User Guide

Norman Hansen

March 9, 2015

1 Plot options

The **plot options** section allows the user to specify the appearance of the desired graphical output. Using the edit fields X-axis and Y-axis, the user can specify which dimensions of the hybrid system should be plotted after the reachability analysis is finished. Figure 47 shows a plot of the dimensions 1 and 2 in a 9-dimensional hybrid system.

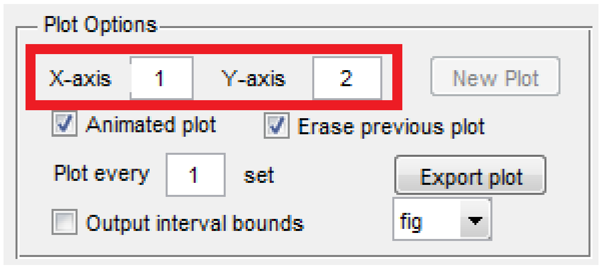
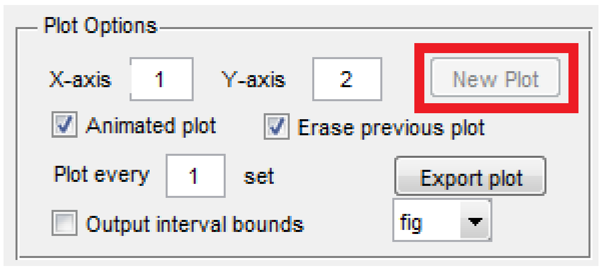


Figure 1: Edit fields to specify the dimensions to plot

If the user wishes to generate plots of other dimensions without restarting the reachability analysis, the **New Plot** button can be used. Therefore, changing the values in the X-axis and Y-axis edit fields to the dimensions for the additional plot and to click the **New Plot** button would suffice. The new plot would appear in a separate figure window.

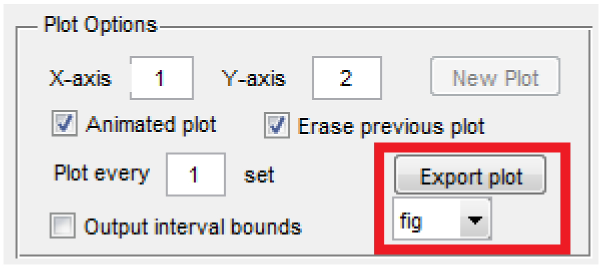
The **Export plot** button can be used to export the plot from the GUI's main window. Firstly, the user has to choose a file format from the dropdown menu. Then, clicking the **Export plot** button will open an input dialog. This dialog allows the user to enter a filename and a sub-path in which the file will be stored. For instance, the input `test` will store the file `test.fig` in the working directory. However, `test/test` will save the file in the folder `test` with name `test.fig`.

The option **Erase previous plot** would delete the plot inside the GUI's main window before the next plot of the newly started reachability analysis is generated. If this option had not been set, the plot of the next reachability



X-axis 1 Y-axis 2
[✓] Animated plot [✓] Erase previous plot
Plot every 1 set
[ ] Output interval bounds
fig

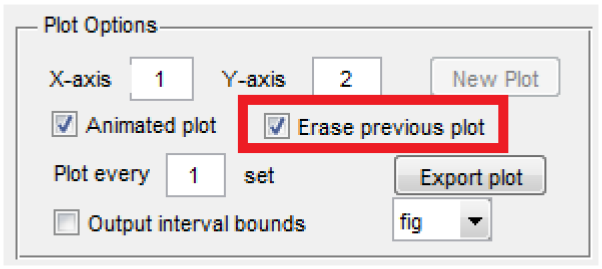
Figure 2: **New plot** button



X-axis 1 Y-axis 2
[✓] Animated plot [✓] Erase previous plot
Plot every 1 set
[ ] Output interval bounds
fig

Figure 3: **Export** button

analysis would be plotted on top of the previous plot remaining in the GUI's main window.



X-axis 1 Y-axis 2
[✓] Animated plot [✓] Erase previous plot
Plot every 1 set
[ ] Output interval bounds
fig

Figure 4: **Erase previous plot** option

The option **Animated plot** will force MATLAB to display each set to be plotted as soon as it has been computed in the figure windows. The advantage of this option is that the user would be able to see the systems in motion and which overlapping sets have been reached first because it was plotted earlier. However, as a drawback, the plots generated are slightly slower.

To increase the plotting speed of computed sets, the edit field Plot every X set allows the user to specify a stepsize to determine which sets should be plotted. For instance, the value 3 will plot only every third set in the flowpipe (e.g. sets 1,4,7, ...). The other sets will be omitted. However, the last computed set is

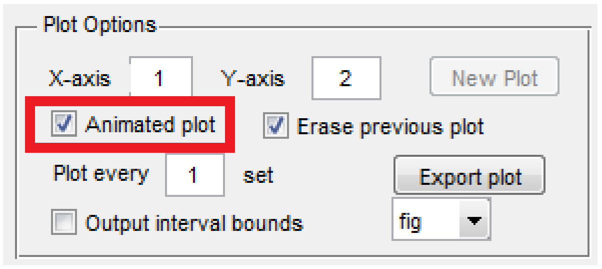

Plot Options
X-axis 1 Y-axis 2
[✓] Animated plot [✓] Erase previous plot
Plot every 1 set
[ ] Output interval bounds
fig

Figure 5: **Animated plot** option

always plotted red, such as sets intersecting a guard or sets in which a fixpoint has been detected[1], which is indicated with orange or turquoise respectively.

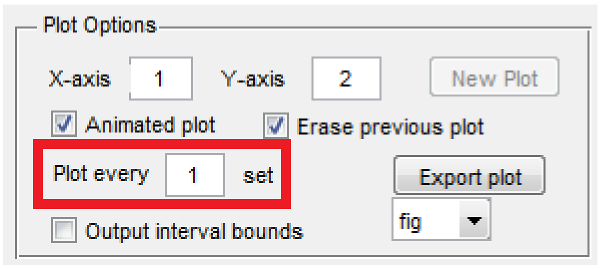

Plot Options
X-axis 1 Y-axis 2
[✓] Animated plot [✓] Erase previous plot
Plot every 1 set
[ ] Output interval bounds
fig

Figure 6: Plot every X set option

Enabling the **Output interval bounds** option will cause the program to generate interval output in the **Interval output** section in the lower right corner.

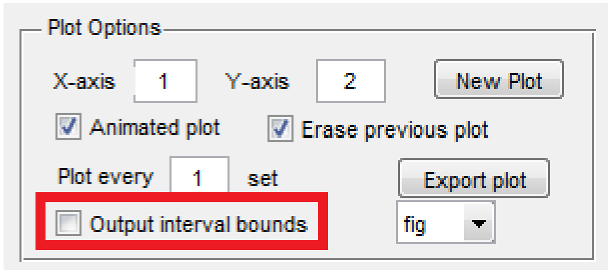
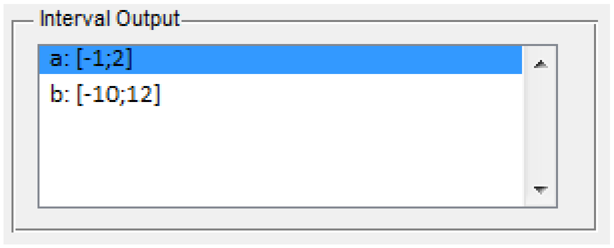

Plot Options
X-axis 1 Y-axis 2
[✓] Animated plot [✓] Erase previous plot
Plot every 1 set
[ ] Output interval bounds
fig

Figure 7: Option to enable interval output

Figure 8 shows an example of such an interval bounds output.

[1]In the event that fixpoints should be detected, the correspondent option in the **Fixpoint detection** section would also have to be enabled. For further informations on fixpoint detection, see 10



a: [-1;2]
b: [-10;12]

Figure 8: Interval output (here for dimensions a and b)

2 Flowpipe construction

The Flowpipe construction section can be used to choose between different scenarios and types for the reachability analysis. The upper dropdown menu allows the user to choose a scenario from the list:

- SpaceEx
- ConstU
- NoScale
- AlgoInv
- AlgoInv2

Each Scenario has its pros and cons. For instance, the **NoScale** scenario has some issues with very overapproximative initial sets. The **ConstU** scenario (from []) however, is generally faster than the others if used with the corresponding initial overapproximation. This assumes too that the input U is piecewise constant between the timesteps. Depending on the choice of the scenario, the accuracy and time elapse of the analysis may change. Moreover, the choice of **AlgoInv** or **AlgoInv2** allows, contrary to the other algorithms, invariants from the hybrid model to be taken into account during the analysis. More about the use and semantics of invariants can be found in section 2.1.

As mentioned previously, the **NoScale** scenario (from [2]) is usually very overapproximative concerning the first computed set. Although overapproximation may be suitable in some cases, we would generally not recommend the use of this scenario because of this drawback.

Choosing **SpaceEx** causes the software to use the algorithm as explained in [1].

When **SpaceEx**, **ConstU**, **AlgoInv** or **AlgoInv2** are chosen, an additional dropdown menu appears on the right side.

This menu allows for the choice of different initial overapproximations. These are **SpaceEx initial overapproximation**[2] and **PreciseOmega0 initial over-**

[2]This overapproximation corresponds to the descriptions of the initial overapproximations from [1]

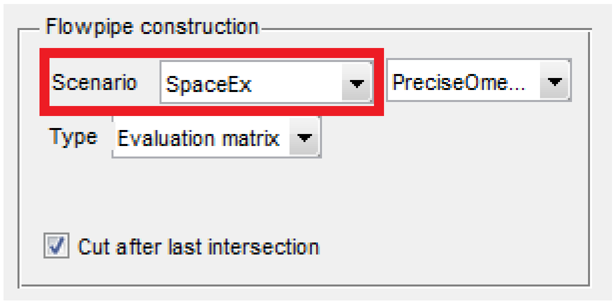


Figure 9: Dropdown menu to chose the flowpipe-construction scenario

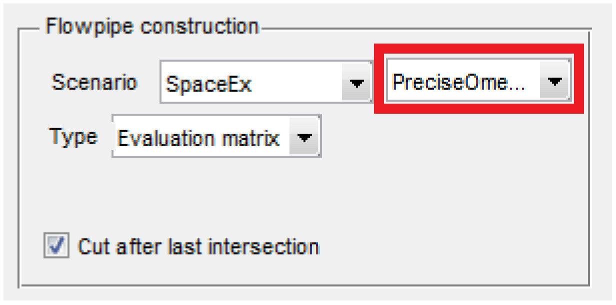


Figure 10: Dropdown menu for the choice of the initial overapproximation

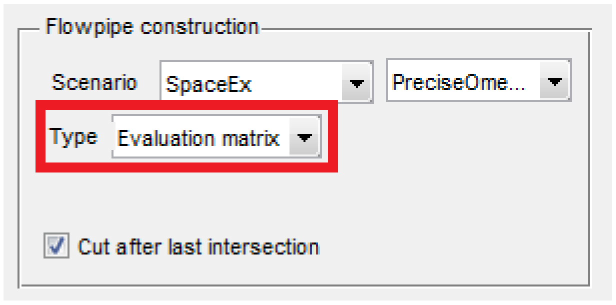
approximation[3] for the scenarios **SpaceEx**, **AlgoInv** and **AlgoInv2**. The **ConstU** scenario defines a third initial overapproximation **ConstU initial overapproximation**[4]. Note that the choice of this initial overapproximation is crucial for the time consumption of the analysis. This results from the evaluation of the support function for this set in each computation step which includes a non-linear optimization.

The lower dropdown menu allows the user to choose the type of flowpipe generation. Possible choices are:

- Evaluation matrix (supported by all scenarios)
- Function representation (supported by SpaceEx and ConstU scenario)

Evaluation matrix will cause the sets to be internally represented using a matrix with the evaluation of all support functions in the specified directions[5]. The choice of **Function representation** results in a very fast flowpipe generation since a chain of function calls, that can be generated very quickly, is used to represent the flowpipe. However, these functions need to be evaluated if plots and cuts, etc. have to be generated. This might take a very long time in cases

[3]The **PreciseOmega0 initial overapproximation** was taken from [3]
[4]**ConstU initial overapproximation** is given in []
[5]Details on the selection of the directions to use are explained in section 7


Flowpipe construction

Scenario: SpaceEx, PreciseOme...

Type: Evaluation matrix

[✓] Cut after last intersection

Figure 11: Dropdown menu to specify the flowpipe construction type

where (multiple) optimizations have to be performed. Moreover, if multiple sets need to be evaluated, some of the calculations for the evaluation are very likely to be performed multiple times, which is very inefficient. Due to these reasons and some implementation related problems with the correctness of the results does the GUI restrict the choice of possible parameters and configurations for the analysis in case the function representation shall be used. The following table gives an overview of our suggestions for the choice of the type for the flowpipe representation:

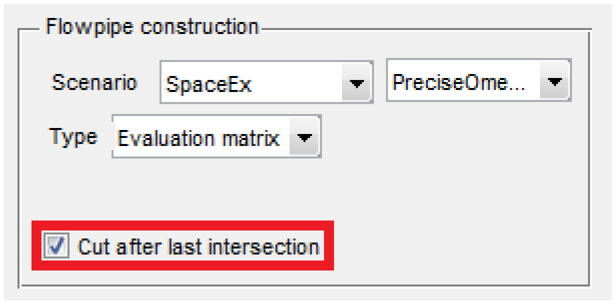
Plots	**Transitions with guards**	**fixpoint detection**	**Flowpipe representation type**
few sets[6]	no	no	function representation
multiple (plots or sets)	*	*	evaluation matrix
*	no	yes	evaluation matrix
*	yes	*	evaluation matrix

The use of **Evaluation matrix** is generally the fastest choice if multiple sets of the flowpipe should be plotted or multiple plots of different dimensions should be generated.

In the case that invariants should be handled, the user has no choice over the use of internal representation since the algorithms **AlgoInv** and **AlgoInv2** work only with the **evaluation matrix** representation. For more information or specifics regarding invariant handling using **AlgoInv** (from [3]) or **AlgoInv2** please read 2.1.

The option **Cut after last intersection** determines whether the flowpipe for the last considered location/mode is cut after a transition has been detected or completely displayed.

Figure 13 shows on the left side the result for a simple example with enabled **Cut after last intersection** option and on the right side the correspondent plot with disabled **Cut after last intersection** option.



Flowpipe construction

Scenario: SpaceEx, PreciseOme...

Type: Evaluation matrix

[✓] Cut after last intersection

Figure 12: Option to cut off the flowpipe after the last detected transition

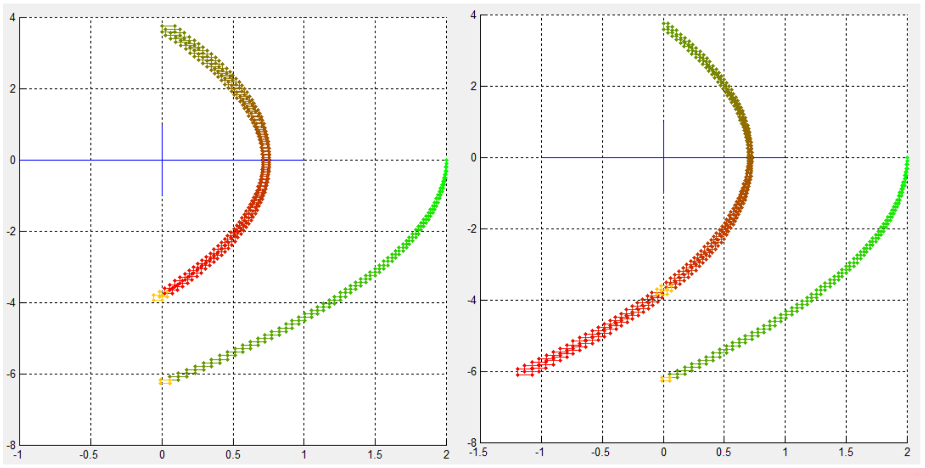


Figure 13: Comparison between flowpipe with enabled **Cut after last intersection** option (left) and disabled option (right)

2.1 Handling invariants

If the model file contains invariants and if **AlgoInv**[7] or **AlgoInv2**[8] is used, they are directly considered. Assuming the use of those algorithms, the reachable states in a location are bounded by the invariant such that no values outside the invariant are part of the computed sets in the flowpipe. The use of invariants with those algorithms do not have any influence on the system's behavior with regards to the transitions and do not necessarily force the system to take a transition.

However, the use of invariants does not affect the decision if spontaneous transitions should be taken. Thus, spontaneous transitions are always taken if their criterion is fulfilled independent of any invariant in the current location/-

[7]taken from [3]

[8]For further explanations see the developer documentation

mode. Nevertheless, the invariant of the destination location/mode has to hold, or else the analysis will terminate at such a point.

If another scenarios such as **SpaceEx**, **ConstU** or **NoScale** are used, invariants contained in the model are completely ignored. Figure 15 shows the difference between an analysis of a model (see figure 14) with invariants using the **SpaceEx** and the **AlgoInv** scenario.

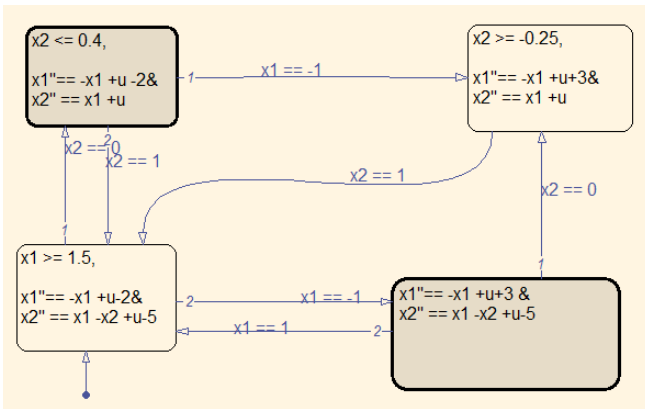


Figure 14: Hybrid automata with invariants

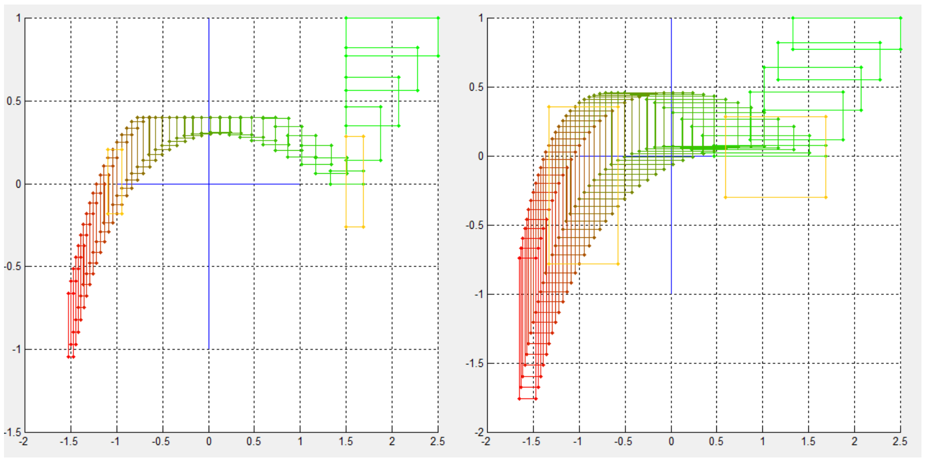
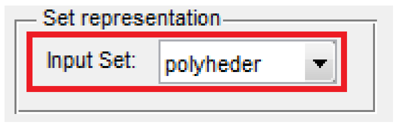


Figure 15: A Hybrid system described in figure 14 using "AlgoInv" (left) and "SpaceEx" scenario (right).

3 Set representation

The set representation section encapsulates the Input Set option. A `.m` model file may define multiple representations for the input set U. For instance, a model file defines the input U as a polytope, a zonotope[9] and a symmetric box. Depending on the representation of U used during the analysis, the total computation time may change. After an input file has been chosen, the dropdown menu containing all defined and supported representations for U appears and the user can choose which of them he wishes to use. It is recommended to use symmetric boxes if U can be represented as symmetric boxes yield a better performance in the analysis (e.g. compared to polytopes, since there is no need to solve linear programs).



Input Set: polyheder

Figure 16: Dropdown menu to specify the representation of the input set

[9]Currently only zonotopes with the origin as center can be used

4 Support function representation

The support function representation section provides the user with the choice of a method to solve linear programs.Linear programs are used to represent the support function of polyhedra. Possible choices are:

- linprog
- fmincon
- cvx
- mpt

Linprog and **fmincon** are build-in matlab functions to solve optimization problems. **CVX** and **MPT** are third-party optimization tools. To make use of **CVX** or **MPT**, the corresponding tool has to be installed on the user's computer. To initialize these tools, the **Initialize additional tools** button can be used.

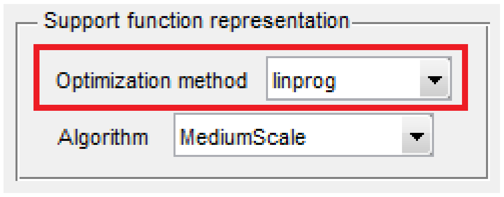


Figure 17: Dropdown menu to choose the optimization method to solve the linear program of a support function

Depending on the user's choice of optimisation methods, different algorithms would then available for use. For **linprog**, the algorithms **MediumScale**, **LargeScale** and **Simplex** can be used. **Fmincon** allows the choice between active-set, interior-pint, trust-region-reflective and **SQP**. The selectable algorithms for **MPT** are **GLPK**, **CDD Criss-Cross**, **CDD Dual Simplex** and **SeDuMi**. **CVX** can only be used with its standard algorithm.

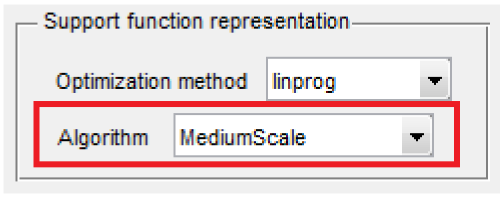


Figure 18: Dropdown menu to choose the optimization algorithm to solve the linear program of a support function

Note that the choice of the algorithm may have a significant influence on the size of overapproximations, the elapsed time and even the ability to evaluate a support function. We recommend to use **MPT** with the **Criss Cross** algorithm in order to achieve the best results concerning the time elapse and urge not to use **CVX** with its standard algorithm (for reasons of time consumption). Moreover, the representation of guards and invariants could be more accurate using **MPT** in cases where other algorithms cause problems[10]. In the event that the chosen optimization method encounters problems during execution, the optimization methods **exitflag** is written with a warning to MATLAB's command window. The respective meanings of the **exitflag** is found in the documentation of the corresponding optimization method. In the case that an **exitflag** states the inability of evaluation of a support function, the computation is aborted and the message on figure 19 is shown to the user.

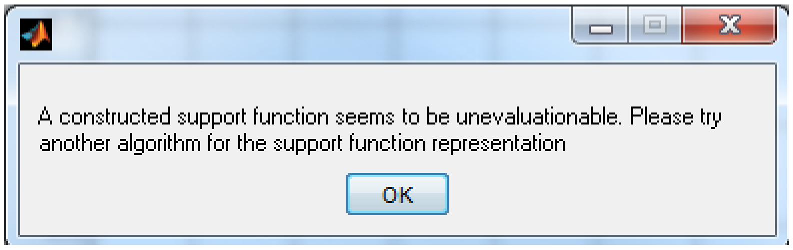

A constructed support function seems to be unevaluationable. Please try another algorithm for the support function representation

Figure 19: Message dialog indicating that the chosen algorithm is not suitable for analyzing the model with the specified parameters

However, even if a warning with an **exitflag** is issued, the analysis might still work fine. For instance, **linprog** issued warnings with exit flag -7 during our tests indicating that a search direction had became too small and no further progress could be made. For this particular **exitflag** the analysis continued without displaying a message dialog. Nevertheless, if such warnings were to occur during an analysis, the computation should be re-executed using another optimization method or algorithm, especially if the result do not appear as expected.

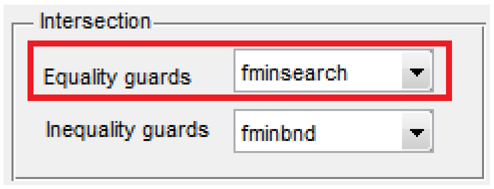
[10]For further details on this issue, see the developers documentation in the section Infinity-problem

5 Intersection

This section elaborates on the calculations of intersection sets with guards. To compute the set resulting from the intersection of a set with an equality guard, an optimization has to be performed unless the **fast_intersection** approach was used. The upper dropdown menu provides the user with different optimization methods he may use to perform this task. Possible choices are:

- fminsearch
- fminunc
- dichotomicSearch
- RayAlgo
- fast_intersection

Fminsearch and **fminunc** are build-in MATLAB functions. The dichotomic search method is implemented as described in [3]. The minimization approach referred to as **RayAlgo** is implemented as described in [4]. The **fast_intersection** method overapproximates the intersection as the minimum of two support functions. As the name indicates this method is very fast because no optimization is performed. The drawback using the **fast_intersection** is the need to re-evaluate intersected sets which yields in additional computation overhead. For further informations regarding this issue read the developers documentation.



Equality guards: fminsearch

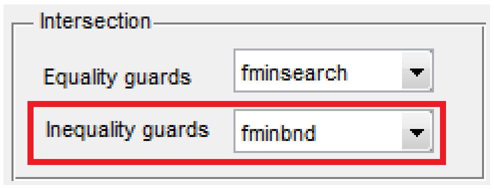
Inequality guards: fminbnd

Figure 20: Dropdown menu to chose the method used to calculate the intersection set between sets from the flowpipe and equality guards

The lower dropdown menu can be used to chose an optimization method which shall be used if the intersection of a set with an inequality guard has to be calculated. Possible choices for this task are:

- fminbnd
- fmincon
- RayAlgo
- fast_intersection

The methods **fminbnd** and **fmincon** are build-in MATLAB functions. The **RayAlgo** is the same for equality guards. The only difference is the search space considered for the minimization of a convex function — only positive real values for the inequality guards. The **fast_intersection** method is also the same for equality guards with the difference, that only one minimum is computed for inequality guard intersections.


Intersection

Equality guards: fminsearch

Inequality guards: fminbnd

Figure 21: Dropdown menu to choose the method to be used to calculate the intersection set between sets from the flowpipe and inequality guards

6 Clustering

The **Clustering** section allows the user to chose how to proceed when multiple sets intersect with the same guard. If the **precluster** option is enabled, all sets which intersect with the guard are clustered using the convex hull and the intersection is then calculated. Figure 23(a) shows that only the intersection of one set (in yellow) — the convex hull of two sets with the guard $y == 0$ — is considered. If this option is disabled, the intersection of each set with the guard is calculated and the resulting sets are clustered using the convex hull method after the intersection calculations. The latter approach results in longer computation time if multiple sets were to intersect with a guard because multiple intersections (instead of one) would have to be calculated. Figure 23(b) illustrates the case where 2 sets (yellow) intersect the guard $y == 0$.

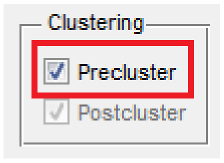


Figure 22: Checkbox to enable pre-clustering

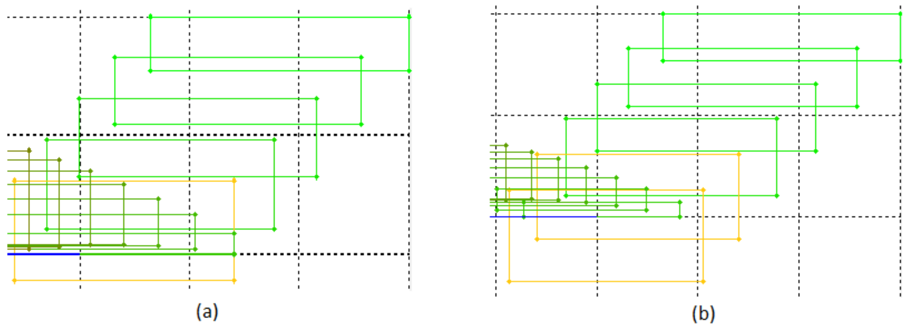


Figure 23: Result with enabled (a) and with disabled (b) precluster option

7 Directions

The **Directions** section allows the user to specify how many directions are to be used to overapproximate the convex sets in the analysis. The **box** option will generate two directions along the axes of every dimension of the hybrid system. Figure 47 shows a plot using the box option.

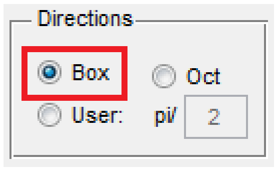


Figure 24: **box** option

The **oct** option will generate eight directions for every pair of dimensions such that the directions in a plane formed by two dimensions are equally distributed.Figure 48 shows the same plot as figure 47 but using the **oct** option instead of the **box** option.

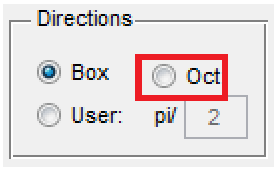


Figure 25: **oct** option

The **user** option enables the user to specify an arbitrary angle between 0 and $\frac{\pi}{2}$, by entering a divisor of π, which should be used for the rotation to generate directions in each pair of dimensions. Figure 49 shows the already above mentioned plot using $\frac{\pi}{6}$ as value for the **user** option.

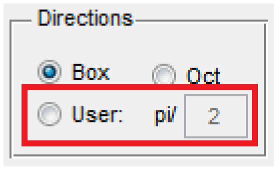


Figure 26: **user** defined directions

Generally, more directions lead to higher accuracy in the representation of the convex sets. However, more directions lead to much longer computation time. The user should consider that for higher dimensional system (i.e. a 9 dimensional system) the use of the **box** option generates in total fewer directions (18) than the use of the **oct** option (162).

8 General options

The general options section allows the user to configure some important general options. Before any reachability analysis can be started, the user need to provide an input file by clicking the **Locate Input File** button. Clicking this button will open a dialog which can be used to pick an input file.

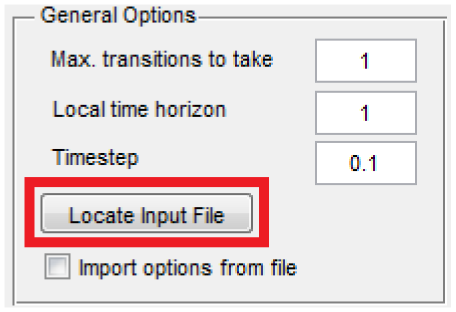


Figure 27: Button to select a model file

After a `.m` input file has been chosen, the user can decide if the values for the Local time horizon and the Timestep shall be imported from the input file or if the values provided by the GUI shall be used. This choice is done by enabling the **Import options from file** checkbox.

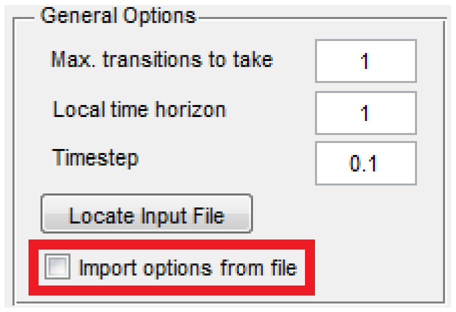
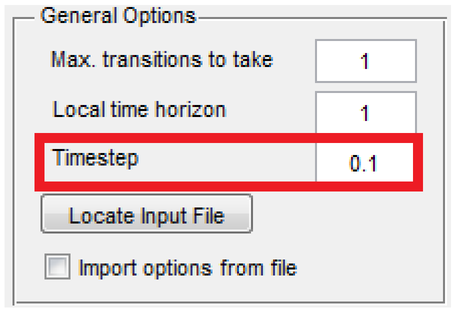


Figure 28: Option to import different parameters from model file

Alternatively, Simulink Stateflow$^{®}$ models can also be used as input files. Chapter 19.2 provides further explanation on this input file format.

If the **Import options from file** checkbox is not enabled, the user can determine the timestep which should be used during the reachability analysis by entering a double value in the Timestep field. A small timestep will result in more accurate results but cause longer computation time. Certain Timestep values might lead to unrecognized intersections with guards (especially if the **Evaluation matrix** representation has been chosen).

The Local time horizon field allows the user to enter a time horizon for the



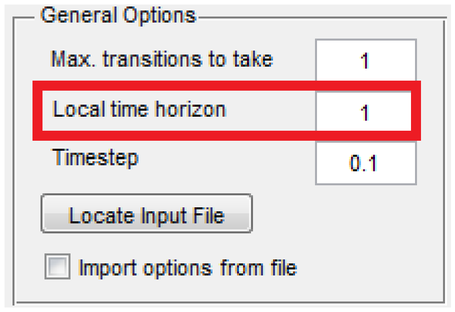
Max. transitions to take 1

Local time horizon 1

Timestep 0.1

Figure 29: Edit field to specify the size of the timestep to use

flowpipe in each location/mode of the model. The construction of the flowpipe in a location/mode will end when the local time horizon is reached. Due to implementation decisions, the flowpipe for each location/mode is calculated to this time horizon. Therefore, the user should choose a value (a double) as small as possible but as big as required.



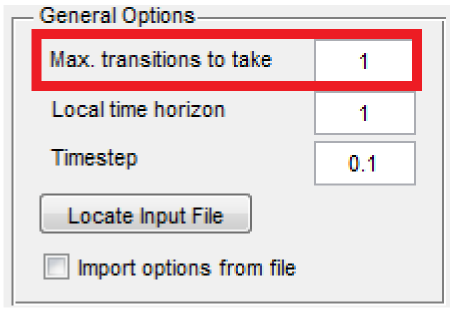
Max. transitions to take 1

Local time horizon 1

Timestep 0.1

Figure 30: Edit field to specify the time horizon

The value in the Max. transitions to take field determines how many discrete steps (transitions) can be performed at maximum during a reachability analysis. A value of 3 will limit the reachability analysis to the computation of three flowpipes (each for one visited location/mode), meaning that at most two taken transitions are considered. Depending on the model, increasing this value can affect the computation time if guard intersections and resets need to be computed.


General Options

Max. transitions to take: 1

Local time horizon: 1

Timestep: 0.1

Figure 31: Specify the maximal number of locations/modes to consider during a single path of the computation

9 Spontaneous transitions

9.1 Semantics

A spontaneous transition is a transition to be taken in the event a certain criterion is met. Compared to usual transitions, this is not expressed by a guard which has to be satisfied in order to take the transition but can be seen as an event that triggers the spontaneous transition. We consider two different types of such events, namely the detection of a fixpoint or the elapse of a certain time.

For fixpoint triggered transitions the characteristics are very simple. If fixpoint detection is enabled and a fixpoint has been detected with the specified tolerance, the transition is taken. The only problem that the user has to keep in mind is that fixpoints can only be detected within a single location. That means that the search for a fixpoint relates always only to the part of the flowpipe generated by the current active location. For instance, if the set $a = [-5; 10]$ was reachable even before a transition was taken and the subset $b = [2; 3]$ is computed, no fixpoint would have been detected although there exists one. In case that b was computed before the transition was taken, the fixpoint would be detected.

For a time triggered spontaneous transition, the absolute point in time, in which the transition should be taken, has to be modelled. If a transition should be taken at time t, it would only be possible to take the transition within the interval $[t; t + timestep]$. Thus, if $t = 1$ and $timestep = 0.7$ the transition could be taken at an elapsed time of $t_{elapsed} = 1.4$. The resulting flowpipe may be completely different if another timestep is used, e.g. 0.1, for which the computed set at the point of reaching time 1 is always smaller and more accurate.

It may occur that a model contains a location with both, guarded and spontaneous transitions. Which transition has to be taken will be decided depending on which criterion is fulfilled first. If the guard is satisfied first the correspondent transition is taken. In case a fixpoint is detected before the guard is satisfied and the specified time for the time triggered transition is reached, the fixpoint triggered transition will be taken. In case the specified time t is reached before satisfying the guard and without detecting a fixpoint hitherto t, the time triggered transition is taken.

10 Fixpoint detection

The Fixpoint detection section enables the user to influence the used fixpoint detection algorithm. First, the fixpoint detection has to be enabled using the correspondent checkbox.

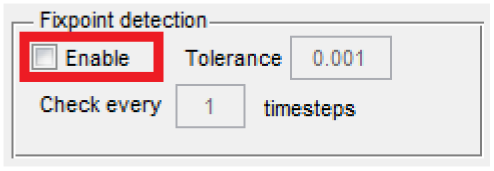


Figure 32: Option to enable the fixpoint detection

The analysis detects a fixpoint if $\Omega_i(l) - \Omega_{i+k}(l) < tolerance$ for all directions l within a location/mode, $\Omega(l)$ being the support function of a reachable set. The Tolerance edit field can be used to define a value describing the maximal gap between two values which are considered equal for the fixpoint detection. A message-box informs the user additionally to the text output in the command window in case a fixpoint is found. Fixpoints are colored turquoise when plotted.

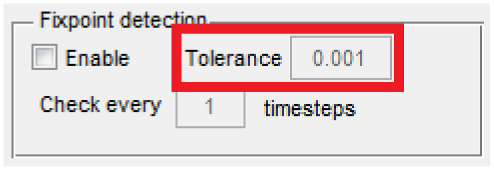


Figure 33: Edit field to specify the recognition tolerance

The value in the lowest edit field determines the value k, defining the stepsize for fixpoint checks. A value of 10 would imply that only every 10th set in the computed flowpipe is compared to the 10th following set.

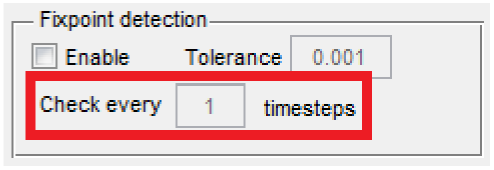


Figure 34:

To enable fixpoint triggered transitions during the reachability analysis, it is required that the user enables the fixpoint detection as described. In case a model contains a fixpoint triggered transition and the fixpoint detection is disabled, a message box informs the user about this flaw. The fixpoint detection is disabled by default because it requires additional computation time.

11 Multithreading

Using the **Enable** checkbox the user can enable the use of multiple threads. Enabling the checkbox will simply call the matlab function **matlabpool local**. If this feature is already in use, the configuration possibilities using the GUI might not work as expected.

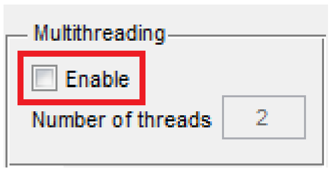


Figure 35: Option to enable multithreading

Using the **Number of threads** edit field, the number of parallel threads can be chosen.

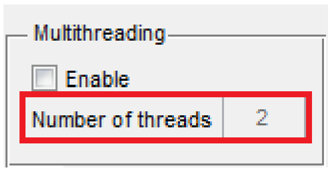


Figure 36: Edit field to specify the number of threads to use

The multithreading feature affects the computation of the evaluation results for all but the **AlgoInv2** scenario directly, which means that the evaluation results for multiple directions are computed in parallel. Furthermore, the evaluation of a support function represented by a function handle will be computed for multiple directions in parallel. This affects, for instance, the reconstruction operations in **AlgoInv2**, the reset computation or the computation of set intersections.

12 Initial values

The two edit fields in this category become active when a Simulink Stateflow model (`.mdl` file) is loaded. The left field can be used to define the initial set to use for the reachability analysis. The right field contains the definition of the input set, previously referred to as U. Using `.mdl` files as input, it is not possible to define input sets U as balls, boxes or zonotopes, but only as polyhedra. The constraints for the initial polyhedra are given as a linear term without multiplication signs or brackets (e.g. $x + 3y - 2.21var3 + 0.5x$) followed by a comparator operator (<=, >=, ==) and a constant. It is also possible to specify constraints of the form $1 <= var3 <= 3.21$.

Multiple constraints are separated by &. The variables used in the left edit field have to be consistent with the variable names used in the `.mdl` model and the variable names used in the right edit field have to be consistent with the names of the variables of the input set in the model. The variable identifiers can be seen on the right side and are displayed in two lists.

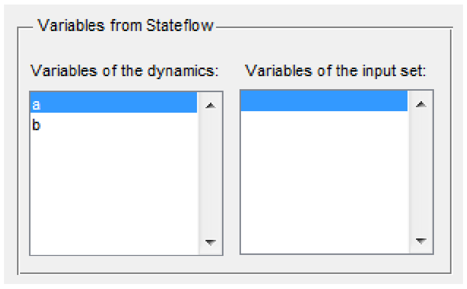


Figure 37: Variables from stateflow section

Moreover, the user has to ensure, that both specified sets are closed in all dimensions.

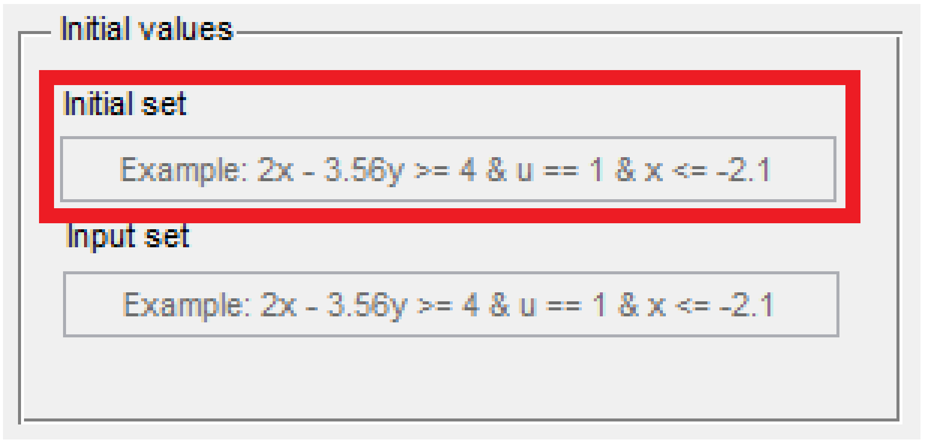

Initial values
Initial set
Example: 2x - 3.56y >= 4 & u == 1 & x <= -2.1
Input set
Example: 2x - 3.56y >= 4 & u == 1 & x <= -2.1

Figure 38: Edit field containing the constraints for the initial set

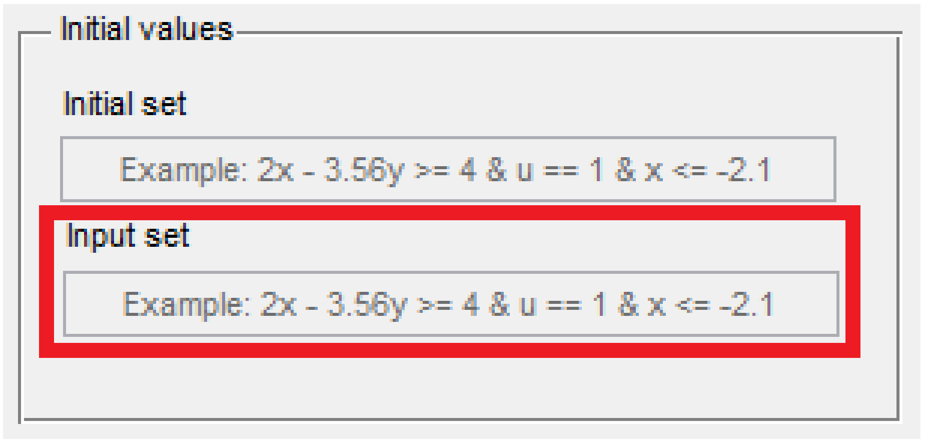
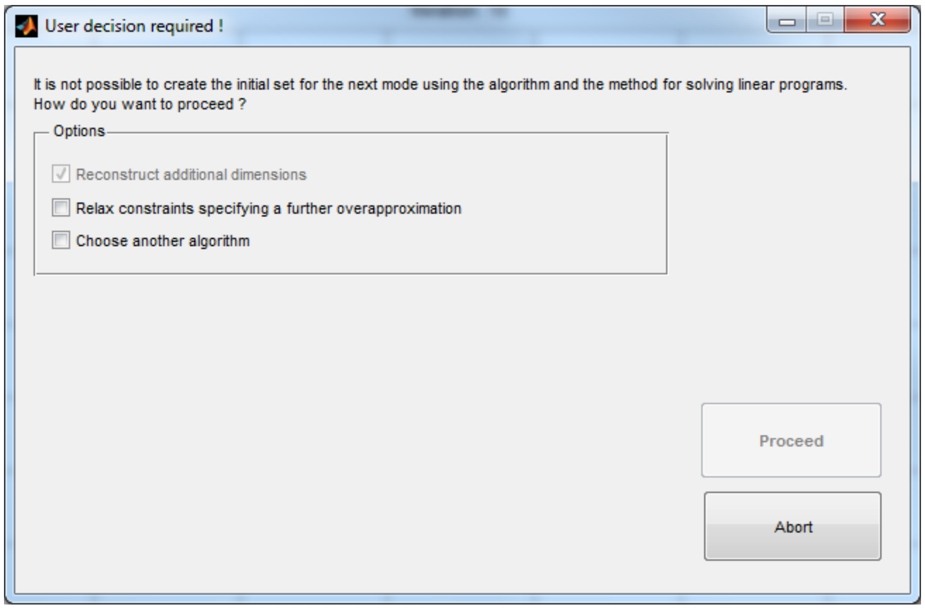

Initial values
Initial set
Example: 2x - 3.56y >= 4 & u == 1 & x <= -2.1
Input set
Example: 2x - 3.56y >= 4 & u == 1 & x <= -2.1

Figure 39: Edit field containing the constraints for the input set U

13 Troubleshooting in case of numerical problems

During the analysis of hybrid systems, the following dialog shown in figure 40 might appear.

In this case, the program might not be able to reconstruct a function handle representation based on the current evaluation results of a support function due to numerical inaccuracies. This problem can sometimes be solved by simply choosing a different optimization algorithm for the representation. By clicking the **Choose another algorithm** checkbox, the user has the option to manually choose or let the program automatically choose a different algorithm.



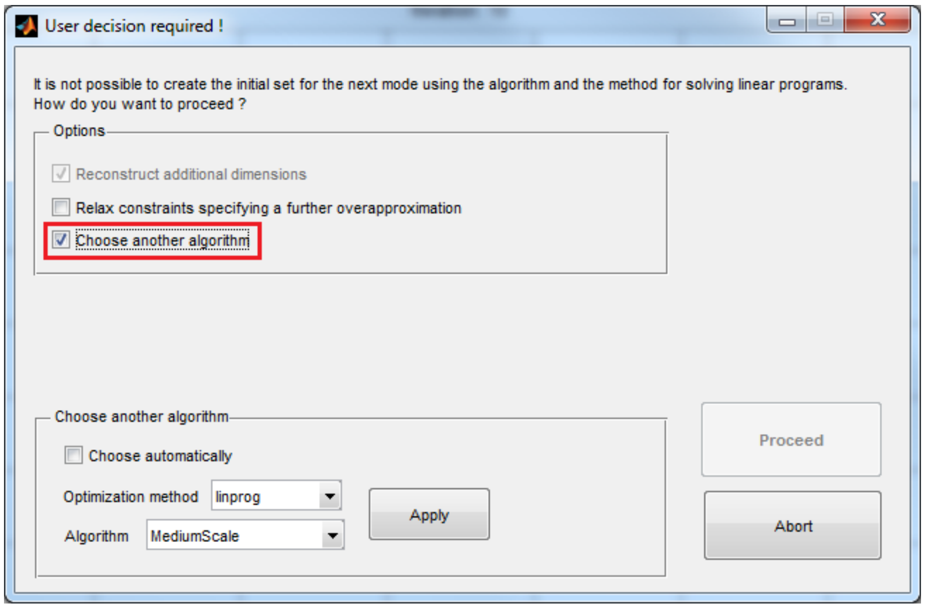
User decision required !

It is not possible to create the initial set for the next mode using the algorithm and the method for solving linear programs.
How do you want to proceed ?

Options

- [✓] Reconstruct additional dimensions
- [ ] Relax constraints specifying a further overapproximation
- [ ] Choose another algorithm

Figure 40: Dialog indicating numerical problems during evaluation of support functions



User decision required !

It is not possible to create the initial set for the next mode using the algorithm and the method for solving linear programs.
How do you want to proceed ?

Options

- [✓] Reconstruct additional dimensions
- [ ] Relax constraints specifying a further overapproximation
- [✓] Choose another algorithm

Choose another algorithm

- [ ] Choose automatically

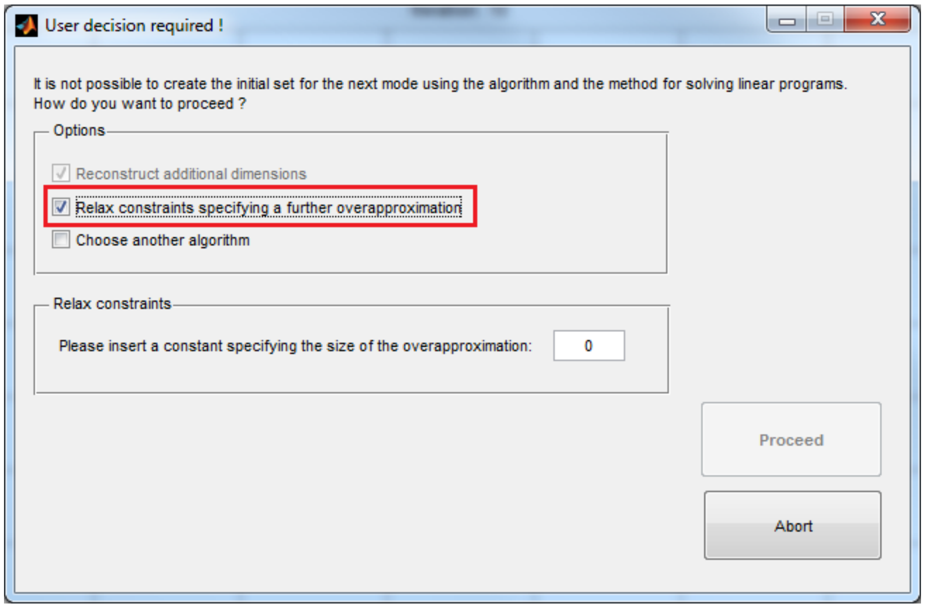
Optimization method: linprog

Algorithm: MediumScale

Figure 41: This checkbox enables the possibility to chose a different algorithm

In the latter case, all algorithms are tested and the first working algorithm is used. To choose an algorithm manually, the user can use the two dropdown menus. With a click on the **Apply** button the choice will be tested. Once a working algorithm has been chosen, the **Proceed** button will cause the reachability analysis to continue using the specified algorithm as optimization method for support function representation.

Choosing a different algorithm might not always work. For such cases, only a further overapproximation can be applied to resolve the problem. This can be done by enabling the **Relax constraints** checkbox.


User decision required !

It is not possible to create the initial set for the next mode using the algorithm and the method for solving linear programs.
How do you want to proceed ?

Options
- [✓] Reconstruct additional dimensions
- [✓] Relax constraints specifying a further overapproximation
- [ ] Choose another algorithm

Relax constraints
Please insert a constant specifying the size of the overapproximation: 0

Figure 42: This checkbox enables the possibility to relax the constraints of the support function's linear program

The edit field in the **Relax constraints** section allows the user to enter a value which will be added to the support function evaluation result in every direction. Thus, the value should be kept as small as possible to minimize the overapproximation.

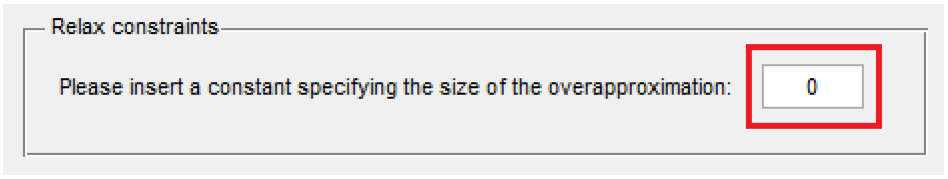
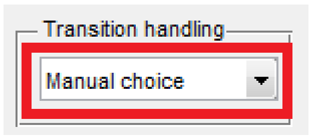

Relax constraints
Please insert a constant specifying the size of the overapproximation: 0

Figure 43:

After typing in the value, pressing enter will start a check if the entered value solves the problem. In case it does, the **Proceed** button will be enabled.

It is also possible to make use of both approaches to continue the computation. If a value has been entered to increase the overapproximation and the automatic choice of an algorithm is made, the test for a suitable algorithm will be made using the value for further overapproximation. In the same way the manual choice of a different algorithm will consider the value for further overapproximation.

14 Transition handling



Transition handling

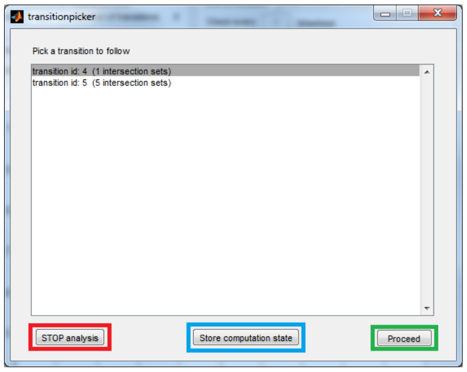
Manual choice

Figure 44: Transition handling section

The dropdown field in this section allows to define which of multiple possible transitions shall be taken. The possible options are:

- Manual choice
- Most intersections
- First detected
- Last detected
- All

Manual choice opens every time a choice can be made a dialog which allows the user to pick one of the detected transitions. Figure 45 shows this dialog.



Pick a transition to follow

```
transition id: 4  (1 intersection sets)
transition id: 5  (5 intersection sets)
```

Figure 45: Dialog for chosing the transition to proceed with

The **Stop analysis** Button (marked in red) causes the analysis to stop and the GUI to plot the result of the analysis up to the computed point. In contrary to the **Stop analysis** button, the **Proceed** button will cause the analysis to proceed taking the transition picked from the list of transitions. In the case where multiple transitions are considered, or it is not clear which transitions should be taken, it is possible to use the **Store computation state** button (marked in blue). After the computation state has been stored, e.g. the second transition can be picked. In the case where the first transition should also be considered, it is possible to load the stored computation state from the **Stored computation steps** section (see 15) which will result in displaying the same dialog to pick a transition again and choose the first transition to proceed. The plots of both analysis results will overlap in the resulting figure.

As an alternative to **Manual choice**, a strategy on how to pick the transition to proceed with can be chosen in the case where the program should run autonomously without further user interaction. As indicated by their names, the option **First detected** will automatically choose the transition that can be taken at the earliest possible time and **Last detected** correspondingly at the latest possible transition. The option **All** will consider all transition paths up to the specified depth in the **General options** section. A further options is **Most intersections**, which chooses the transition with the most intersecting sets from the computed flowpipe.

If there are to be multiple candidate transitions, with which the the computation may proceed, the transition with the highest priority is chosen.

15 Stored computation steps

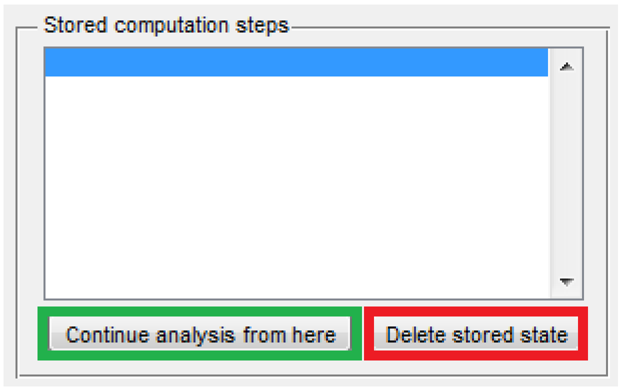


Figure 46: Stored computation states section

This sections allows the user to continue an analysis from a previously stored computation state. To load a stored state from the list, it has to be selected and the **Continue analysis from here** button (green) needs to be clicked. If a stored state is no longer needed, it can be deleted by selecting the state from the list and clicking the **Delete stored state button** (red). Please note, that all stored computation states are deleted in case a new analysis is started using the **Start** Button in the lower left corner of the GUI main window.

16 Interval output

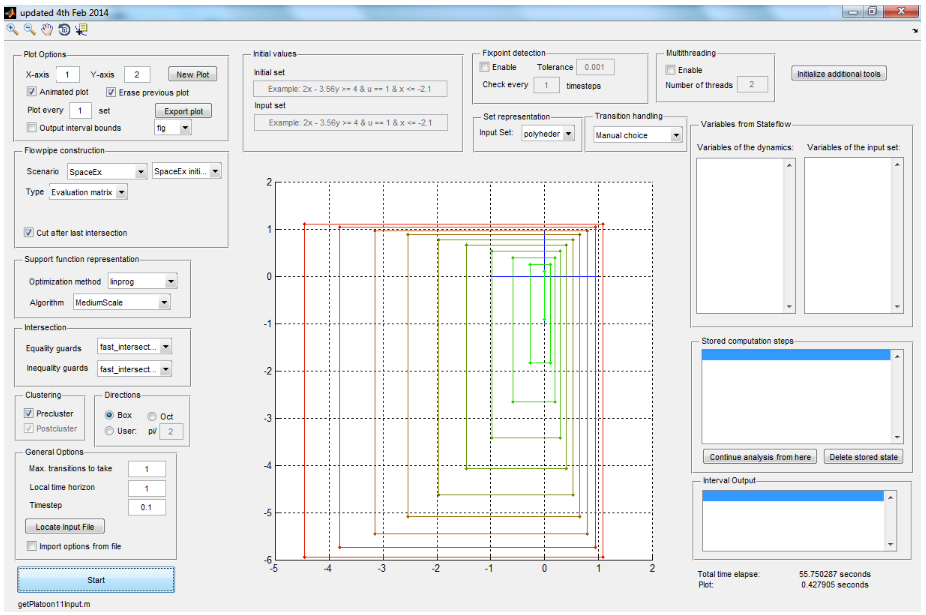
This section displays the interval bounds for each dimension in the case where the **Output interval bounds** option has been enabled. Enabling this option may increase the time for the analysis where directions are badly chosen. (There are dimensions without directions along the dimension axes.)

17 Further displayed information

Under the **Start** button is the name of the currently loaded model file displayed which will be analyzed when the start button is pressed. The time consumption **Total time elapse** and the time used to generate the plot **Plot** for the analysis displayed after it has been terminated are shown in the lower right corner.

Further information is displayed in *MATLAB*'s command window. These are, for instance, the time consumption to generate the flowpipe for single modes, log outputs of the steps of the analysis that are currently being executed or have already been completed, notifications and warnings indicating possible problems during the analysis and critical error messages.

18 Additional Screenshots


updated 4th Feb 2014

Plot Options
X-axis 1 Y-axis 2
[✓] Animated plot [✓] Erase previous plot
Plot every 1 set
[ ] Output interval bounds fig

Flowpipe construction
Scenario SpaceEx SpaceEx initi...
Type Evaluation matrix
[✓] Cut after last intersection

Support function representation
Optimization method linprog
Algorithm MediumScale

Intersection
Equality guards fast_intersect...
Inequality guards fast_intersect...

Clustering
[✓] Precluster
[✓] Postcluster

Directions
[✓ Box] Oct
User: pi/ 2

General Options
Max. transitions to take 1
Local time horizon 1
Timestep 0.1
[ ] Import options from file

getPlatoon11Input.m

Initial values
Initial set
Example: 2x - 3.56y >= 4 & u == 1 & x <= -2.1
Input set
Example: 2x - 3.56y >= 4 & u == 1 & x <= -2.1

Fixpoint detection
[ ] Enable Tolerance 0.001
Check every 1 timesteps

Multithreading
[ ] Enable
Number of threads 2

Set representation
Input Set: polyheder

Transition handling
Manual choice

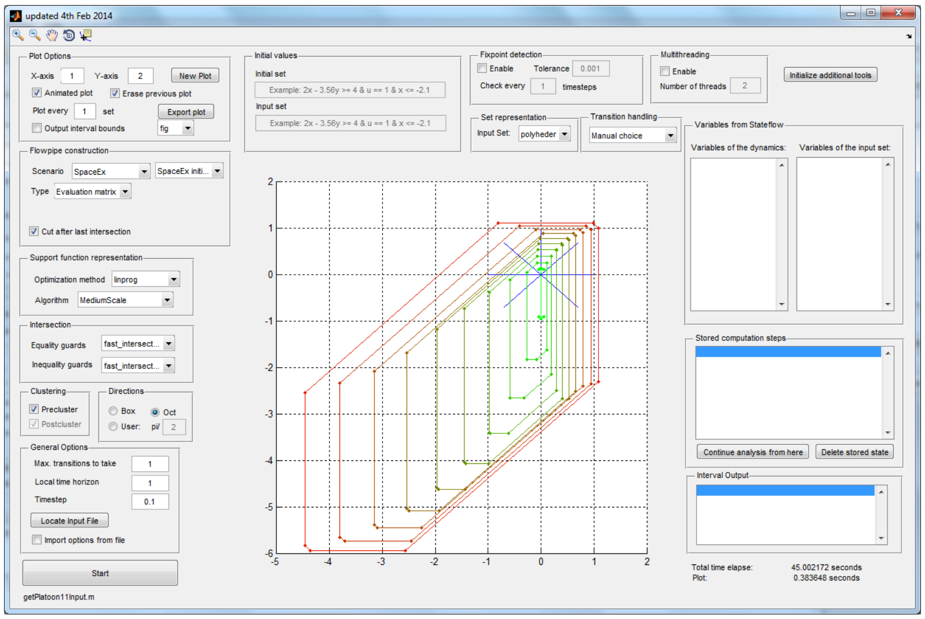
Variables from Stateflow
Variables of the dynamics:
Variables of the input set:

Stored computation steps

Interval Output

Total time elapse: 55.750287 seconds
Plot: 0.427905 seconds

Figure 47: Plot of dimensions 1 and 2 of a 9-dimesional system


updated 4th Feb 2014

Plot Options: X-axis 1, Y-axis 2; [✓] Animated plot; [✓] Erase previous plot; Plot every 1 set; [ ] Output interval bounds; fig

Flowpipe construction: Scenario SpaceEx, SpaceEx initi...; Type Evaluation matrix; [✓] Cut after last intersection

Support function representation: Optimization method linprog; Algorithm MediumScale

Intersection: Equality guards fast_intersect...; Inequality guards fast_intersect...

Clustering: [✓] Precluster; [✓] Postcluster

Directions: Box [✓ Oct] User: pi/ 2

General Options: Max. transitions to take 1; Local time horizon 1; Timestep 0.1; [ ] Import options from file

getPlatoon11Input.m

Initial values: Initial set: Example: 2x - 3.56y >= 4 & u == 1 & x <= -2.1; Input set: Example: 2x - 3.56y >= 4 & u == 1 & x <= -2.1

Fixpoint detection: [ ] Enable; Tolerance 0.001; Check every 1 timesteps

Multithreading: [ ] Enable; Number of threads 2

Set representation: Input Set: polyheder

Transition handling: Manual choice

Variables from Stateflow: Variables of the dynamics: Variables of the input set:

Stored computation steps

Interval Output

Total time elapse: 45.002172 seconds
Plot: 0.383648 seconds

Figure 48: Same output as in figure 47 but with use of the **oct**-option

19 Input file format

19.1 Matlab Functions as Input Files

The basic input file format is a simple text file defining a MATLAB function without input parameters. The output parameters of the function define the hybrid system and some options.

We explain the file format using a bouncing ball example. The following line defines the function name **getBouncingBallInput** which has to be the file name too.

```
[states ,q,qa, transitions ,I ,U, configuration] = getBouncingBallInput
```

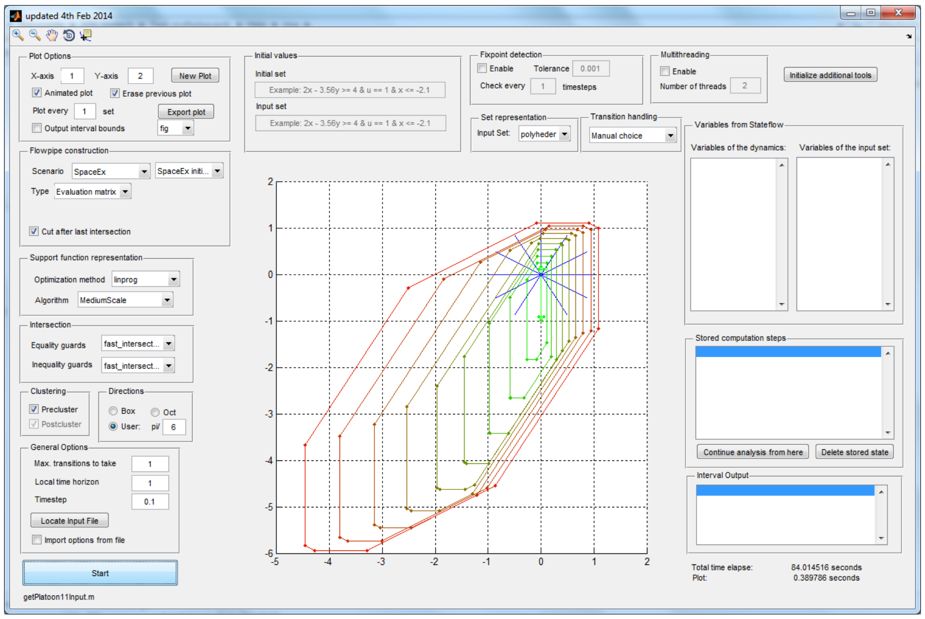
Output parameter `q` defines the locations/modes of the hybrid system. In our example the system consists of one location/mode with identifier 1.

```
q = [ 1 ];
```

The initial location/mode is defined setting the value of `qa` to the start locations/modes identifier.

```
qa = 1;
```

The properties of a location are modelled using the `states` cell array. Each location/mode has an index defined through `q` and a correspondent entry in the `states` cell array containing a structure defining the properties.


updated 4th Feb 2014

Plot Options
X-axis 1 Y-axis 2
[✓] Animated plot [✓] Erase previous plot
Plot every 1 set
[ ] Output interval bounds fig

Flowpipe construction
Scenario SpaceEx SpaceEx initi...
Type Evaluation matrix
[✓] Cut after last intersection

Support function representation
Optimization method linprog
Algorithm MediumScale

Intersection
Equality guards fast_intersect...
Inequality guards fast_intersect...

Clustering
[✓] Precluster
[✓] Postcluster

Directions
( ) Box ( ) Oct
(●) User: pi/ 6

General Options
Max. transitions to take 1
Local time horizon 1
Timestep 0.1
[ ] Import options from file

getPlatoon11Input.m

Initial values
Initial set
Example: 2x - 3.56y >= 4 & u == 1 & x <= -2.1
Input set
Example: 2x - 3.56y >= 4 & u == 1 & x <= -2.1

Fixpoint detection
[ ] Enable Tolerance 0.001
Check every 1 timesteps

Multithreading
[ ] Enable
Number of threads 2

Set representation
Input Set: polyheder

Transition handling
Manual choice

Variables from Stateflow
Variables of the dynamics:
Variables of the input set:

Stored computation steps

Interval Output

Total time elapse: 84.014516 seconds
Plot: 0.389786 seconds

Figure 49: Same output as in figure 47 but with use of the **user**-option with $\frac{\pi}{6}$

The continuous dynamics of systems described in the input file can be of the form $\dot{x} = Ax + b + BU$, U being a convex input set. Next, we define these continuous dynamics for each location/mode, setting the values for the correspondent matrix A to

```
states{1}.A = [0,1;0,0];
```

The matrix A for state 1 implies that the system is 2 dimensional. This has to be consistent with the dimensions in all other locations/modes.

We set the values for the constant part of the flow description to

```
states{1}.b = [0; -9.81];
```

Similar to A we define B describing a 2 dimensional free input.

```
states{1}.B = [1  0;0  1];
```

The matrix B has to be consistent with the matrices B from all other locations/modes in dimensionality as all locations refer to the same free input set U. Please note, that U has not to be of the same dimensionality as the hybrid system (implicitly defined through matrices A). Furthermore, the smaller the dimensionality of U, the faster evaluations can be computed.

To complete the description of the hybrid system, the transitions have to be defined. This is done using the `transitions` cell array of structures defining with each cell of the array a different transition.

The bouncing ball example has only one transition with an equality guard and an inequality guard. The equality guard is specified as following

```
transitions{1}.eguards_dir = [1  0];
transitions{1}.eguards_val = [0];
```

and describes the guard condition that $\begin{pmatrix}0 & 1\end{pmatrix} \cdot \begin{pmatrix}dimension1 & dimension2\end{pmatrix} \leq 0$
The inequality guard is specified similarly as

```
transitions{1}.iguards_dir = [0  1];
transitions{1}.iguards_val = [0];
```

describing the guard condition that $\begin{pmatrix}0 & 1\end{pmatrix} \cdot \begin{pmatrix}dimension1 & dimension2\end{pmatrix} \leq 0$ If further guards are be introduced, the matrices `eguards_dir` or respectively `iguards_dir` have to be enlarged with subsequent rows describing the normal vectors of the hyperplanes being the newly considered guard and the corresponding `_val` vector has to be enlarged by the value describing the distance of the hyperplane to the origin.

The `bouncingball` example uses a reset on its single transition where the second dimension is set to its value multiplied by -0.6. This is modelled using a reset matrix

```
transitions{1}.ResetMatrix = [1,0;  0,-0.6];
```

The reset matrix's dimensionality is equal to the system's dimensionality. If a dimension is set to zero the correspondent row in the matrix has to be set to 0. Furthermore, it is possible to add constants to specific dimensions during a reset. Those constants are specified using vector W. A value c in the second entry of W would mean that c is added to dimension 2 in case the transition is taken. Because the `bouncingball` example does not define any resets using constants, we set the entries of W for both dimensions to 0.

```
transitions{1}.W =  [0;0];
```

Up to now, the guards and resets for the transitions have been defined but the start and the endpoint specification are missing. As the model consists of only one location/mode, the startpoint and the endpoint are the same

```
transitions{1}.from = 1;
transitions{1}.to = 1;
```

The values of both structure fields refers to the index of the correspondent location/mode in q and the states cell array.

The initial set I is defined using a polyheder representation such that $C * x \leq d$.

```
I.polyheder.C = [1  0;-1  0;0  1;0  -1]  ;
I.polyheder.d = [2;-2;0;0];
```

Once again, the dimensionaity of I has to be the same as the dimensionality of the system (A,W and `ResetMatrix` matrices)

The input set U can be specified using different representations of convex sets, like polyheder (same definition as I), Zonotopes (actually only with center 0) and symmetric box.

```
U.polyheder.C = [1 0;-1 0;0 1;0 -1];        %polyheder definition
U.polyheder.d = [0;0;0;0];
```

```
U.zonotope.c=[0;0]; % zonotope definition
U.zonotope.g=0.01*eye(2,2);
```

```
U.box = [0;0]; % definition as symmetric box
```

The computation speed can increase if U is of lower dimensionality and also in the case if U can be defined as such from the symmetric box representation.

We will now extend the example as introduced above with a new spontaneous transition. Therefore, we introduce a further transition from location 1 to 1 by adding an entry to the transitions cell array

```
transitions{2}.from = 1;
transitions{2}.to = 1;
```

To make the transition spontaneous a correspondingly named field can be added and set to true.

```
transitions{2}.spontaneous = true;
```

To make the new transition with index 2 triggerable upon reaching a fixpoint, set

```
transitions{2}.timeelapse = -1;
```

To make the transition triggerable upon reaching a certain point in time t, set it to

```
transitions{2}.timeelapse = 10; %t=10 in this case
```

Thus, a value greater than zero defines a point in time in which the transition is taken if the system is in the location/mode where the transition starts. If the value for `timeelapse` is set to -1, the transition is considered a spontaneous transition which has to be taken when a fixpoint has been reached. This sort of transition works only, if the user has enabled the fixpoint detection in the user interface.

The `.m` input file format allows to define some standard options, too. Those are the timestep r and the local time horizon T. To set both values, the **configuration** structure is used.

```
configuration.timehorizon = T;
configuration.timestep = r;
```

From both values the number of sets in the flowpipe is calculated as $\frac{T}{r}$.

There is one more feature supported by the model using the `.m` file format. This is the possibility to define invariants for locations/modes. Invariants of the form $C * x \leq d$ are modeled similar to the initial set or guards. For the state which should have an invariant, a correspondent field `C` is added containing the matrix C of the invariant equation. The values d are assigned correspondingly to the field `d`.

```
states{index}.C = C;
states{index}.d = d;
```

19.2 Simulink - Stateflow Charts® as Input Files

As an alternative to `.m` files describing a MATLAB function as input file, it is also possible to model a hybrid system graphically using Simulink Stateflow and to use the generated file as input for the analysis. Figure 50 shows a Simulink Stateflow model for a two tank system.

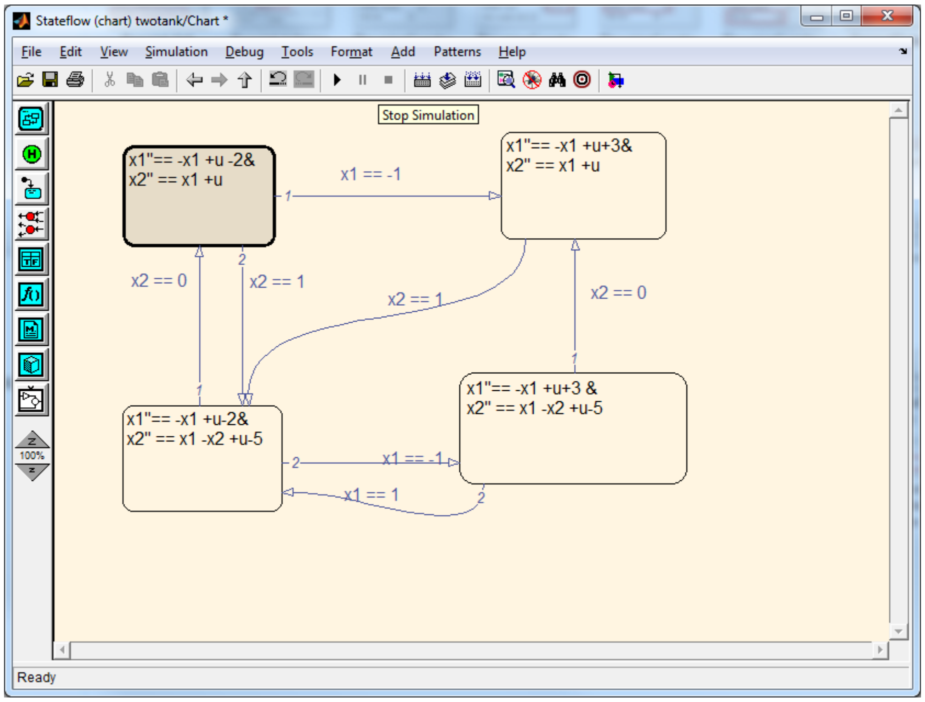


Figure 50: Simulink Stateflow model - two tank

To create a model, the user has to create a new Simulink model and add a Stateflow chart to that model. Only the content of the Stateflow chart is relevant and used as input file. The states of a Stateflow chart are considered to be locations/modes of a hybrid automaton. We intentionally do not use Stateflow syntax in order to avoid the misinterpretation of our models with hybrid automata semantics as real Stateflow models with possibly different semantics. In other words, we use Stateflow only to create our own models with our own syntax and semantics.

To define each location's flow, the corresponding equations are written to the location's label. Each location of the hybrid system has to define a flow for every variable of the system. If one of the equations contains a variable without

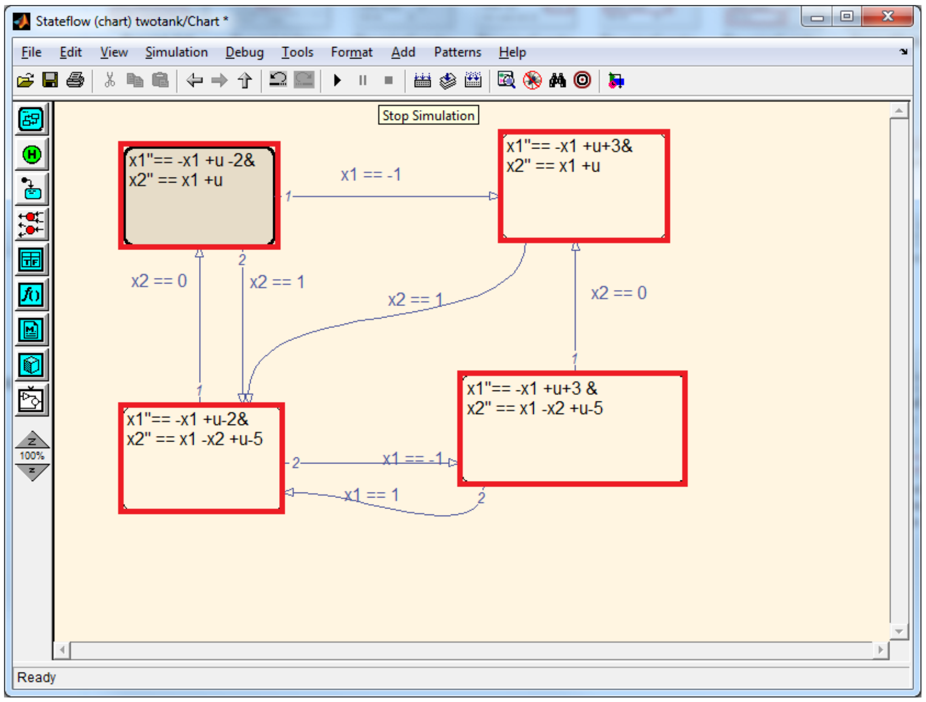


Figure 51: States are interpreted as locations

specifying a flow, this variable is considered to be a variable of the input set. Furthermore, the naming and the type (input set or not) of the variables for all locations, transitions and resets has to be consistent. Equations describing the flow begin with the name of the variable whose flow shall be described, followed by a `"==` and a linear term over all variables. The coefficients of the variables in the linear term may only be followed by a variable name. The names of the variables have to start with a letter and may contain numbers. The different flow equations of one mode are separated using `&` as delimiter. To improve the readability, spaces and line breaks can be used.

Transitions between the states can be added using drag and drop. Every transition has to define a guard or to be a spontaneous transition. The two-tank example given above only uses equality guards. These, for example, are denoted by `x2 == 1`. Inequality guards are denoted accordingly with `<=` or `>=`. Moreover, multiple guards can be used simultaneously at a single transition being separated using & as delimiter. Figure 52 shows an example for the combination of both guard types on one transition (underlined in red). After the guard definitions a `,` has to follow even if no reset is used. Resets are also allowed with spontaneous transitions. The transition on the figure will only be taken if both guards are fulfilled.

The green underlined part of the labeling of the transition on figure 52 de-

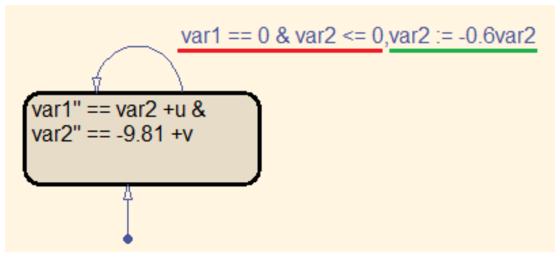


Figure 52: Statflow model of a bouncing ball example

scribes the reset which is applied when the transition is taken. Because every transition has a guard, the definition of the reset (if provided) follows the definition of the guard(s) with a comma as separator. The reset is defined similar to the definition of the flow for a variable in a location. It starts with the name of the variable to reset followed by `:=` and a linear term over the flow variables (without any multiplication signs between coefficients and variables). Multiple linear terms are concatenated using & as delimiter. As before, space and line breaks can be used to make the transition label better readable. To define spontaneous transitions, keywords are used. The use of the keyword `fixpoint` at a transition label indicates that this transition is a spontaneous transition which has to be taken if a fixpoint was detected. The user has to ensure, while using the GUI, that the corresponding detection is enabled. To define a spontaneous transition after a passed time, the keyword `time` followed by a constant defining the time at which the transition has to be taken is used[11].

The initial location is modelled using a transition starting from no location but with the initial location as destination, as can be seen on figure 52. Contrary to the MATLAB function input file format, the graphical modelling does not support the definition of initial sets, the input set U or any computation parameters, as the local time horizon, the timestep size, etc. When using the Simulink Stateflow file format, these values have to be provided using the GUI.

Invariants can be added to the model writing the invariant condition above the flow equations in the locations. Multiple invariants are combined using the `&` sign and a comma terminates the invariant definition. Figure 53 shows a model which defines invariants for three of its four locations.

19.3 Tutorial

This section explains how to perform a complete reachability analysis using the previously described tool. Firstly, a model has to be created. The easiest way to create an input file is using Simulink Stateflow. By executing the MATLAB command **simulink**, it is possible to create a new Simulink model. To create the input file add a Stateflow element to the new Simulink model. Double-clicking

[11]please note that the denoted value relates to the total time passed and not to the time passed in the location under consideration.

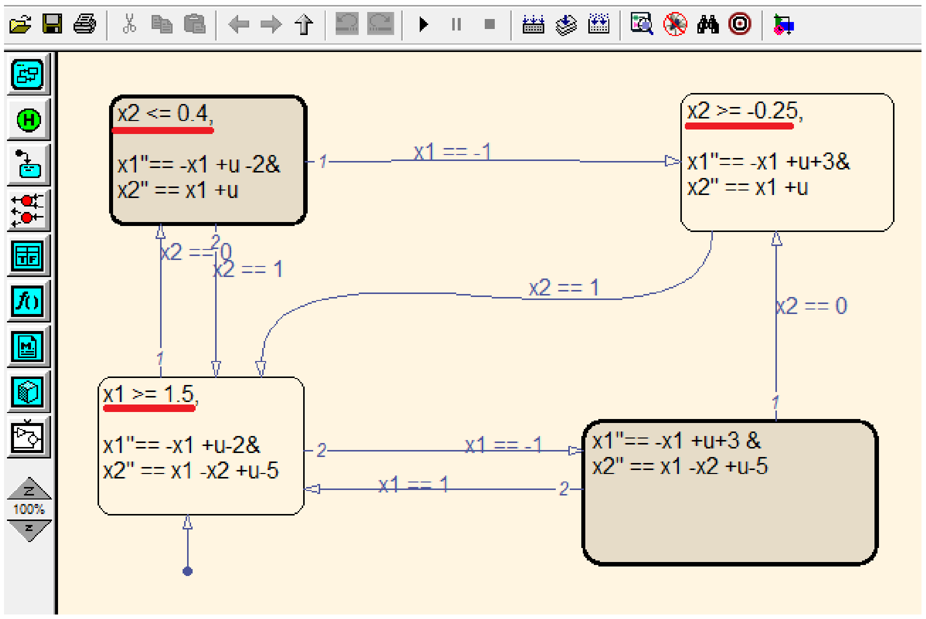
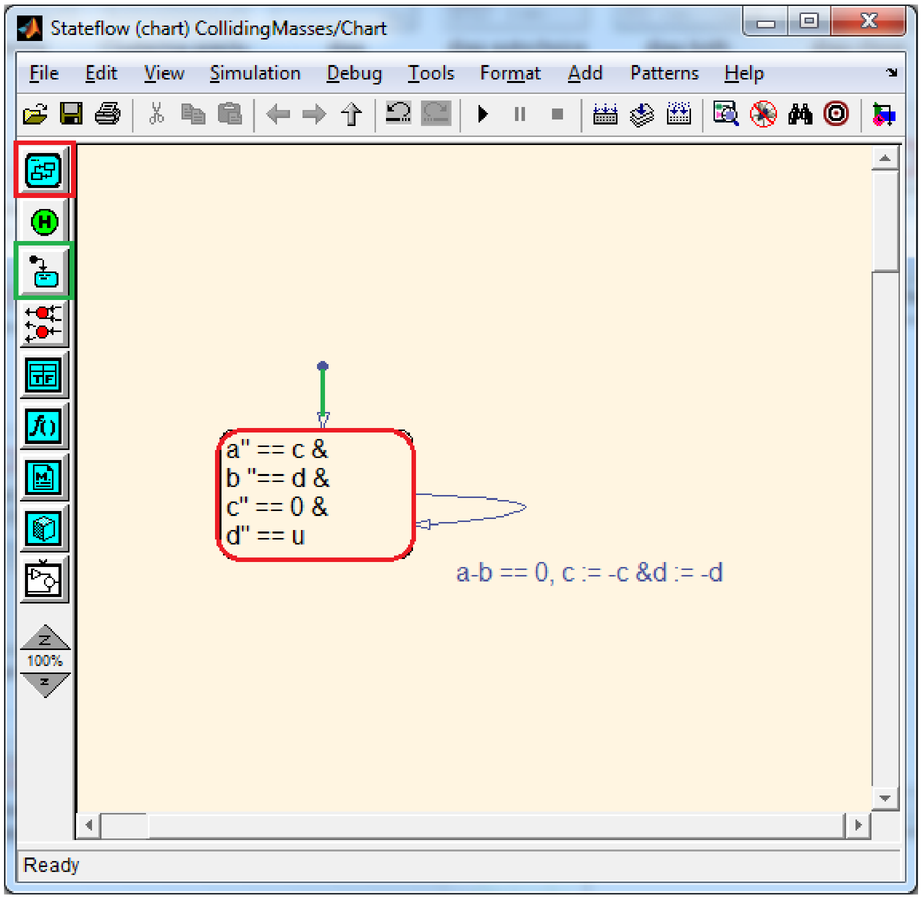


Figure 53: *.mdl model with invariants

on the Stateflow chart in the Simulink model opens the Stateflow view. In this short tutorial, we will model and test a colliding-masses model, which consists of only one state. Add the state to the Stateflow model using the red marked button on figure 54.

To define the added states as the initial state of the model, use the green marked button on figure 54. The transition can be added using drag and drop. The label of the states defines the variables of the system and their flow in this state. For further information on defining the flow, see section 19.2. The transition's labelling defines the transition guards and the reset of the variables if the transition is taken. After completing the modelling, save the model file to an arbitrary location as a `.mdl` file.

To analyze the created model start the reachability analysis tool. This can be done by navigating to the correspondent directory in MATLAB and executing the `gui()` function. Load the model to analyze using the **Locate input file** button. Change the file type filter to ***.mdl** and locate the model file on your hard drive. Next, we define the initial values for the calculation. As initial set, we want to use `a==0 & b==3 & c==2 & d==-1`. Enter this string in the left edit field in the category **Initial values**. The right edit field in this category defines the initial values for the input set. Enter `u==0` in this field to use a single point 0 as input set. The category **General options** allows users to define the number of transitions to consider (= Max. number of iterations-1), the time horizon for the calculation in each location and the sampling time (timestep). Assuming we want to analyze the reachable sets up to a horizon of 1 with a


Stateflow (chart) CollidingMasses/Chart
File Edit View Simulation Debug Tools Format Add Patterns Help
100%
Ready

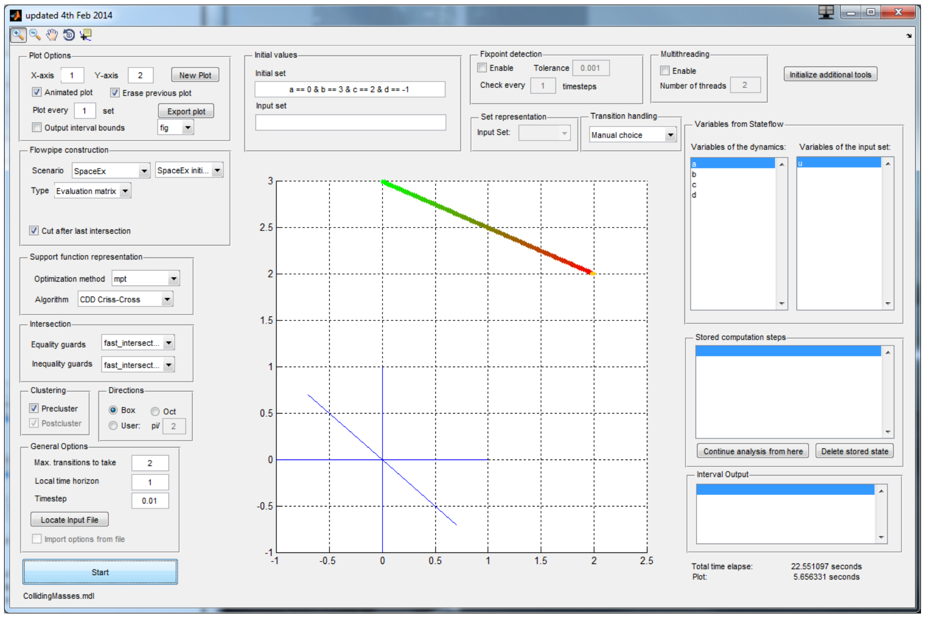
a'' == c &
b ''== d &
c'' == 0 &
d'' == u

a-b == 0, c := -c &d := -d

Figure 54: Colliding masses stateflow model

sampling time of 0.01 taking at most one transition. To plot dimensions other than the dimensions 1 and 2, change the values for the edit fields **X-axis** and **Y-axis** in the category **Plot options**. At this point, the reachability analysis can be started.

However, it is possible to speed up the computation by changing some parameters. By default, the **Precluster** and **box** option are already selected. To increase the speed of calculating the intersection sets, ensure that **fast_intersection** is chosen for equality and inequality guards in the intersection section. To increase the calculation speed choose **mpt** as optimization method and **CDD Criss-Cross** as the algorithm. To use **mpt**, it has to be available on the computing machine. The GUI allows for the initialization of **mpt** using the **Initialize additional tool** button. Finally, choose **Evaluation matrix** as type in the category **Flowpipe construction** and start the analysis. Figure 55 shows the result of the analysis.



updated 4th Feb 2014

Plot Options
X-axis 1 Y-axis 2 New Plot
[✓] Animated plot [✓] Erase previous plot
Plot every 1 set Export plot
[ ] Output interval bounds fig

Flowpipe construction
Scenario SpaceEx SpaceEx initi...
Type Evaluation matrix
[✓] Cut after last intersection

Support function representation
Optimization method mpt
Algorithm CDD Criss-Cross

Intersection
Equality guards fast_intersect...
Inequality guards fast_intersect...

Clustering
[✓] Precluster
[✓] Postcluster

Directions
[✓ Box] Oct User: pi/ 2

General Options
Max. transitions to take 2
Local time horizon 1
Timestep 0.01
Locate Input File
[ ] Import options from file

Start

CollidingMasses.mdl

Initial values
Initial set a == 0 & b == 3 & c == 2 & d == -1
Input set

Fixpoint detection
[ ] Enable Tolerance 0.001
Check every 1 timesteps

Multithreading
[ ] Enable
Number of threads 2

Initialize additional tools

Set representation
Input Set:

Transition handling
Manual choice

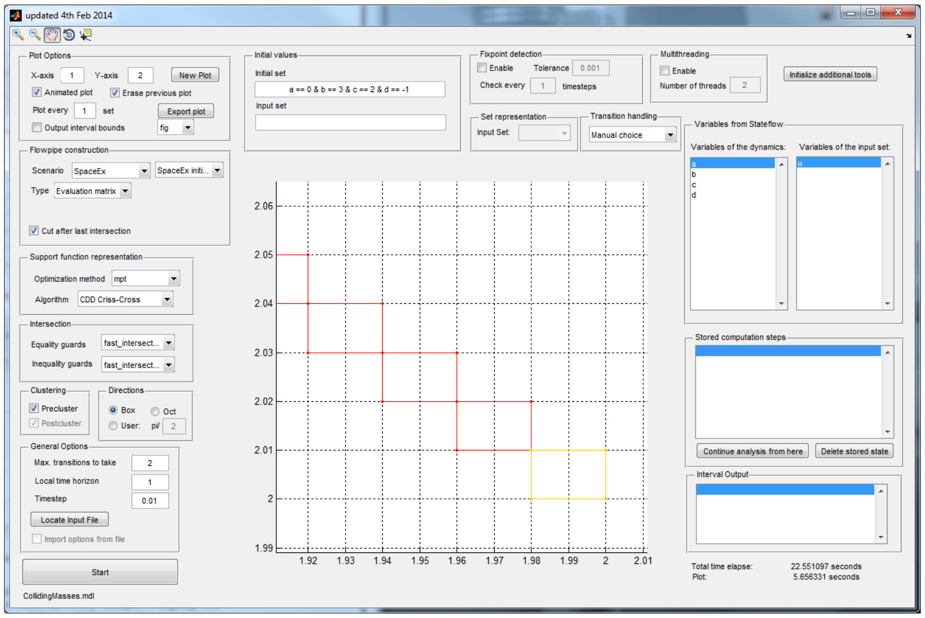
Variables from Stateflow
Variables of the dynamics: a b c d
Variables of the input set: u

Stored computation steps
Continue analysis from here Delete stored state

Interval Output

Total time elapse: 22.551097 seconds
Plot: 5.656331 seconds

Figure 55: Colliding masses plot with fast options



updated 4th Feb 2014

Plot Options
X-axis 1 Y-axis 2 New Plot
[✓] Animated plot [✓] Erase previous plot
Plot every 1 set Export plot
[ ] Output interval bounds fig

Flowpipe construction
Scenario SpaceEx SpaceEx initi...
Type Evaluation matrix
[✓] Cut after last intersection

Support function representation
Optimization method mpt
Algorithm CDD Criss-Cross

Intersection
Equality guards fast_intersect...
Inequality guards fast_intersect...

Clustering
[✓] Precluster
[✓] Postcluster

Directions
[✓ Box] Oct User: pi/ 2

General Options
Max. transitions to take 2
Local time horizon 1
Timestep 0.01
Locate Input File
[ ] Import options from file

Start

CollidingMasses.mdl

Initial values
Initial set a == 0 & b == 3 & c == 2 & d == -1
Input set

Fixpoint detection
[ ] Enable Tolerance 0.001
Check every 1 timesteps

Multithreading
[ ] Enable
Number of threads 2

Initialize additional tools

Set representation
Input Set:

Transition handling
Manual choice

Variables from Stateflow
Variables of the dynamics: a b c d
Variables of the input set: u

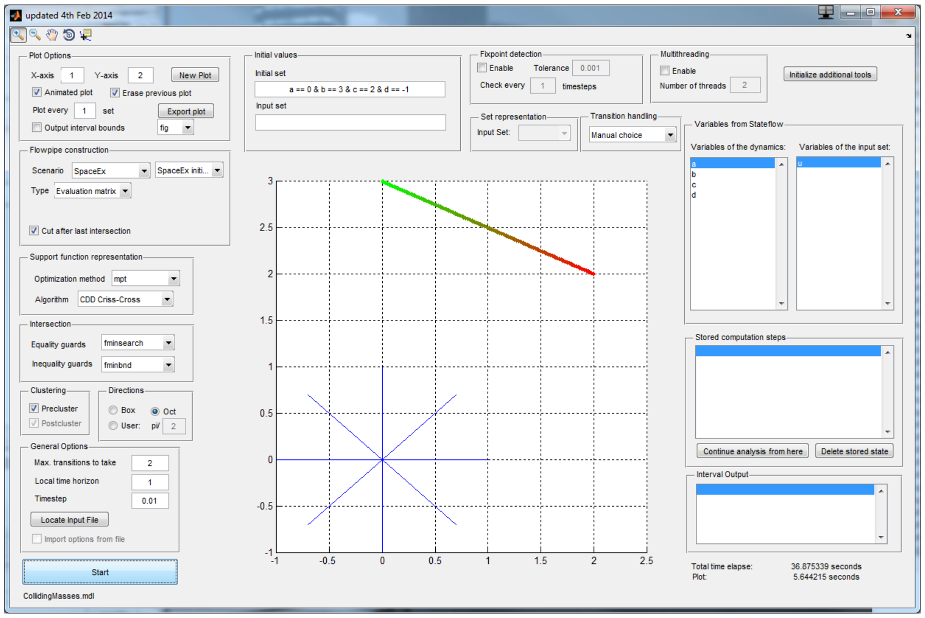
Stored computation steps
Continue analysis from here Delete stored state

Interval Output

Total time elapse: 22.551097 seconds
Plot: 5.656331 seconds

Figure 56: Colliding masses plot with fast options enlarged

Assuming a more accurate (and therefore time consuming) analysis shall be done, choosing different options may help. For instance, disable the **precluster** option in the category **Clustering** and chose **Oct** in the **Directions** category. Use this time **fminsearch** as method to calculate intersections with equality guards and **fminbnd** for inequality guards. A further change which could be made is the optimization method which can be e.g. changed to **linprog** with algorithm **MediumScale**. The use of these options causes the analysis to require much more time before a result is achieved. Figure 57 shows the output result using the different options.



Plot Options
X-axis 1 Y-axis 2
[✓] Animated plot [✓] Erase previous plot
Plot every 1 set
[ ] Output interval bounds fig

Flowpipe construction
Scenario SpaceEx SpaceEx initi...
Type Evaluation matrix
[✓] Cut after last intersection

Support function representation
Optimization method mpt
Algorithm CDD Criss-Cross

Intersection
Equality guards fminsearch
Inequality guards fminbnd

Clustering
[✓] Precluster
[✓] Postcluster

Directions
Box [✓ Oct]
User: pi/ 2

General Options
Max. transitions to take 2
Local time horizon 1
Timestep 0.01
[ ] Import options from file

CollidingMasses.mdl

Initial values
Initial set
a == 0 & b == 3 & c == 2 & d == -1
Input set

Fixpoint detection
[ ] Enable Tolerance 0.001
Check every 1 timesteps

Multithreading
[ ] Enable
Number of threads 2

Set representation
Input Set:

Transition handling
Manual choice

Variables from Stateflow
Variables of the dynamics: a b c d
Variables of the input set: u

Stored computation steps

Interval Output

Total time elapse: 36.875339 seconds
Plot: 5.644215 seconds

Figure 57: Colliding masses plot with slow (intended more accurate) options

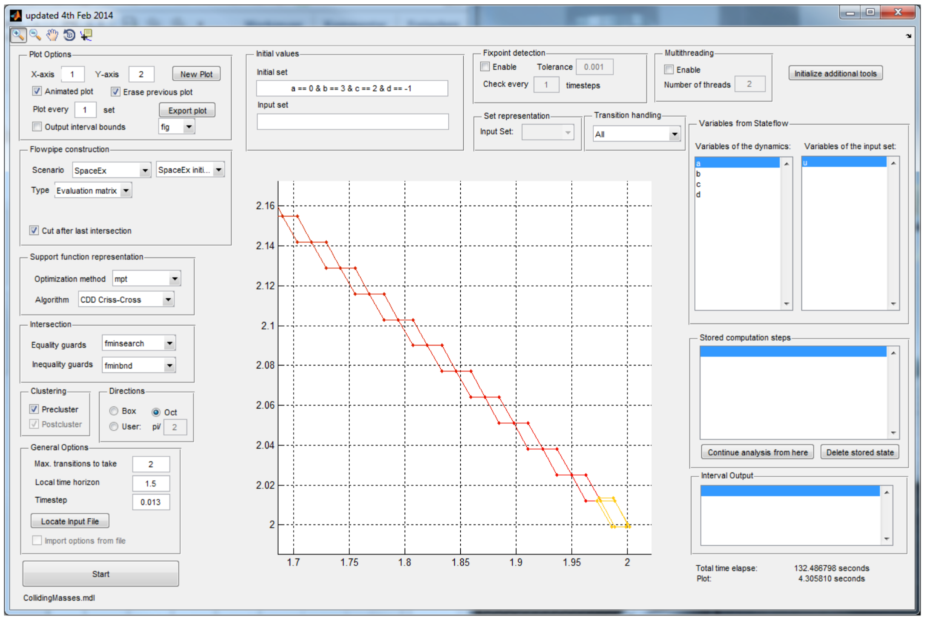
19.4 Adding spontaneous transitions

For this part of the tutorial load the two-tank example as shown on figure 51. The initial set for this example should be $1.5 <= x1 <= 2.5 \& x2 == 1$. Assume we want to trigger a spontaneous transition from the initial location (lower left) to the location on the figures upper right.

To add this spontaneous transition to the two-tank model from 51, add a corresponding transition with the label `time 0.1` to the `.mdl` file. How this should look is shown on figure 60.

To add this spontaneous transition to the `.m` file format add

```
transitions{8}.from = 3;
transitions{8}.to = 1;
```



updated 4th Feb 2014

Plot Options
X-axis 1 Y-axis 2
[✓] Animated plot [✓] Erase previous plot
Plot every 1 set
[ ] Output interval bounds fig

Initial values
Initial set: a == 0 & b == 3 & c == 2 & d == -1
Input set:

Fixpoint detection
[ ] Enable Tolerance 0.001
Check every 1 timesteps

Multithreading
[ ] Enable
Number of threads 2

Set representation
Input Set:

Transition handling
All

Flowpipe construction
Scenario SpaceEx SpaceEx initi...
Type Evaluation matrix
[✓] Cut after last intersection

Support function representation
Optimization method mpt
Algorithm CDD Criss-Cross

Intersection
Equality guards fminsearch
Inequality guards fminbnd

Clustering
[✓] Precluster
[✓] Postcluster

Directions
( ) Box [✓ Oct]
( ) User: pi/ 2

General Options
Max. transitions to take 2
Local time horizon 1.5
Timestep 0.013
[ ] Import options from file

CollidingMasses.mdl

Variables from Stateflow
Variables of the dynamics: a b c d
Variables of the input set: u

Stored computation steps

Interval Output

Total time elapse: 132.486798 seconds
Plot: 4.305810 seconds

Figure 58: Colliding masses plot with slow (intended more accurate) options enlarged

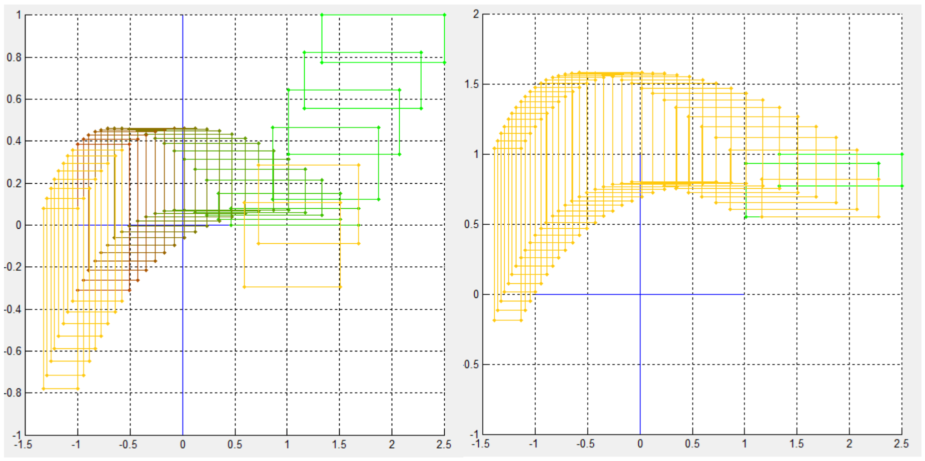


Figure 59: original 2 tank example (left) and 2 tank with added spontaneous transition (right)

```
transitions{8}.spontaneous = true;
transitions{8}.timeelapse = 0.1;
```

The resulting plot of the model with the spontaneous transition is expected

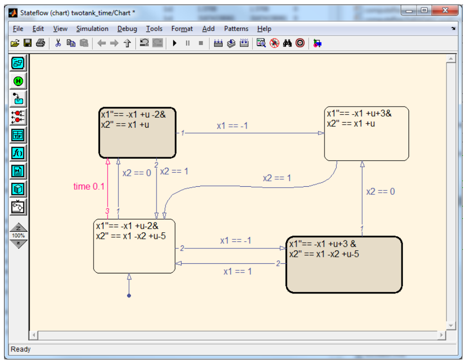


Figure 60: 2 tank example with additional time triggered transition

to be same as the result shown on figure 59's plot on the right side.

References

[1] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. Spaceex: Scalable verification of hybrid systems. In Shaz Qadeer Ganesh Gopalakrishnan, editor, *Proc. 23rd International Conference on Computer Aided Verification (CAV)*, LNCS. Springer, 2011.

[2] Colas Guernic and Antoine Girard. Reachability analysis of hybrid systems using support functions. In *Proceedings of the 21st International Conference on Computer Aided Verification*, CAV '09, pages 540–554, Berlin, Heidelberg, 2009. Springer-Verlag.

[3] Colas Le Guernic. *Reachability Analysis of Hybrid Systems with Linear Continuous Dynamics*. PhD thesis, Grenoble University, November 2009.

[4] Rajarshi Ray. *Reachability Analysis of Hybrid Systems using Support Functions*. PhD thesis, Grenoble University, May 2012.