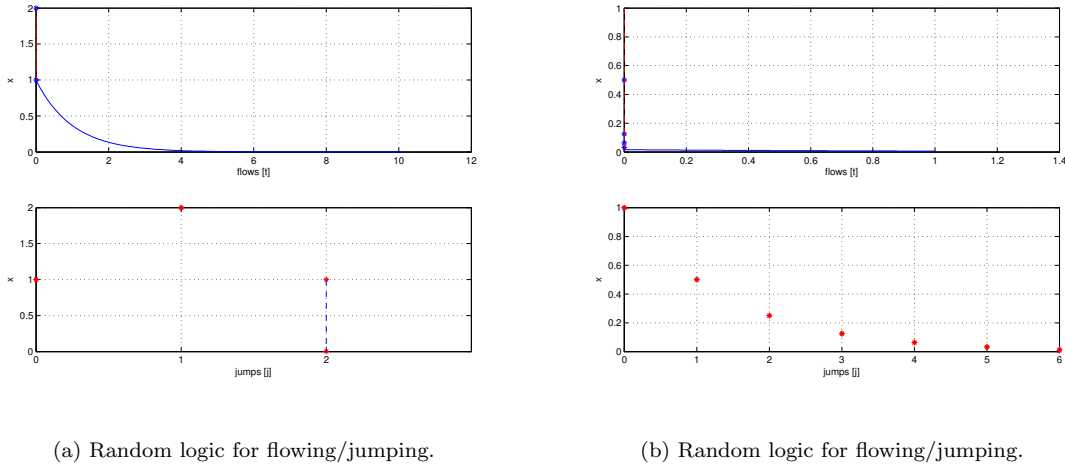


Figure 23: Solution of Example 5.6

**Solution reaches the boundary of  $C$  from where jumps are not possible:** Replacing the flow set by  $[1/2, 1]$  a solution starting from  $x_0 = 1$  with  $T = 10, J = 20$  and  $rule = 2$  flows for all time until it reaches the boundary of  $C$  where jumps are not possible. Figure 24(b) shows this.

Note that in this implementation, the Stop Logic is such that when the state of the hybrid system is not in  $(C \cup D)$ , then the simulation is stopped. In particular, if this condition becomes true while flowing, then the last value of the computed solution will not belong to  $C$ . It could be desired to be able to recompute the solution so that its last point belongs to the corresponding set. From that point, it should be the case that solutions cannot be continued.

For MATLAB/Simulink files of this example, see Examples/Example.1.8.

□

**Example 5.7** (Linear time-invariant plant)

Following the model of a physical component in (4), a linear time-invariant model of the physical component is defined by

$$f(x, u) = f_P(x, u) = A_P x + B_P u, \quad h(x, u) = h(x, u) = M_P x + N_P u$$

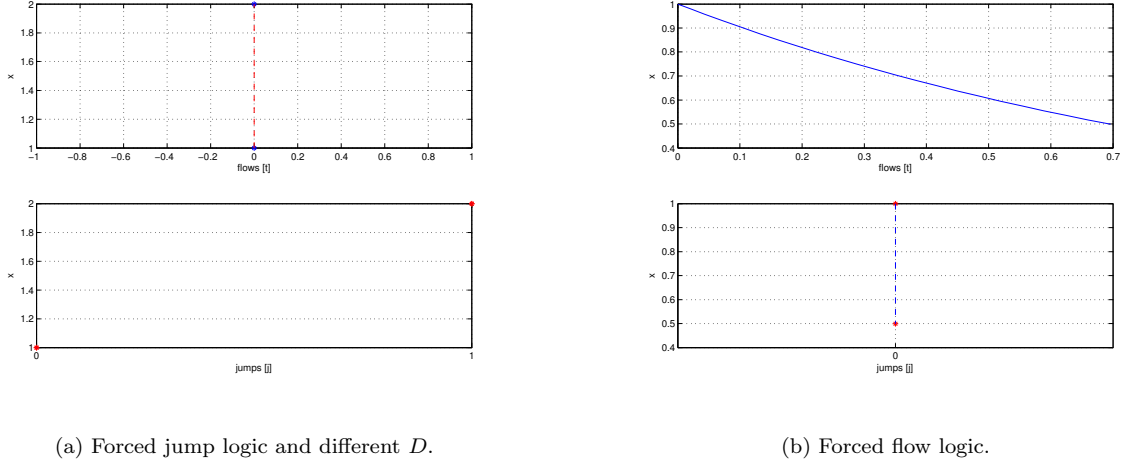


Figure 24: Solution of Example 5.6 with premature stopping.

where  $A_P$ ,  $B_P$ ,  $M_P$ , and  $N_P$  are matrices of appropriate dimensions. State and input constraints can directly be embedded into the set  $C_P$ . For example, the constraint that  $x$  has all of its components nonnegative and that  $u$  has its components with norm less or equal than one is captured by

$$C_P = \{(x, u) \in \mathbb{R}^{n_P} \times \mathbb{R}^{m_P} \mid x_i \geq 0 \forall i \in \{1, 2, \dots, n_P\}\} \\ \cap \{(x, u) \in \mathbb{R}^{n_P} \times \mathbb{R}^{m_P} \mid |u_i| \leq 1 \forall i \in \{1, 2, \dots, m_P\}\}$$

For example, the evolution of the temperature of a room with a heater can be modeled by a linear-time invariant system with state  $x$  denoting the temperature of the room and with input  $u = (u_1, u_2)$ , where  $u_1$  denotes whether the heater is turned on ( $u_1 = 1$ ) or turned off ( $u_1 = 0$ ) while  $u_2$  denotes the temperature outside the room. The evolution of the temperature is given by

$$\dot{x} = -x + [z_\Delta \quad 1] \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad \text{when } (x, u) \in C_P = \{(x, u) \in \mathbb{R} \times \mathbb{R}^2 \mid u_1 \in \{0, 1\}\} \quad (29)$$

where  $z_\Delta$  is a constant representing the heater capacity. □

**Example 5.8** (Implementation of a FSM)

Transitions of the state of a FSM are triggered by changes of its input  $v$ . The system can be modeled as a cascade of two systems, in which an external signal drives the FSM. The FSM is modeled as the difference equation in (7). Assuming that the input feeds the input  $u$  of the FSM, the state of the FSM is updated according to the transition function evaluated at the current input, that is,  $q$  is updated according to

$$q^+ = \delta(q, u)$$

The above model can be summarized as follows:

$$f(q, u) := \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad (30)$$

$$C := \{(q, u) \in \{1, 2\} \times \{1, 2\} \times \{1, 2\} \times \{1, 2\} \mid \delta(q, u) = q\}, \quad (31)$$

$$g(q, u) := \delta(q, u) = \begin{bmatrix} 3 - u_1 \\ 3 - u_2 \end{bmatrix}, \quad (32)$$

$$D := \{(q, u) \in \{1, 2\} \times \{1, 2\} \times \{1, 2\} \times \{1, 2\} \mid \delta(q, u) \in \{1, 2\} \times \{1, 2\} \setminus q\}, \quad (33)$$

$$y := h(q) = q \quad (34)$$

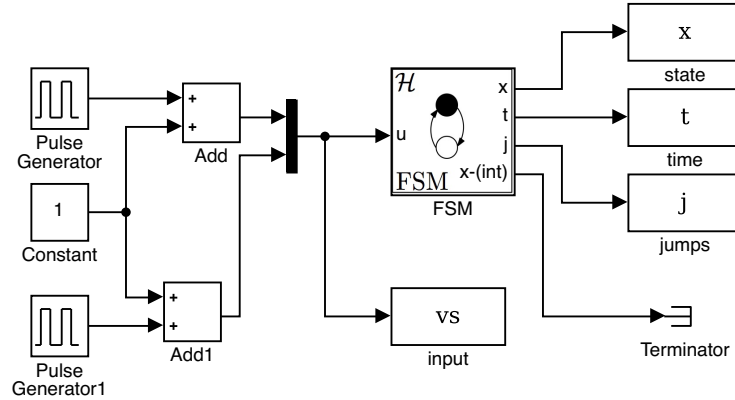


Figure 25: Simulation of a Finite State Machine (FSM) in Example 5.8.

where the input and the state are given by  $u = (u_1, u_2) \in \{1, 2\} \times \{1, 2\}$ , and  $q = (q_1, q_2) \in \{1, 2\} \times \{1, 2\}$ , respectively.

For the hybrid system FSM in Figure 25 we have the following Matlab embedded functions that describe the sets  $C$  and  $D$  and functions  $f$  and  $g$ . Figure 26 depicts the corresponding inputs and state of the FSM.

Flow map

```

1 function xdot = f(x, u)
2 %-----
3 % Matlab M-file Project: HyEQ Toolbox @ Hybrid Systems Laboratory (HSL),
4 % https://hybrid.soe.ucsc.edu/software
5 % http://hybridsimulator.wordpress.com/
6 %-----
7 % Project: Simulation of a hybrid system Finite state machine (FSM)
8 % Description: Flow map
9 %-----
10 %
11 % See also HYEQSOLVER, PLOTARC, PLOTARC3, PLOTFLows, PLOTHARC,
12 % PLOTHARCCOLOR, PLOTHARCCOLOR3D, PLOTHYBRIDARC, PLOTJUMPS.
13 % Copyright @ Hybrid Systems Laboratory (HSL),
14 % Revision: 0.0.0.3 Date: 05/20/2015 3:42:00
15 %-----
16 % flow map: xdot=f(x,u);
17
18 n = length(x); % size of the q state
19 qdot = zeros(n,1); % dynamics of the q state
20 xdot = [qdot];

```

Flow set

```

1 function v = C(x, u)
2 %-----
3 % Matlab M-file Project: HyEQ Toolbox @ Hybrid Systems Laboratory (HSL),
4 % https://hybrid.soe.ucsc.edu/software
5 % http://hybridsimulator.wordpress.com/
6 %-----
7 % Project: Simulation of a hybrid system Finite state machine (FSM)
8 % Description: Flow set
9 %-----

```

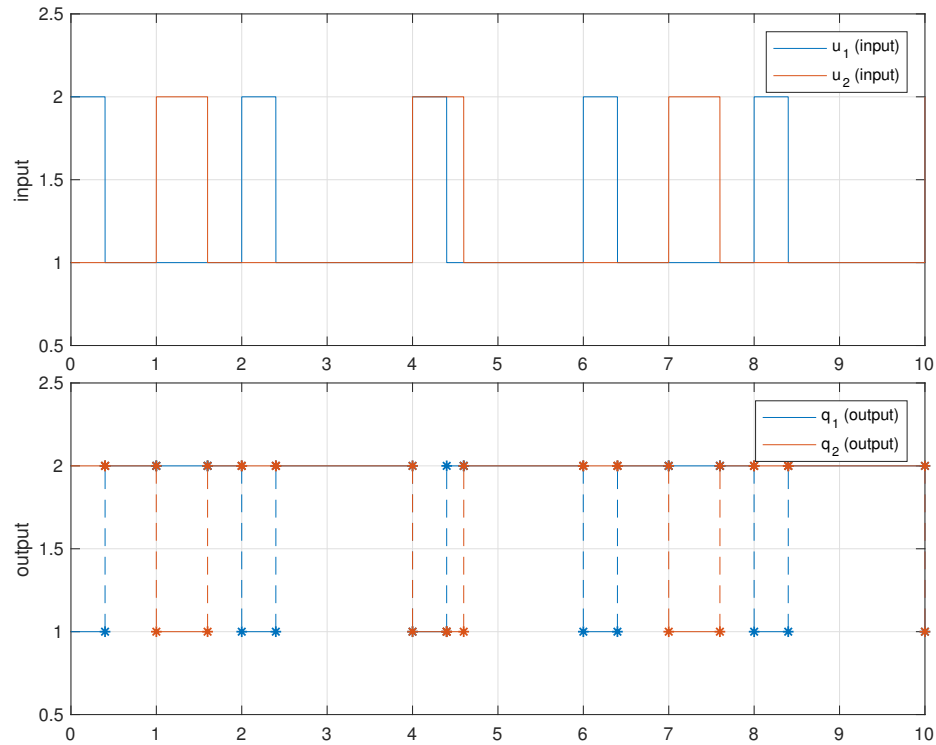


Figure 26: Finite state machine states and inputs.

```

10  %-----
11  %   See also HYEQSOLVER, PLOTARC, PLOTARC3, PLOTFLAWS, PLOTHARC,
12  %   PLOTHARCCOLOR, PLOTHARCCOLOR3D, PLOTHYBRIDARC, PLOTJUMPS.
13  %   Copyright © Hybrid Systems Laboratory (HSL),
14  %   Revision: 0.0.0.3 Date: 05/20/2015 3:42:00
15  %
16  % Check on flow conditions
17  % E.g.,
18  % if (x(1) >= u(1)) % flow condition
19  %     v = 1; % report flow
20  % else
21  %     v = 0; % do not report flow
22  % end
23
24  xtemp = zeros(2,1);
25  xtemp = x;
26  x = xtemp;
27
28  % n = length(x); % size of the q state
29  qplus = zeros(2,1); % dynamics of the q state
30  % e.g.,
31  qplus(1) = 3-u(1);
32  qplus(2) = 3-u(2);
33  if x(1)==qplus(1) && x(2)==qplus(2)
34     v = 1; % report flow
35  else

```

```

36     v = 0; % do not report flow
37 end
38

```

#### Jump map

```

1 function xplus = g(x, u)
2 %-----
3 % Matlab M-file Project: HyEQ Toolbox @ Hybrid Systems Laboratory (HSL),
4 % https://hybrid.soe.ucsc.edu/software
5 % http://hybridsimulator.wordpress.com/
6 %-----
7 % Project: Simulation of a hybrid system Finite state machine (FSM)
8 % Description: Jump map
9 %-----
10 %-----
11 % See also HYEQSOLVER, PLOTARC, PLOTARC3, PLOTFLows, PLOTHARC,
12 % PLOTHARCCOLOR, PLOTHARCCOLOR3D, PLOTHYBRIDARC, PLOTJUMPS.
13 % Copyright @ Hybrid Systems Laboratory (HSL),
14 % Revision: 0.0.0.3 Date: 05/20/2015 3:42:00
15 %-----
16 % jump map: xplus = g(x,u);
17
18 % n = length(x); % size of the q state
19 qplus = zeros(2,1); % dynamics of the q state
20 % e.g.,
21 qplus(1) = 3-u(1);
22 qplus(2) = 3-u(2);
23
24 xplus = [qplus];
25

```

#### Jump set

```

1 function v = D(x, u)
2 %-----
3 % Matlab M-file Project: HyEQ Toolbox @ Hybrid Systems Laboratory (HSL),
4 % https://hybrid.soe.ucsc.edu/software
5 % http://hybridsimulator.wordpress.com/
6 %-----
7 % Project: Simulation of a hybrid system Finite state machine (FSM)
8 % Description: Jump set
9 %-----
10 %-----
11 % See also HYEQSOLVER, PLOTARC, PLOTARC3, PLOTFLows, PLOTHARC,
12 % PLOTHARCCOLOR, PLOTHARCCOLOR3D, PLOTHYBRIDARC, PLOTJUMPS.
13 % Copyright @ Hybrid Systems Laboratory (HSL),
14 % Revision: 0.0.0.3 Date: 05/20/2015 3:42:00
15 %
16 % Check on jump conditions
17 % % E.g.,
18 % if (x(1) <= u(1)) && (x(2) <= 0) % jump condition
19 %     v = 1; % report jump
20 % else
21 %     v = 0; % do not report jump
22 % end

```

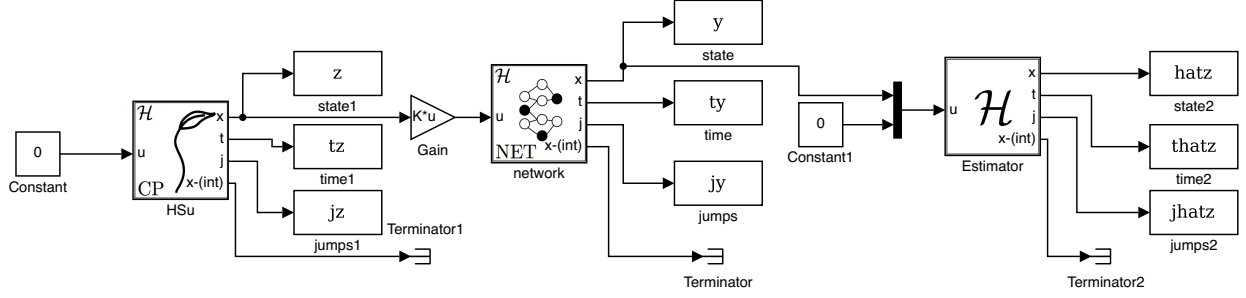


Figure 27: Estimation over a network scheme in Example 5.9.

```

23 xtemp = zeros(2,1);
24 xtemp = x;
25 x = xtemp;
26
27 % n = length(x); % size of the q state
28 qplus = zeros(2,1); % dynamics of the q state
29 % e.g.,
30 qplus(1) = 3-u(1);
31 qplus(2) = 3-u(2);
32 if x(1)==qplus(1) && x(2)==qplus(2)
33     v = 0; % do not report jump
34 else
35     v = 1; % report jump
36 end

```

□

**Example 5.9** (Estimation Over a Network) Consider a physical process given in terms of the state-space model

$$\dot{x} = Ax, \quad y = Mx, \quad x \in \mathbb{R}^{n_P}, y \in \mathbb{R}^{r_P} \quad (35)$$

where  $x$  is the state and  $y$  is the measured output. The output is digitally transmitted through a network. At the other end of the network, a computer receives the information and runs an algorithm that takes the measurements of  $y$  to estimate the state  $x$  of the physical process. In Figure 27 the scheme of the simulation is presented. We consider an estimation algorithm with a state variable  $\hat{x} \in \mathbb{R}^{n_P}$ , which denotes the estimate of  $x$ , that is appropriately reset to a new value involving the information received. More precisely, denoting the transmission times by  $t_i$  and  $L$  a constant matrix to be designed, the estimation algorithm updates its state as follows

$$\hat{x}^+ = \hat{x} + L(y - M\hat{x}) \quad (36)$$

at every instant information is received. In between such events, the algorithm updates its state continuously so as to match the evolution of the state of the physical process, that is, via

$$\dot{\hat{x}} = A\hat{x} \quad (37)$$

Modeling the network as a hybrid system, which, in particular, assumes zero transmission delay, the state variables of the entire system are  $j_N, \tau_N \in \mathbb{R}, m_N \in \mathbb{R}^{r_P}$ , and  $x, \hat{x} \in \mathbb{R}^{n_P}$ . Then, transmissions occur when  $\tau_N \leq 0$ , at which events the state of the network is updated via

$$\tau_N^+ \in [T_N^{*\min}, T_N^{*\max}], \quad j_N^+ = j_N + 1, \quad m_N^+ = y$$

and the state of the algorithm is updated via (36). Note that since the state of the physical process does not change at such events, we can use the following trivial difference equation to update it at the network events:

$$x^+ = x$$

In between events, the state of the network is updated as

$$\dot{\tau}_N = -1, \quad \dot{j}_N, \quad \dot{m}_N = 0$$

the state of the algorithm changes continuously according to (37), and the state of the physical process changes according to (35). Considering the equations above, we arbitrarily pick the following data for each of the subsystems in Figure 27:

- Physical process:

$$f_P(x, u) := Ax + Bu, \quad C_P := \mathbb{R}^{n_p} \times \mathbb{R} \quad (38)$$

$$G_P(x, u) := x, \quad D_P := \emptyset, \quad y = h(x) := Mx \quad (39)$$

where

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ -2 & 1 & -1 & 0 \\ 2 & -2 & 0 & -2 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad M = [1 \ 0 \ 0 \ 0], \quad (40)$$

$n_p = 4$ ,  $r_p = 1$ ,  $x \in \mathbb{R}^{n_p}$ ,  $y \in \mathbb{R}^{r_p}$ , and  $u \in \mathbb{R}$ .

- Network:

$$f(x_N, u_N) := \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}, \quad C := \{(x_N, u_N) \mid \tau_N \geq 0\}, \quad (41)$$

$$g(x_N, u_N) := \begin{bmatrix} u_N \\ j_N + 1 \\ \tau_r \end{bmatrix}, \quad D := \{(x_N, u_N) \mid \tau_N \leq 0\}, \quad y_N = h(x_N) := x_N \quad (42)$$

where  $\tau_r \in [T_N^{*\min}, T_N^{*\max}]$  is a random variable that models the time in-between communication instances. Also, the state and the input are given by  $x_N = (m_N, j_N, \tau_N) \in \mathbb{R} \times \mathbb{R} \times \mathbb{R}$ , and  $u_N = y \in \mathbb{R}^{r_p}$ , respectively.

- Estimator:

$$f(x_E, u_E) := \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix} x_E + \begin{bmatrix} B \\ 0 \end{bmatrix} v, \quad C := \{(x_E, u_E) \mid j_E = j_N\}, \quad (43)$$

$$g((\hat{x}, j_E), u_E) := \begin{bmatrix} \hat{x} + L(m_N - M\hat{x}) \\ j_N \end{bmatrix}, \quad D := \{(x, u) \mid j_E \in \mathbb{N} \setminus j_N\}, \quad (44)$$

where  $L$ , which is designed as in [6], is given by

$$L := \begin{bmatrix} 1 \\ -0.9433 \\ -0.6773 \\ 1.6274 \end{bmatrix}, \quad (45)$$

the input and the state are given by  $u_E = (x_N, v) = ((m_N, j_N, \tau_N), v) \in \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R}$ , and  $x_E = (\hat{x}, j_E) \in \mathbb{R}^4 \times \mathbb{R}$ , respectively. Notice that the estimator block (Estimator) in Figure 27 is implemented using a regular hybrid system block with embedded functions.



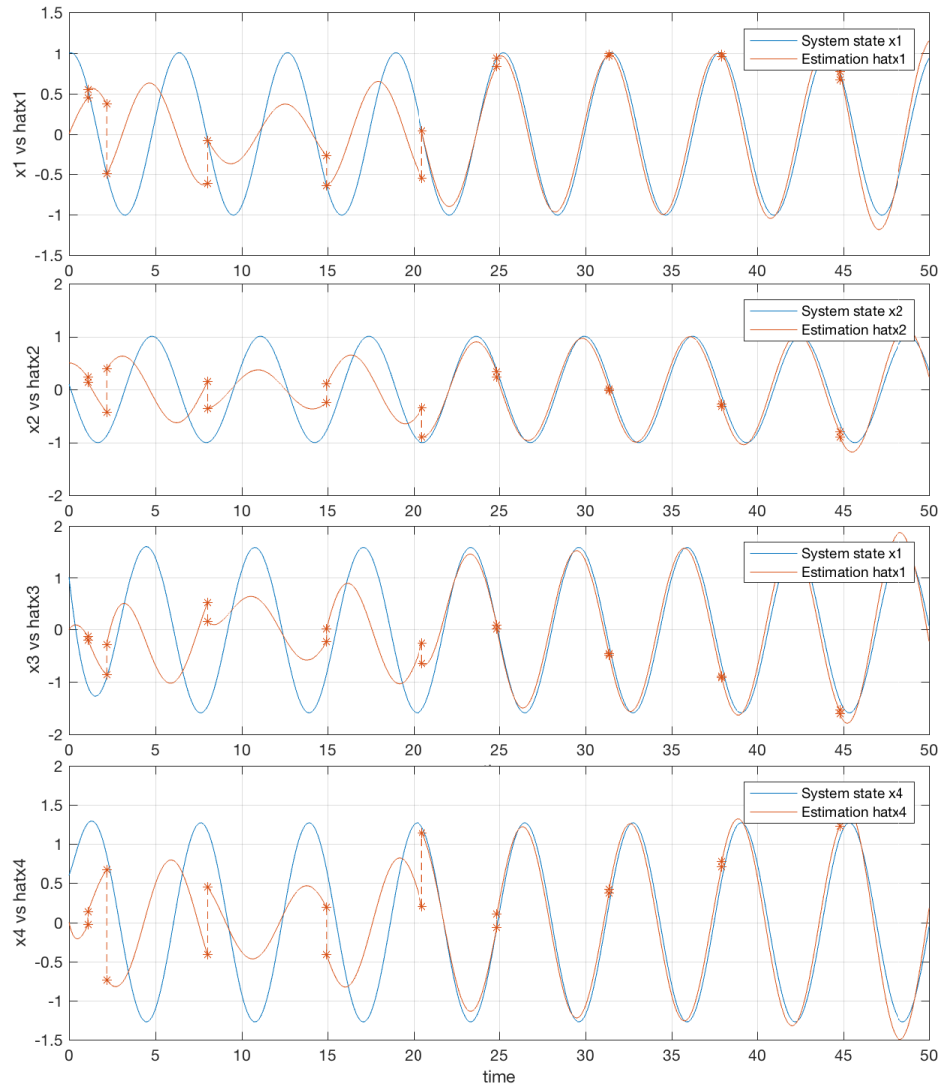


Figure 28: Estimated states Vs. continuous plant states

For each hybrid system in Figure 27 (HSu, network, and Estimator) we have the following Matlab embedded functions that describe the sets  $C$  and  $D$  and functions  $f$  and  $g$ . Figure 28 depicts the corresponding hybrid arc for the position state.

- Continuous process:

Flow map

```

1 function zdot = f(z, u, ctes)
2 %-----
3 % Matlab M-file Project: HyEQ Toolbox @ Hybrid Systems Laboratory (HSL),
4 % https://hybrid.soe.ucsc.edu/software
5 % http://hybridsimulator.wordpress.com/
6 %-----
7 % Project: Simulation of a hybrid system (plant with constraints in the
8 % state and the input)
9 % Description: Flow map

```

```

10 %-----
11 %-----
12 % See also HYEQSOLVER, PLOTARC, PLOTARC3, PLOTFLows, PLOTHARC,
13 % PLOTHARCCOLOR, PLOTHARCCOLOR3D, PLOTHYBRIDARC, PLOTJUMPS.
14 % Copyright @ Hybrid Systems Laboratory (HSL),
15 % Revision: 0.0.0.3 Date: 05/20/2015 3:42:00
16 %-----
17 % flow map: xdot=f(x,u);
18 % ctes = [A,B,M',L,P];
19
20 A = ctes(:,1:4);
21 B = ctes(:,5);
22 % M = ctes(:,6)';
23 % L = ctes(:,7);
24
25 zdot = A*z + B*u;

```

#### Flow set

```

1 function v = C(x, u)
2 %-----
3 % Matlab M-file Project: HyEQ Toolbox @ Hybrid Systems Laboratory (HSL),
4 % https://hybrid.soe.ucsc.edu/software
5 % http://hybridsimulator.wordpress.com/
6 %-----
7 % Project: Simulation of a hybrid system
8 % Description: Flow set
9 %-----
10 %-----
11 % See also HYEQSOLVER, PLOTARC, PLOTARC3, PLOTFLows, PLOTHARC,
12 % PLOTHARCCOLOR, PLOTHARCCOLOR3D, PLOTHYBRIDARC, PLOTJUMPS.
13 % Copyright @ Hybrid Systems Laboratory (HSL),
14 % Revision: 0.0.0.3 Date: 05/20/2015 3:42:00
15 %
16 % Check on flow conditions
17 % E.g.,
18 % if (x(1) >= u(1)) % flow condition
19 %     v = 1; % report flow
20 % else
21 %     v = 0; % do not report flow
22 % end
23
24
25 v = 1; % report flow
26

```

#### Jump map

```

1 function xplus = g(x, u)
2 %-----
3 % Matlab M-file Project: HyEQ Toolbox @ Hybrid Systems Laboratory (HSL),
4 % https://hybrid.soe.ucsc.edu/software
5 % http://hybridsimulator.wordpress.com/
6 %-----
7 % Project: Simulation of a hybrid system

```

```

8 % Description: Jump map
9 %-----
10 %-----
11 % See also HYEQSOLVER, PLOTARC, PLOTARC3, PLOTFLows, PLOTHARC,
12 % PLOTHARCCOLOR, PLOTHARCCOLOR3D, PLOTHYBRIDARC, PLOTJUMPS.
13 % Copyright @ Hybrid Systems Laboratory (HSL),
14 % Revision: 0.0.0.3 Date: 05/20/2015 3:42:00
15 %-----
16 % jump map: xplus = g(x,u);
17
18 xplus = zeros(4,1);
19 xplus = x;
20
21

```

#### Jump set

```

1 function v = D(x, u)
2 %-----
3 % Matlab M-file Project: HyEQ Toolbox @ Hybrid Systems Laboratory (HSL),
4 % https://hybrid.soe.ucsc.edu/software
5 % http://hybridsimulator.wordpress.com/
6 %-----
7 % Project: Simulation of a hybrid system
8 % Description: Jump set
9 %-----
10 %-----
11 % See also HYEQSOLVER, PLOTARC, PLOTARC3, PLOTFLows, PLOTHARC,
12 % PLOTHARCCOLOR, PLOTHARCCOLOR3D, PLOTHYBRIDARC, PLOTJUMPS.
13 % Copyright @ Hybrid Systems Laboratory (HSL),
14 % Revision: 0.0.0.3 Date: 05/20/2015 3:42:00
15 %
16 % Check on jump conditions
17 % % E.g.,
18 % if (x(1) <= u(1)) && (x(2) <= 0) % jump condition
19 %     v = 1; % report jump
20 % else
21 %     v = 0; % do not report jump
22 % end
23
24 v = 0; % do not report jump

```

- Network:

#### Flow map

```

1 function xdot = f(x,vs)
2 %-----
3 % Matlab M-file Project: HyEQ Toolbox @ Hybrid Systems Laboratory (HSL),
4 % https://hybrid.soe.ucsc.edu/software
5 % http://hybridsimulator.wordpress.com/
6 %-----
7 % Project: Simulation of a hybrid system Digital network (net)
8 % Description: Flow map
9 %-----

```

```

10 %-----
11 % See also HYEQSOLVER, PLOTARC, PLOTARC3, PLOTFLows, PLOTHARC,
12 % PLOTHARCCOLOR, PLOTHARCCOLOR3D, PLOTHYBRIDARC, PLOTJUMPS.
13 % Copyright @ Hybrid Systems Laboratory (HSL),
14 % Revision: 0.0.0.3 Date: 05/20/2015 3:42:00
15
16 n = length(vs); % measured input size
17
18 msdot = zeros(n,1); % measured continuous dynamics
19 jdot = 0;
20 tau_sdot = -1; % Timer tau_s
21 xdot = [msdot;jdot;tau_sdot];
22
Flow set
1 function v = C(x, vs, Tnmax)
2 %-----
3 % Matlab M-file Project: HyEQ Toolbox @ Hybrid Systems Laboratory (HSL),
4 % https://hybrid.soe.ucsc.edu/software
5 % http://hybridsimulator.wordpress.com/
6 %-----
7 % Project: Simulation of a hybrid system Digital network (net)
8 % Description: Flow set
9 %-----
10 %-----
11 % See also HYEQSOLVER, PLOTARC, PLOTARC3, PLOTFLows, PLOTHARC,
12 % PLOTHARCCOLOR, PLOTHARCCOLOR3D, PLOTHYBRIDARC, PLOTJUMPS.
13 % Copyright @ Hybrid Systems Laboratory (HSL),
14 % Revision: 0.0.0.3 Date: 05/20/2015 3:42:00
15 %
16 % Check on flow conditions
17 % E.g.,
18 % if (x(1) >= u(1)) % flow condition
19 %     v = 1; % report flow
20 % else
21 %     v = 0; % do not report flow
22 % end
23
24 tau_s = x(end); % timer state
25
26 if tau_s>=0 && tau_s<= Tnmax
27     v = 1; % report flow
28 elseif tau_s> Tnmax
29     v = 0; % do not report flow
30 else
31     v = 0;
32 end
end

Jump map
1 function xplus = g(x, vs, Tnmax, Tnmin, tk)
2 %-----
3 % Matlab M-file Project: HyEQ Toolbox @ Hybrid Systems Laboratory (HSL),
4 % https://hybrid.soe.ucsc.edu/software

```

```

5 % http://hybridsimulator.wordpress.com/
6 %-----
7 % Project: Simulation of a hybrid system Analog-to-Digital converter (ADC)
8 % Description: Jump map
9 %-----
10 %-----
11 % See also HYEQSOLVER, PLOTARC, PLOTARC3, PLOTFLows, PLOTHARC,
12 % PLOTHARCCOLOR, PLOTHARCCOLOR3D, PLOTHYBRIDARC, PLOTJUMPS.
13 % Copyright @ Hybrid Systems Laboratory (HSL),
14 % Revision: 0.0.0.3 Date: 05/20/2015 3:42:00
15
16 n = length(vs); % measured input size
17 xtemp = zeros(n+2,1);
18 xtemp = x;
19 x = xtemp;
20
21
22 j = x(n+1);
23 msplus = vs; % output = measured input
24 % The value tau_s is updated as a function of vs, e.g.,
25 tau_splus = tk(j+1); % Timer tau_s
26 j_plus = j+1;
27 xplus = [msplus;j_plus;tau_splus];
28
29
30

```

#### Jump set

```

1 function v = D(x, vs)
2 %-----
3 % Matlab M-file Project: HyEQ Toolbox @ Hybrid Systems Laboratory (HSL),
4 % https://hybrid.soe.ucsc.edu/software
5 % http://hybridsimulator.wordpress.com/
6 %-----
7 % Project: Simulation of a hybrid system Analog-to-Digital converter (ADC)
8 % Description: Jump set
9 %-----
10 %-----
11 % See also HYEQSOLVER, PLOTARC, PLOTARC3, PLOTFLows, PLOTHARC,
12 % PLOTHARCCOLOR, PLOTHARCCOLOR3D, PLOTHYBRIDARC, PLOTJUMPS.
13 % Copyright @ Hybrid Systems Laboratory (HSL),
14 % Revision: 0.0.0.3 Date: 05/20/2015 3:42:00
15 %
16 % Check on jump conditions
17 % % E.g.,
18 % if (x(1) <= u(1)) && (x(2) <= 0) % jump condition
19 %     v = 1; % report jump
20 % else
21 %     v = 0; % do not report jump
22 % end
23
24
25 tau_s = x(end); % timer state

```

```

26
27 if tau_s>=0
28     v = 0; % do not report jump
29 elseif tau_s<= 0
30     v = 1; % report jump
31 else
32     v = 0;
33 end

```

- Estimator:

Flow map

```

1 function xdot = f(x, v,ctes)
2 %-----
3 % Matlab M-file Project: HyEQ Toolbox @ Hybrid Systems Laboratory (HSL),
4 % https://hybrid.soe.ucsc.edu/software
5 % http://hybridsimulator.wordpress.com/
6 %-----
7 % Project: Simulation of a hybrid system
8 % Description: Flow map
9 %-----
10 %-----
11 % See also HYEQSOLVER, PLOTARC, PLOTARC3, PLOTFLows, PLOTHARC,
12 % PLOTHARCCOLOR, PLOTHARCCOLOR3D, PLOTHYBRIDARC, PLOTJUMPS.
13 % Copyright @ Hybrid Systems Laboratory (HSL),
14 % Revision: 0.0.0.3 Date: 05/20/2015 3:42:00
15 %-----
16 % flow map: xdot=f(x,u);
17 % ctes = [A,B,M',L,P];
18
19 A = ctes(:,1:4);
20 B = ctes(:,5);
21 % M = ctes(:,6)';
22 % L = ctes(:,7);
23 n = length(A);
24 u = v(end);
25 z = x(1:n);
26 jdot = 0;
27 zdot = A*z + B*u;
28 xdot = [zdot;jdot];

```

Flow set

```

1 function out = C(x, v)
2 %-----
3 % Matlab M-file Project: HyEQ Toolbox @ Hybrid Systems Laboratory (HSL),
4 % https://hybrid.soe.ucsc.edu/software
5 % http://hybridsimulator.wordpress.com/
6 %-----
7 % Project: Simulation of a hybrid system
8 % Description: Flow set
9 %-----
10 %-----
11 % See also HYEQSOLVER, PLOTARC, PLOTARC3, PLOTFLows, PLOTHARC,

```

```

12 % PLOTHARCCOLOR, PLOTHARCCOLOR3D, PLOTHYBRIDARC, PLOTJUMPS.
13 % Copyright @ Hybrid Systems Laboratory (HSL),
14 % Revision: 0.0.0.3 Date: 05/20/2015 3:42:00
15 %
16 % Check on flow conditions
17 % E.g.,
18 % if (x(1) >= u(1)) % flow condition
19 %     v = 1; % report flow
20 % else
21 %     v = 0; % do not report flow
22 % end
23
24 j = x(end); % internal communication memory event
25 jnet = v(end-2); % communication event
26
27 if j == jnet
28     out = 1; % report flow
29 else
30     out = 0;
31 end

```

#### Jump map

```

1 function xplus = g(x, v, ctes)
2 %-----
3 % Matlab M-file Project: HyEQ Toolbox @ Hybrid Systems Laboratory (HSL),
4 % https://hybrid.soe.ucsc.edu/software
5 % http://hybridsimulator.wordpress.com/
6 %-----
7 % Project: Simulation of a hybrid system
8 % Description: Jump map
9 %-----
10 %-----
11 % See also HYEQSOLVER, PLOTARC, PLOTARC3, PLOTFLAWS, PLOTHARC,
12 % PLOTHARCCOLOR, PLOTHARCCOLOR3D, PLOTHYBRIDARC, PLOTJUMPS.
13 % Copyright @ Hybrid Systems Laboratory (HSL),
14 % Revision: 0.0.0.3 Date: 05/20/2015 3:42:00
15 %-----
16 % jump map: xplus = g(x,u);
17
18 A = ctes(:,1:4);
19 M = ctes(:,6)';
20 L = ctes(:,7);
21
22 n = 4;
23 z = zeros(4,1);
24 zplus = zeros(4,1);
25 xplus = zeros(5,1);
26 z = x(1:4);
27 jnet = v(2); % communication event
28 y = v(1);
29
30 jplus = jnet;
31 zplus = z + L*(y-M*z);

```

```
32 xplus = [zplus;jplus];
```

Jump set

```
1 function out = D(x, v)
2 %-----
3 % Matlab M-file Project: HyEQ Toolbox @ Hybrid Systems Laboratory (HSL),
4 % https://hybrid.soe.ucsc.edu/software
5 % http://hybridsimulator.wordpress.com/
6 %-----
7 % Project: Simulation of a hybrid system
8 % Description: Jump set
9 %-----
10 %-----
11 % See also HYEQSOLVER, PLOTARC, PLOTARC3, PLOTFLows, PLOTHARC,
12 % PLOTHARCCOLOR, PLOTHARCCOLOR3D, PLOTHYBRIDARC, PLOTJUMPS.
13 % Copyright @ Hybrid Systems Laboratory (HSL),
14 % Revision: 0.0.0.3 Date: 05/20/2015 3:42:00
15 %
16 % Check on jump conditions
17 % % E.g.,
18 % if (x(1) <= u(1)) && (x(2) <= 0) % jump condition
19 %     v = 1; % report jump
20 % else
21 %     v = 0; % do not report jump
22 % end
23
24
25 n = length(x)-1; % measured input size
26 j = x(end); % internal communication memory event
27 jnet = v(end-2); % communication event
28
29 if j ~= jnet
30     out = 1; % report jump
31 else
32     out = 0;
33 end
34
```

□

**Example 5.10** (Sample-and-hold Feedback Control) Consider a physical process modeled as in Example 5.7 and a state feedback interconnection. The algorithm (static gain) uses measurements of its output and controls the input of the physical process with the goal of steering its state to zero. Suppose the sampling device is ideal and that the signals are connected to the plant via a DAC modeled as in (10)-(11).

The interconnection between the models of the physical process, the sampling device, the finite state machine, and the DAC has the feedback topology shown in Figure 29. In particular, the output of the DAC is connected to the input  $u$  of the physical process by a matrix gain  $K$ , while the input  $v$  of the finite state machine is equal to the output  $y$  of the physical process at every sampling instant.

- Physical process:

$$f_P(x, u) := Ax + Bu, \quad C_P := \mathbb{R}^2 \times \mathbb{R}^2, \quad (46)$$

$$G_P(x, u) := x, \quad D_P := \emptyset, \quad y = h(x) := x \quad (47)$$



where  $A = \begin{bmatrix} 0 & 1 \\ 0 & -b/m \end{bmatrix}$ ,  $B = [1 \quad 1/m]$ , and  $x = (x_1, x_2) \in \mathbb{R}^2$ .

- Analog-to-Digital Converter (ADC):

$$f(x, u) := \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad C := \{(x, u) \mid \tau_s \in [0, T_s^*]\}, \quad (48)$$

$$g(x, u) := \begin{bmatrix} u \\ 0 \end{bmatrix}, \quad D := \{(x, u) \mid \tau_s \geq T_s^*\}, \quad y = h(x) := x \quad (49)$$

where  $x = (m_s, \tau_s) \in \mathbb{R} \times \mathbb{R}^2$ , and  $u \in \mathbb{R}^2$ .

- Zero-Order Hold (ZOH):

$$f(x, u) := \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad C := \{(x, u) \mid \tau_s \in [0, T_s^*]\}, \quad (50)$$

$$g(x, u) := \begin{bmatrix} u \\ 0 \end{bmatrix}, \quad D := \{(x, u) \mid \tau_s \geq T_s^*\}, \quad y = h(x) := x \quad (51)$$

where  $x = (m_s, \tau_s) \in \mathbb{R} \times \mathbb{R}^2$ , and  $u \in \mathbb{R}^2$ .

The feedback gain is given by  $K = \begin{bmatrix} k_1 \\ k_2 \end{bmatrix}$ .

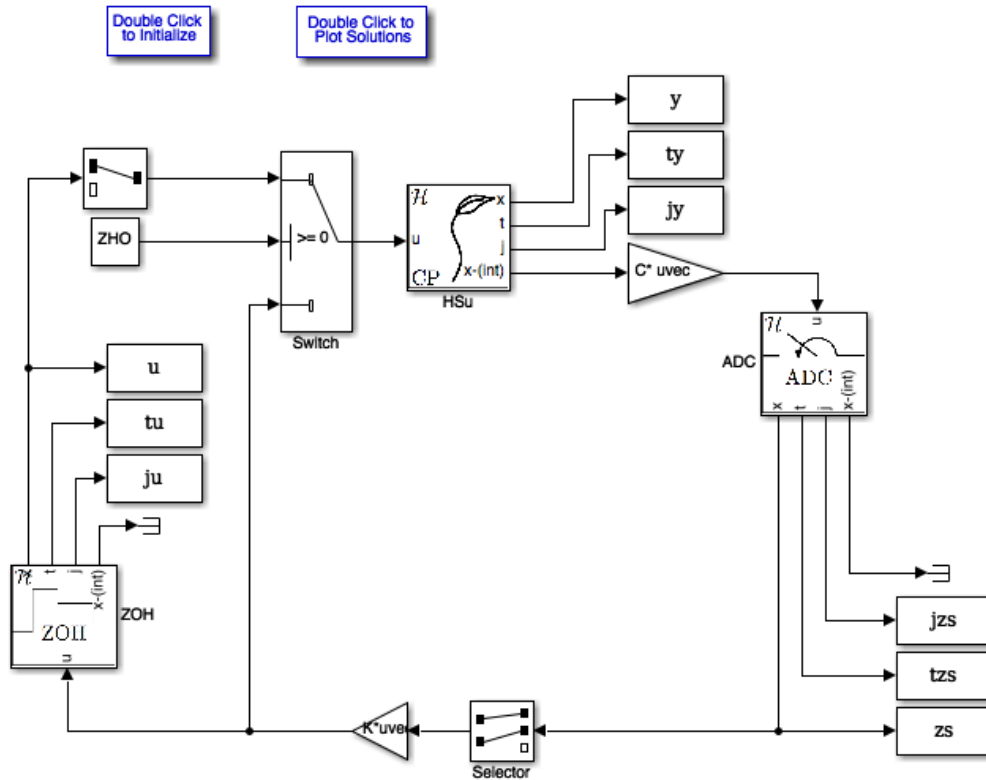


Figure 29: Sample and hold feedback interconnection for Example 5.10

For each hybrid system in Figure 29 (HSu, ADC, and ZOH) we have the following Matlab embedded functions that describe the sets  $C$  and  $D$  and functions  $f$  and  $g$ . Figure 30 presents a numerical simulation of the interconnected system.

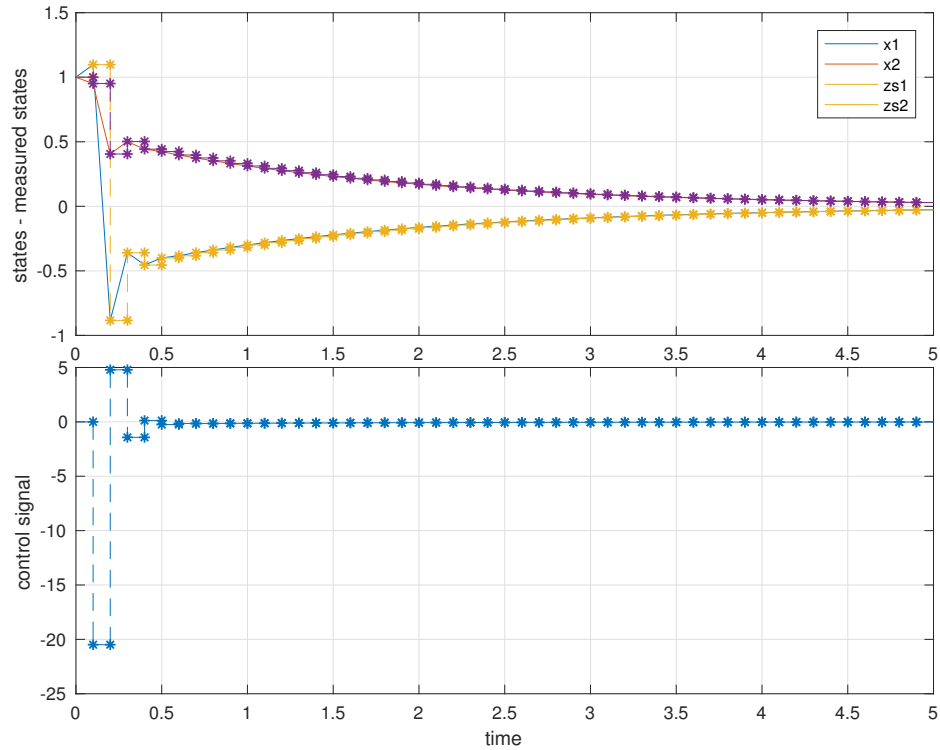


Figure 30: Measured states and inputs.

- Continuous process:

Flow map

```

1 function xdot = f(x, u, ctes)
2 %-----
3 % Matlab M-file Project: HyEQ Toolbox @ Hybrid Systems Laboratory (HSL),
4 % https://hybrid.soe.ucsc.edu/software
5 % http://hybridsimulator.wordpress.com/
6 %-----
7 % Project: Simulation of a hybrid system (plant with constraints in the
8 % state and the input)
9 % Description: Flow map
10 %-----
11 %-----
12 % See also PLOTARC, PLOTARC3, PLOTFLAWS, PLOTHARC, PLOTHARCCOLOR,
13 % PLOTHARCCOLOR3D, PLOTHYBRIDARC, PLOTJUMPS.
14 % Copyright @ Hybrid Systems Laboratory (HSL),
15 % Revision: 0.0.0.3 Date: 05/20/2015 3:42:00
16 %-----
17 % flow map: xdot=f(x,u);
18
19 % ctes = [A,B,C,K'];
20
21 A = ctes(:,1:2);
22 B = ctes(:,3);
23 C = ctes(:,4:5);

```

```

24 K = ctes(:,6)';
25
26 xdot = A*x + B*u;

```

#### Flow set

```

1 function v = C(x, u)
2 %-----
3 % Matlab M-file Project: HyEQ Toolbox @ Hybrid Systems Laboratory (HSL),
4 % https://hybrid.soe.ucsc.edu/software
5 % http://hybridsimulator.wordpress.com/
6 %-----
7 % Project: Simulation of a hybrid system
8 % Description: Flow set
9 %-----
10 %-----
11 % See also PLOTARC, PLOTARC3, PLOTFLows, PLOTHARC, PLOTHARCCOLOR,
12 % PLOTHARCCOLOR3D, PLOTHYBRIDARC, PLOTJUMPS.
13 % Copyright @ Hybrid Systems Laboratory (HSL),
14 % Revision: 0.0.0.3 Date: 05/20/2015 3:42:00
15 %
16 % Check on flow conditions
17 % E.g.,
18 % if (x(1) >= u(1)) % flow condition
19 %     v = 1; % report flow
20 % else
21 %     v = 0; % do not report flow
22 % end
23
24
25 v = 1; % report flow
26

```

#### Jump map

```

1 function xplus = g(x, u)
2 %-----
3 % Matlab M-file Project: HyEQ Toolbox @ Hybrid Systems Laboratory (HSL),
4 % https://hybrid.soe.ucsc.edu/software
5 % http://hybridsimulator.wordpress.com/
6 %-----
7 % Project: Simulation of a hybrid system
8 % Description: Jump map
9 %-----
10 %-----
11 % See also PLOTARC, PLOTARC3, PLOTFLows, PLOTHARC, PLOTHARCCOLOR,
12 % PLOTHARCCOLOR3D, PLOTHYBRIDARC, PLOTJUMPS.
13 % Copyright @ Hybrid Systems Laboratory (HSL),
14 % Revision: 0.0.0.3 Date: 05/20/2015 3:42:00
15 %-----
16 % jump map: xplus = g(x,u);
17
18 xplus = x;
19
20

```

### Jump set

```
1 function v = D(x, u)
2 %-----
3 % Matlab M-file Project: HyEQ Toolbox @ Hybrid Systems Laboratory (HSL),
4 % https://hybrid.soe.ucsc.edu/software
5 % http://hybridsimulator.wordpress.com/
6 %-----
7 % Project: Simulation of a hybrid system
8 % Description: Jump set
9 %-----
10 %-----
11 % See also PLOTARC, PLOTARC3, PLOTFLows, PLOTHARC, PLOTHARCCOLOR,
12 % PLOTHARCCOLOR3D, PLOTHYBRIDARC, PLOTJUMPS.
13 % Copyright @ Hybrid Systems Laboratory (HSL),
14 % Revision: 0.0.0.3 Date: 05/20/2015 3:42:00
15 %
16 % Check on jump conditions
17 % % E.g.,
18 % if (x(1) <= u(1)) && (x(2) <= 0) % jump condition
19 %     v = 1; % report jump
20 % else
21 %     v = 0; % do not report jump
22 % end
23
24 v = false; % do not report jump
```

- ADC and ZOH:

### Flow map

```
1 function xdot = f(x, vs)
2 %-----
3 % Matlab M-file Project: HyEQ Toolbox @ Hybrid Systems Laboratory (HSL),
4 % https://hybrid.soe.ucsc.edu/software
5 % http://hybridsimulator.wordpress.com/
6 %-----
7 % Project: Simulation of a hybrid system Analog-to-Digital converter (ADC)
8 % Description: Flow map
9 %-----
10 %-----
11 % See also HYEQSOLVER, PLOTARC, PLOTARC3, PLOTFLows, PLOTHARC,
12 % PLOTHARCCOLOR, PLOTHARCCOLOR3D, PLOTHYBRIDARC, PLOTJUMPS.
13 % Copyright @ Hybrid Systems Laboratory (HSL),
14 % Revision: 0.0.0.3 Date: 05/20/2015 3:42:00
15
16 n = length(vs); % measured input size
17
18 msdot = zeros(n,1); % measured continuous dynamics
19 tau_sdot = 1; % Timer tau_s
20 xdot = [msdot;tau_sdot];
21
```

### Flow set

```

1 function v = C(x, vs, Ts)
2 %-----
3 % Matlab M-file Project: HyEQ Toolbox @ Hybrid Systems Laboratory (HSL),
4 % https://hybrid.soe.ucsc.edu/software
5 % http://hybridsimulator.wordpress.com/
6 %-----
7 % Project: Simulation of a hybrid system Analog-to-Digital converter (ADC)
8 % Description: Flow set
9 %-----
10 %-----
11 % See also HYEQSOLVER, PLOTARC, PLOTARC3, PLOTFLAWS, PLOTHARC,
12 % PLOTHARCCOLOR, PLOTHARCCOLOR3D, PLOTHYBRIDARC, PLOTJUMPS.
13 % Copyright @ Hybrid Systems Laboratory (HSL),
14 % Revision: 0.0.0.3 Date: 05/20/2015 3:42:00
15 %
16 % Check on flow conditions
17 % E.g.,
18 % if (x(1) >= u(1)) % flow condition
19 %     v = 1; % report flow
20 % else
21 %     v = 0; % do not report flow
22 % end
23
24 n = length(vs); % measured input size
25
26 tau_s = x(end); % timer state
27
28 if tau_s>=0 && tau_s<= Ts
29     v = 1; % report flow
30 elseif tau_s> Ts
31     v = 0; % do not report flow
32 else
33     v = 0;
34 end

```

#### Jump map

```

1 function xplus = g(x, vs)
2 %-----
3 % Matlab M-file Project: HyEQ Toolbox @ Hybrid Systems Laboratory (HSL),
4 % https://hybrid.soe.ucsc.edu/software
5 % http://hybridsimulator.wordpress.com/
6 %-----
7 % Project: Simulation of a hybrid system Analog-to-Digital converter (ADC)
8 % Description: Jump map
9 %-----
10 %-----
11 % See also HYEQSOLVER, PLOTARC, PLOTARC3, PLOTFLAWS, PLOTHARC,
12 % PLOTHARCCOLOR, PLOTHARCCOLOR3D, PLOTHYBRIDARC, PLOTJUMPS.
13 % Copyright @ Hybrid Systems Laboratory (HSL),
14 % Revision: 0.0.0.3 Date: 05/20/2015 3:42:00
15
16
17 n = length(vs) % measured input size

```

```

18
19 msplus = vs; % output = measured input
20 tau_splus = 0; % Timer tau_s
21 xplus = [msplus;tau_splus];
22
23
24

```

#### Jump set

```

1 function v = D(x, vs, Ts)
2 %-----
3 % Matlab M-file Project: HyEQ Toolbox @ Hybrid Systems Laboratory (HSL),
4 % https://hybrid.soe.ucsc.edu/software
5 % http://hybridsimulator.wordpress.com/
6 %-----
7 % Project: Simulation of a hybrid system Analog-to-Digital converter (ADC)
8 % Description: Jump set
9 %-----
10 %-----
11 % See also HYEQSOLVER, PLOTARC, PLOTARC3, PLOTFLAWS, PLOTHARC,
12 % PLOTHARCCOLOR, PLOTHARCCOLOR3D, PLOTHYBRIDARC, PLOTJUMPS.
13 % Copyright @ Hybrid Systems Laboratory (HSL),
14 % Revision: 0.0.0.3 Date: 05/20/2015 3:42:00
15 %
16 % Check on jump conditions
17 % % E.g.,
18 % if (x(1) <= u(1)) && (x(2) <= 0) % jump condition
19 %     v = 1; % report jump
20 % else
21 %     v = 0; % do not report jump
22 % end
23
24 n = length(vs); % measured input size
25
26 tau_s = x(end); % timer state
27
28 if tau_s>=0 && tau_s<= Ts
29     v = 0; % do not report jump
30 elseif tau_s> Ts
31     v = 1; % report jump
32 else
33     v = 0;
34 end

```

There are numerous practical examples of systems that can be modeled within the general model for sample-and-hold feedback control defined above. For example, one “classical” example is the control of the temperature of a room by turning on and off a heater so as to keep the temperature within a desired range; see the model in (29). Another widely known example is the control of the level of a water tank.

□

## 6 Further Reading

Installation files for the HyEQ Toolbox described in this paper can be found at MATLAB Central and at the author's website

<https://hybrid.soe.ucsc.edu/software>.

Also, resources and examples are shared by the HyEQ Toolbox users in the blog

<http://hybridsimulator.wordpress.com>.

## 7 Acknowledgments

We would like to thank Giampiero Campa for his thoughtful feedback and advice as well as Torstein Ingebrigtsen Bo for his comments and initial version of the lite simulator code. Also, we would like to include the list of people that help us to test this toolbox:

- Cenk Oguz Saglam - University of California, Santa Barbara
- Bharani Malladi - The University of Arizona

## References

- [1] R. G. Sanfelice, D. A. Copp, and P. Nanez. A toolbox for simulation of hybrid systems in Matlab/Simulink: Hybrid Equations (HyEQ) Toolbox. In *Proceedings of Hybrid Systems: Computation and Control Conference*, pages 101–106, 2013.
- [2] R. Goebel, R. Sanfelice, and A. Teel. Hybrid dynamical systems. *Control Systems, IEEE*, 29(2):28–93, april 2009.
- [3] Ricardo G. Sanfelice and Andrew R. Teel. Dynamical properties of hybrid systems simulators. *Automatica*, 46(2):239–248, 2010.
- [4] R. G. Sanfelice. Interconnections of hybrid systems: Some challenges and recent results. *Journal of Nonlinear Systems and Applications*, 2(1-2):111–121, 2011.
- [5] R. G. Sanfelice. Cmpe142 Class Notes - Introduction to Cyber-Physical Systems. <https://hybrid.soe.ucsc.edu/cmpe149-249-2016>, 2017.
- [6] F. Ferrante, F. Gouaisbaut, R. G. Sanfelice, and S. Tarbouriech. State estimation of linear systems in the presence of sporadic measurements. *Automatica*, 73:101–109, November 2016.