Hybrid Stability Automata

User Manual

Peter Uth

July 2017

Contents

1 Overview

A hybrid stability automaton (HSA) is a method of modeling a dynamical system as a series of control modes and switches that are dependent on stability criteria. By employing numerical continuation to access fundamental stability properties and determine stable and unstable operating conditions, HSA can provide valuable information for safety analyses of physical systems. Refer to [1, 2] for the complete details on HSA.

This document provides details on how to generate HSA using the MATLAB tools developed for this work. HSA can be created for the following types of systems:

- One-dimensional, one-parameter (1D1P)
- One-dimensional, multi-parameter (1DMP)
- Multi-dimensional, one-parameter (MD1P)
- Multi-dimensional, multi-parameter (MDMP)

Note that currently the multi-dimensional tools only work for two-dimensional systems. However, these tools are designed to accept higher-dimensional systems given some modifications planned as future work.

2 Installation

2.1 Dependencies

Dependencies for the HSA tools are listed below.

- MATLAB
  - symbolic math toolbox
- Continuation-based System Analysis tool (COSY)
  - `cosy.m`
  - `cm_pseudo_arclength.m`
  - `cm_solution.m`
- GraphViz [3, 4, 5]
- dot2tex (optional) [6]
  - Python 2.6 or 2.7
  - pyparsing
  - preview-latex
  - PGF/TikZ

2.2 Installation Process

With the dependencies from Section 2.1 installed, place the three COSY files into the working directory or add the containing folder to the MATLAB path. The folder containing the GraphViz file `dot.exe` must be added to the system's environmental variables. Note that the HSA code has only been tested using the Windows operating system.

3 One-Dimensional Systems

HSA for one-dimensional systems (i.e. systems with one state) can be generated using the `hsa1D.m` file. For one-parameter systems (i.e. 1D1P), the `hsa1D` function can be called directly using the results from a COSY continuation process. For multi-parameter systems (i.e. 1DMP), `dataMP.m` must first be called and the resulting data is then passed to `hsa1D`. Details on these HSA tools and examples for 1D1P and 1DMP systems are presented in the following sections.

3.1 hsa1D.m

The `hsa1D` function creates HSA for 1D systems.

3.1.1 Inputs

- **label** - Name of the file used for labeling outputs. Input as a string.
- **F** - System equation(s). Input as a function handle. Use data output of `dataMP` for multi-parameter systems.
- **B** - Solution branch data resulting from a COSY continuation analysis. Use data output from `dataMP` for multi-parameter systems.
- **BP** - Solution branch point data resulting from a COSY continuation analysis. Use data output from `dataMP` for multi-parameter systems.
- **P** - Parameter type selection:
  - For fixed-p case, input as the numerical p-value to be analyzed.
  - For general case, input as empty (i.e. `[]`).
- **opt** - User options, if desired. Input as struct with any or all of the following fields:
  - **dot2tex** - Enables LaTeX-style formatting in the visualization output. Input as a string (e.g. `opt.dot2tex = 'on'`). Alternatively, a string input alone can be used if this is the only desired option (e.g. pass 'dot2tex' to `hsa1D` instead of `opt`).

- **texlab** - Replaces states and parameters, default x and p, with other symbols for visualization purposes. Input as a cell with a replacement defined within each row (e.g. `opt.texlab = {'x' '\gamma ';'p_1' '\alpha '}` will show x as γ and p_1 as α). Requires dot2tex. A blank space should be entered at the end of a LaTeX symbol designation for proper formatting. A cell input alone can be used if this is the only desired option.
- **ztol** - Change zero tolerance. Default is `opt.ztol = 0.00001`.
- **PD** - Parameter designation matrix, used only for multi-parameter systems. An array input alone can be used if this is the only desired option. See `dataMP.m` description below for more detail.

3.1.2 Outputs

- Stability data for all modes as a struct.
- HSA visualization as a PNG file. If dot2tex is enabled, a PDF is also created.
- Executable function m-file containing the dynamical equations with stability information. Stability behavior is displayed in the command window when accessed (i.e. the value towards which the system converges or whether the system diverges depending on the current state).

3.2 dataMP.m

The `dataMP` function automatically runs the COSY continuation process for each parameter designation defined by the user. The output data can then be passed to the `hsa1D` function to produce a multi-parameter HSA.

3.2.1 Inputs

- **f** - System equation. Input as a function handle.
- **PD** - Parameter designation matrix. Each row initiates a COSY continuation analysis on subset dynamics created using the designations within each column (column index = parameter number). Each row must have one NaN designation to identify the parameter activated for continuation.
- **x0** - Initial state conditions for COSY. Input as column array where the number of elements is the same as the number of rows in the parameter designation matrix.
- **p0** - Initial conditions of the active parameter for COSY. Input as column array where the number of elements is the same as the number of rows in the parameter designation matrix.
- **plim** - Parameter limits for the COSY processes. Input as two-column array where the number of rows is the same as the number of rows in the parameter designation matrix. First column contains the lower limit, second column contains the upper limit.

3.2.2 Outputs

- **F** - Function cell. First entry is the original dynamics, subsequent entries are the subset dynamics as defined by the parameter designation matrix.
- **B** - COSY output solution branch data for all subset analyses.
- **BP** - COSY output solution branch point data for all subset analyses.

3.3 One-Parameter Examples

3.3.1 Saddle Node Bifurcation, Fixed-p

Code to create an HSA for the prototypical saddle node bifurcation system, $\dot{x} = p + x^2$, is shown below.

```
F = @(x,u,p) p+x^2;
x0 = -1;
p0 = -1;
c = cosy(F,1,0,1);
c.p_min = -1;
c.p_max = 1;
[B,BP] = c.trace_solution_branches(x0,[],p0);
mode = hsa1D('SN_fp',F,B,BP,-0.5);
```

Lines 1-7 identify the function handle, establish COSY settings, and run COSY to obtain the solution data. In line 8, this data is then passed to the `hsa1D` function. Also in line 8, the label `SN_fp` and the fixed-p value -0.5 are input to the `hsa1D` function. Note that no options were selected for this process, so `P` is the last input into `hsa1D`. Figure 1 shows the resulting output visualization.

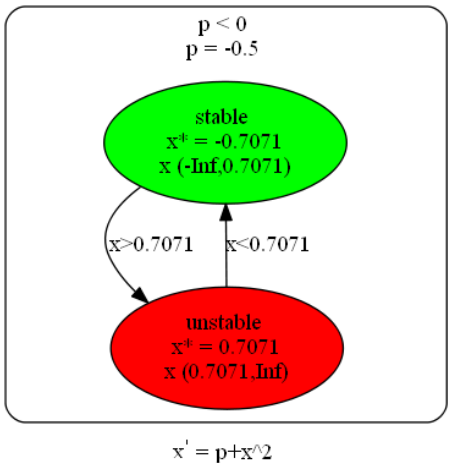

Figure 1: Hybrid stability automaton for a saddle node bifurcation, fixed-p case.

3.3.2 Saddle Node Bifurcation, General Case

To produce the general case automaton shown in Figure 2, line 8 from Section 3.3.1 is replaced with:

```
mode = hsa1D('SN_gen',F,B,BP,[]);
```

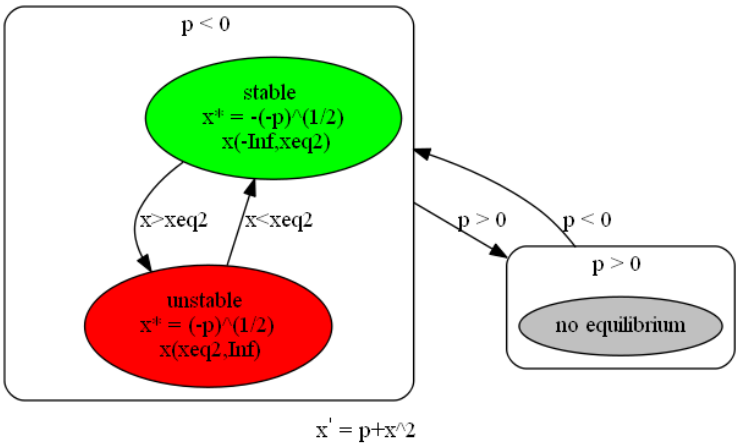

Figure 2: Hybrid stability automaton for a saddle node bifurcation, general case.

Figure 3 shows the general saddle node HSA with the dot2tex option enabled using the function call:

```
mode = hsa1D('SN_gen',F,B,BP,[],'dot2tex');
```

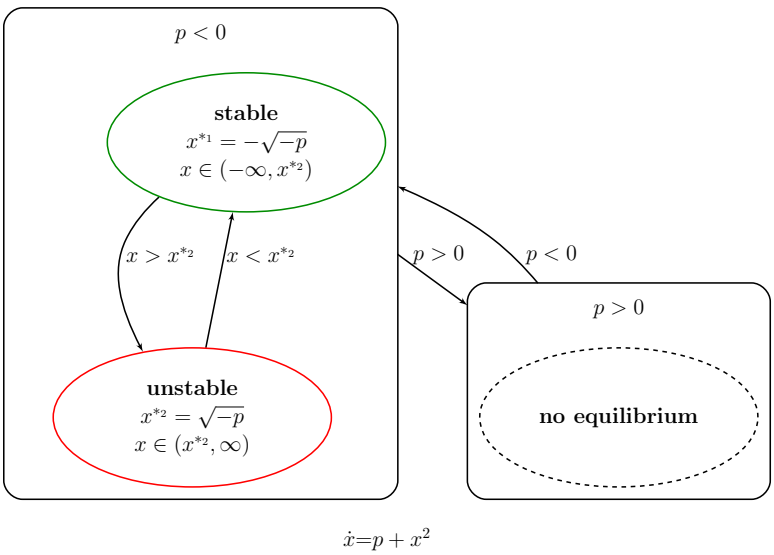

Figure 3: Hybrid stability automaton for a saddle node bifurcation with dot2tex enabled, general case.

3.4 Multi-Parameter Examples

3.4.1 General Case

Consider the 1DMP system $\dot{x} = p_1 + p_2 x + p_3 x^2 - p_4 x^3$ with the parameter designations

$$PD = \begin{bmatrix} NaN & 0 & 0 & 1 \\ 0 & NaN & 0 & 1 \\ NaN & 0 & 1 & 0 \end{bmatrix}, \tag{1}$$

which instructs the HSA tools to vary p_1 for cases 1 and 3, vary p_2 for case 2, and hold the other parameters as constants (0 or 1 in this example, but any constant values can be used). The following code can be used to create this system's HSA.

```
f = @(x,u,p) p(1)+p(2)*x+p(3)*x^2-p(4)*x^3;
PD = [ NaN 0 0 1 ; 0 NaN 0 1 ; NaN 0 1 0 ];
x0 = [-1;-1;-1];
p0 = [-1;-1;-1];
plim = [-1 1;-1 1;-1 1];
[F,B,BP] = dataMP(f,PD,x0,p0,plim);
mode = hsa1D('ODMP_gen',F,B,BP,[],PD);
```

Lines 1-5 set up the problem and establish settings for the continuation process. Line 6 calls the `dataMP` function to create the continuation results for the three subset dynamics. Line 7 takes these results and produces the HSA for the general case. Figure 4 shows the HSA.

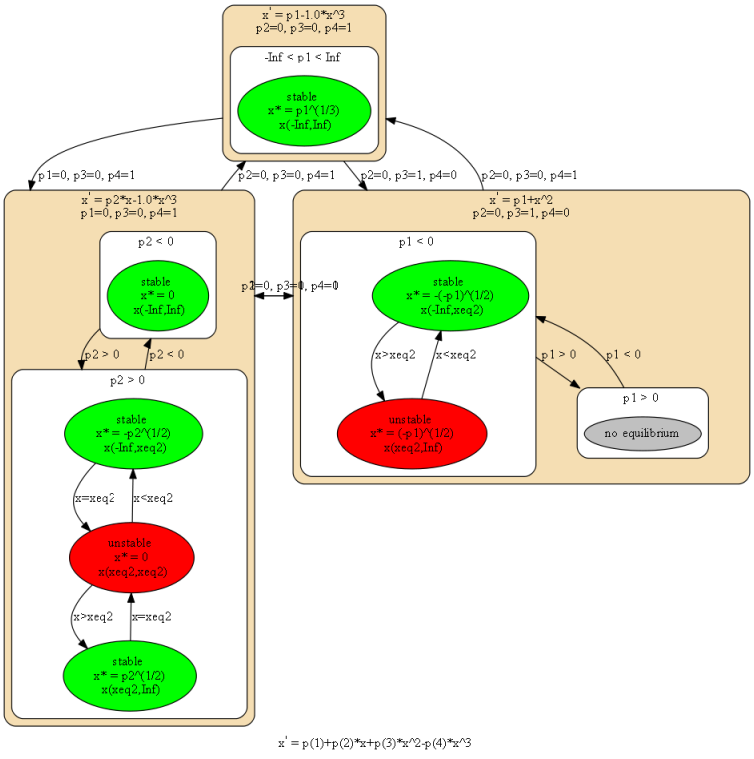

Figure 4: Hybrid stability automaton for a multi-parameter system, general case.

3.4.2 Fixed-p Case

To produce a fixed-p automaton of the system from Section 3.4.1 with $p = 0.5$, dot2tex enabled, and states/parameters replaced with symbols in the visualization, line 7 can be replaced with the following lines:

```
opt.PD = PD;
opt.dot2tex = 'on';
opt.texlab = {'x' '\gamma';'p_1' '\alpha';
                  'p_2' 'v';'p_3' 'h';'p_4' '\rho'};
mode = hsa1D('ODMP_fp',F,B,BP,0.5,opt);
```

The resulting HSA is shown in Figure 5.

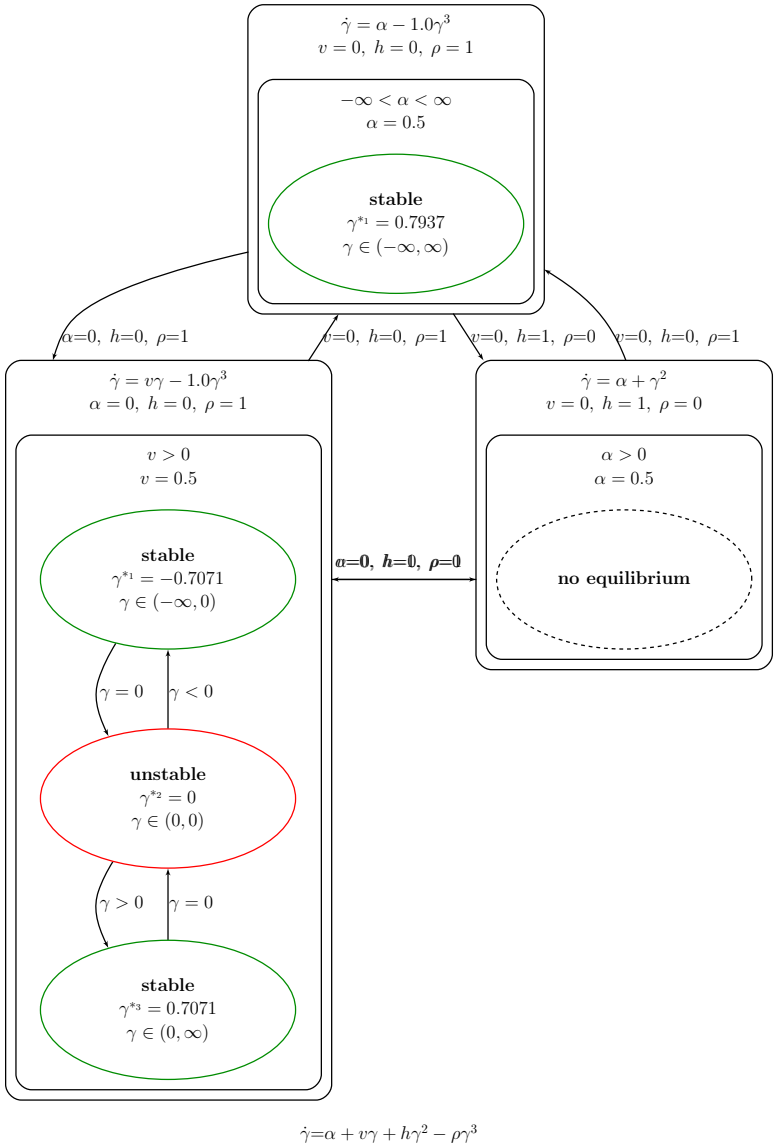

Figure 5: Hybrid stability automaton for a multi-parameter system, fixed-p case.

4 Multi-Dimensional Systems

HSA for multi-dimensional systems (i.e. systems with more than one state) can be generated using the `hsaMD.m` file. Similar to the 1D tools, the `hsaMD` function can be called directly for one-parameter systems (i.e. MD1P). For multi-parameter systems (i.e. MDMP), `dataMP` must first be called and the resulting data is then passed to `hsaMD`. A separate basin of attraction identification step can be applied using `basinMD.m` if desired. Details on the multi-dimensional tools and examples for MD1P and MDMP systems are presented in the following sections.

4.1 hsaMD.m

The `hsaMD` function creates HSA for MD systems. The inputs and outputs of this function are the same as `hsa1D` except for the following differences:

- The input function state must have multiple elements (e.g. `x(1)` and `x(2)`).
- The outputs do not explicitly define the basins of attraction. In the visualization, the basins are generically defined. In the executable file, the stability behavior is not displayed when accessed unless data from the `basinMD` function is available.

4.2 basinMD.m

Unlike for 1D systems, the basins of attraction for stable equilibria in MD systems cannot be automatically deduced from the continuation data output. The `basinMD` function can be used to approximate a basin using different methods. Currently, only repeated simulation methods are available, but the `basinMD.m` file is structured to allow the addition of new method selections. Basin identification only works for fixed-p cases.

4.2.1 Inputs

- **mode** - Output mode data from the `hsaMD` function.
- **type** - Basin of attraction identification method. Input as string. Currently available:
  - **Simulation** - Repeated simulation using fixed-step sampling.
  - **MonteCarlo** - Repeated simulation using random sampling.
- **xlimit** - The desired state limits for the basin identification process. Input as array where the row index is the state number, column 1 contains the lower limits, and column 2 contains the upper limits.
- **segs** - Number of samples. For the `Simulation` type, this is the number of samples in each dimension (e.g. `segs=10` will yield 100 samples in a 2D system). For the `MonteCarlo` type, this is the total number of random samples.
- **tspan** - Time span for the simulations. Input as row array where the first element is the start time and the second element is the end time.

4.2.2 Outputs

- Basin of attraction approximations for all stable basins as a struct.
- Visualizations of the basin approximations.

4.3 One-Parameter Example

Code to create an HSA for the 2D dynamical system,

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} -px_2 - x_1^3 \\ -x_2^2 - x_1 \end{bmatrix}, \tag{2}$$

is shown below.

```
F = @(x,u,p) [ -p*x(2)-x(1)^3 ; -x(2)^2-x(1) ];
x0 = [-1;-1];
p0 = -1;
c = cosy(F,2,0,1);
c.p_min = -10;
c.p_max = 10;
c.x_max = [10;10];
c.x_min = [-10;-10];
c.cm.max_steps = 15000;
[B,BP] = c.trace_solution_branches(x0,[],p0);
mode = hsaMD('MD1P_gen',F,B,BP,[],'dot2tex');
```

The resulting HSA is shown in Figure 6.

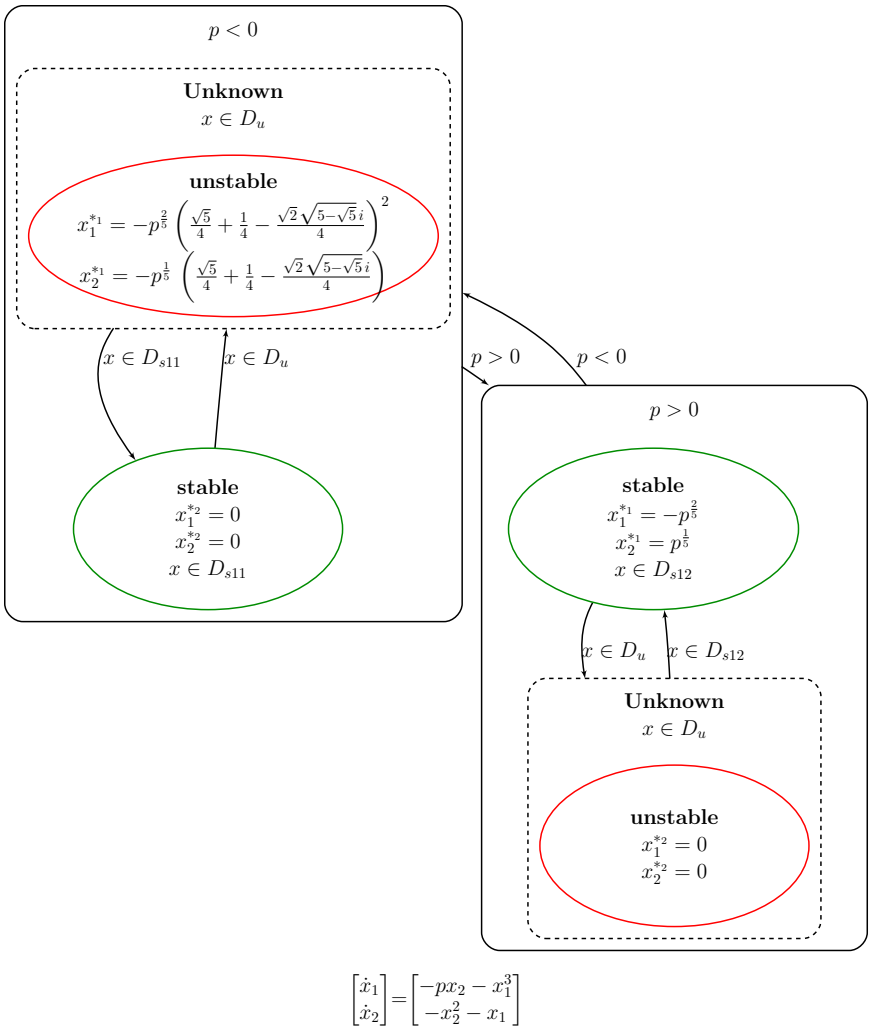

Figure 6: Hybrid stability automaton for a 2D, one-parameter system, general case.

4.4 Multi-Parameter Example

Consider the MDMP system,

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} -p_1 x_2 + p_2 x_1 - x_1^3 \\ -p_3 \left(x_2^2 + x_1 \right) + p_2 x_1 - p_4 x_1^2 \end{bmatrix}, \tag{3}$$

with the parameter designation matrix,

$$PD = \begin{bmatrix} NaN & 0 & 1 & 0 \\ 1 & NaN & 0 & 1 \end{bmatrix}. \tag{4}$$

The following code can be used to create a fixed-p HSA for this system with $p = -0.5$.

```
f = @(x,u,p) [ -p(1)*x(2)+p(2)*x(1)-x(1)^3 ;
          -p(3)*(x(2)^2+x(1))+p(2)*x(1)-p(4)*x(1)^2 ];
PD = [ NaN 0 1 0 ; 1 NaN 0 1 ];
```

```
x0 = [0 0;0 0];
p0 = [-1;-1];
plim = [-10 10;-10 10];
[F,B,BP] = dataMP(f,PD,x0,p0,plim);
opt.PD = PD;
opt.dot2tex = 'on';
mode = hsaMD('MDMP_fp',F,B,BP,-0.5,opt);
```

The resulting HSA is shown in Figure 7.

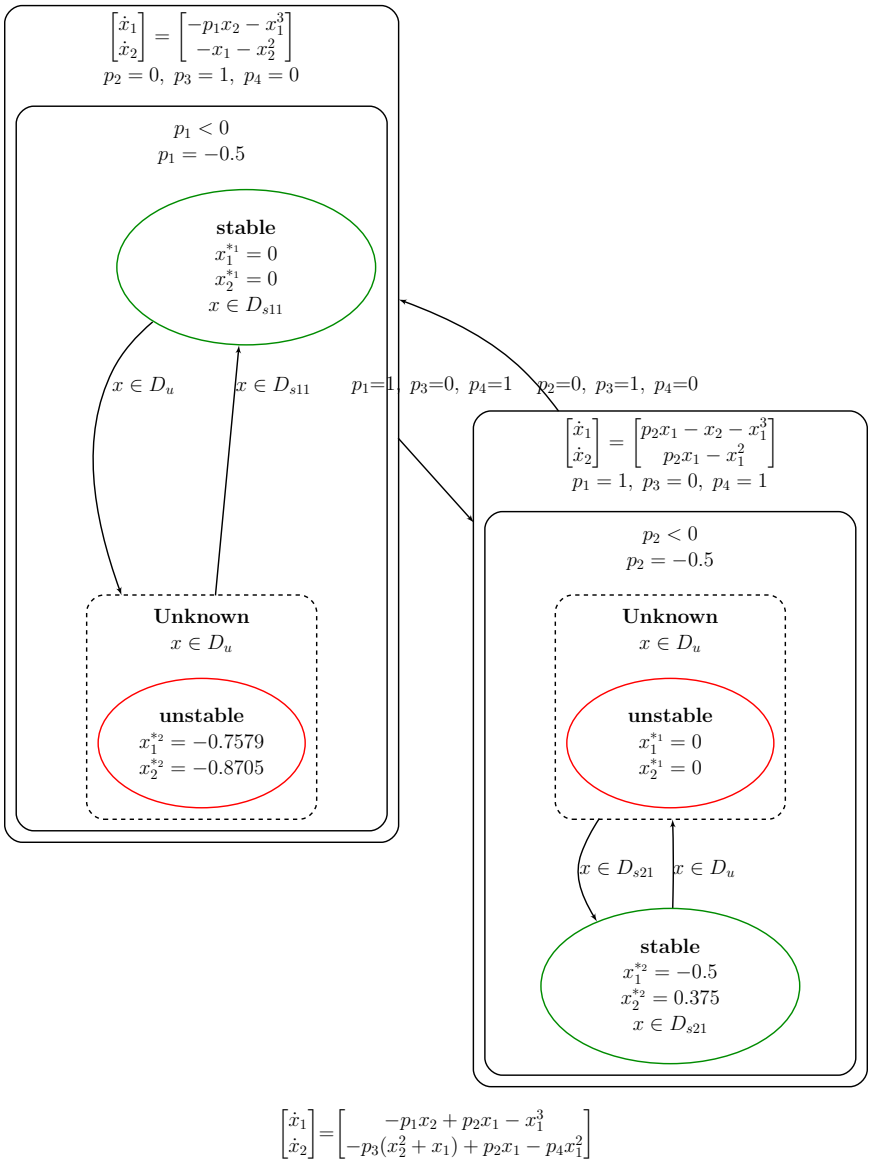

Figure 7: Hybrid stability automaton for a 2D, multi-parameter system, fixed-p case.

The `basinMD` function can be used to obtain approximations for the generically defined D_{s11} and D_{s21} basins of attraction. The following code establishes settings and applies the

`basinMD` function with the `Simulation` method selected.

```
xlimit = [-3 3;-3 3];
tspan = [0 20];
segs = 10;
basin = basinMD(mode,'Simulation',xlimit,segs,tspan);
```

The visualizations for the resulting basin approximations are shown in Figures 8 and 9.

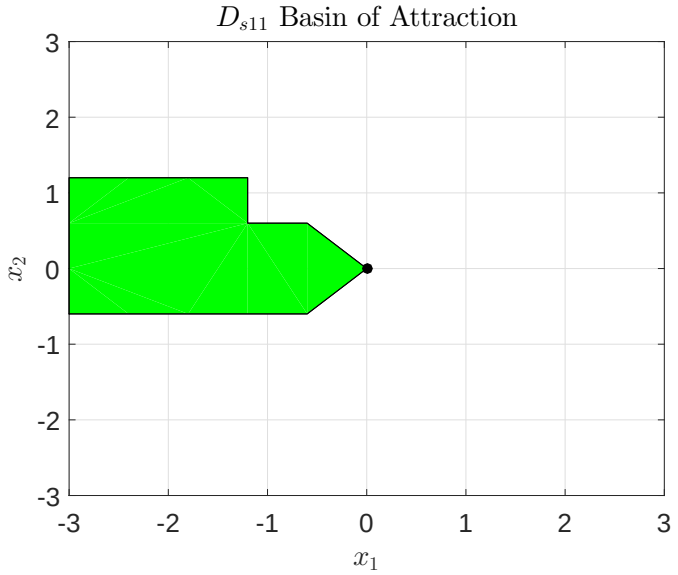

Figure 8: Basin of attraction approximation for D_{s11}.

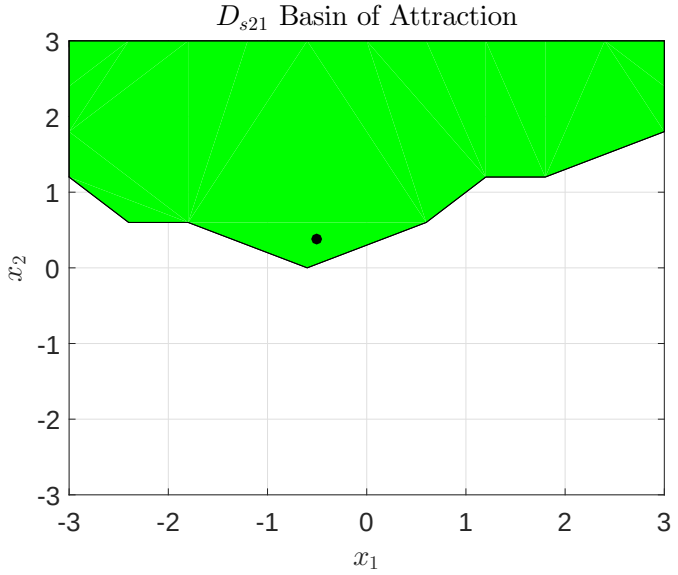

Figure 9: Basin of attraction approximation for D_{s21}.

References

[1] Uth, P., *Stability-Based Hybrid Automata for Safety Verification Using Continuation Methods*, Master's thesis, University of Washington, 2017.

[2] Uth, P., Narang-Siddarth, A., and Clark, M., "Construction of Stability-Based Hybrid Automata for Safety Verification Using Continuation Methods," *AIAA Guidance, Navigation, and Control Conference*, 2018 (in review).

[3] Gansner, E. R. and North, S. C., "An Open Graph Visualization System and its Applications to Software Engineering," *Software Practice and Experience*, Vol. 30, No. 11, 2000, pp. 1203–1233.

[4] Gansner, E., Koutsofios, E., and North, S., "Drawing Graphs with DOT," Technical report, AT&T Research, 2006, [online] http://www.graphviz.org/Documentation/dotguide.pdf.

[5] Ellson, J., Gansner, E. R., Koutsofios, E., North, S. C., and Woodhull, G., "Graphviz and Dynagraph Static and Dynamic Graph Drawing Tools," *Springer Graph Drawing Software*, 2004, pp. 127–148.

[6] Fauske, K. M., "dot2tex - a GraphViz to LaTeX converter," [online] https://dot2tex.readthedocs.io/en/latest/.