Pierre-Jean Meyer
Alex Devonport
Murat Arcak

# Interval Reachability Analysis

## Bounding Trajectories of Uncertain Systems with Boxes for Control and Verification

October 21, 2020

Springer Nature

# Preface

Reachability analysis deals with the problem of evaluating the set of all the successor states that can be reached in finite time by a system starting from a given set of initial states, and plays a key role in verifying or enforcing by control the satisfaction of safety-critical specifications. Although there exists a wide variety of methods and set representations to approximate reachable sets, this book focuses on computing reachable set over-approximations in the form of multi-dimensional intervals (hyper-rectangles) due to their greater simplicity of use and lower computational complexity.

The main motivation and purpose of this book is to provide to the readers a tutorial presentation of several approaches for interval reachability analysis. Instead of focusing on mathematical developments of theorems that can already be found in the relevant publications cited in each chapter, we rather create a unified framework for the high-level description of the core idea of each method, and discuss the main requirements and assumptions that need to be satisfied to apply them. This didactic unification of various interval reachability methods was initiated with our recent development of TIRA (*Toolbox for Interval Reachability Analysis*), which is a MATLAB library gathering most of the methods described in this book. The toolbox and its documentation are publicly available at the following address:

`https://gitlab.com/pj_meyer/TIRA`

and an overview of the toolbox structure and its use is provided in Appendix B.

The field of reachability analysis and the interests in interval reachability analysis are reviewed in Chap. 1, where we also provide the formal definition of the reachability problems considered throughout this book.

Part I describes six main methods for interval reachability analysis. The presentation of all these methods and their respective sub-methods and variations follows the same structure, which focuses on first clearly stating the requirements and limitations for the application of the method, and then listing the main steps needed to obtain the final interval over-approximation of the reachable set. The methods presented in this book rely on interval analysis reviewed in Chap. 2, monotonicity in Chap. 3, mixed monotonicity for continuous-time and discrete-time systems in Chap. 4, and for sampled-data systems in Chap. 5, growth bounds and contraction in Chap. 6, and sampling-based approaches in Chap. 7.

In Part II, we then provide several applications in both academic and more realistic scenarios to illustrate how these reachability methods are used and in what kind of problems they are most relevant. In Chap. 8, reachability analysis is used on both a tunnel diode oscillator and a quadrotor model to verify whether the systems satisfy safety specifications (keeping the state in a safe set at all time) and reachability specifications (reaching a target set in finite time). In Chap. 9, we use the volume of the interval over-approximation of the reachable set to define a measure of the robustness for the controller of a medical exoskeleton. Finally, Chap. 10 presents how reachability analysis plays a central role in the field of abstraction-based control synthesis, where the continuous model of a marine vessel is abstracted into a finite transition system, and a controller for the continuous system with respect to a reach–avoid specification is synthesized by applying model checking algorithms on the finite abstraction.

The intended audience for this book includes graduate students, researchers, and practicing engineers. We believe that the exposition will appeal not only to those with previous experience with reachability analysis or interval methods, but also to those discovering interval reachability analysis for the first time or using it to solve problems similar to those provided in Part II. Previous knowledge and experience on reachability analysis are not required to read this book, since we made our best efforts to present the methods in the most pedagogical manner.

Berkeley, California
October 2020

*Pierre-Jean Meyer*
*Alex Devonport*
*Murat Arcak*

# Contents

**Part I Reachability Methods**

**Part II Applications**

# Appendix B
# TIRA
# Toolbox for Interval Reachability Analysis

This appendix gives an overview of *TIRA: Toolbox for Interval Reachability Analysis*, which is a MATLAB toolbox, initially published in [1], gathering several of the methods presented in this book, and is continually updated to include new ones. The toolbox and its documentation are publicly available at the following address:

`https://gitlab.com/pj_meyer/TIRA`

## B.1 Toolbox Description

The development choices of TIRA were motivated by three main design goals: providing a library of several methods for interval reachability analysis; being easily extensible with new methods; and being usable by non-expert users without requiring a detailed knowledge of the theory and implementation of these methods. To achieve these objectives, we designed the toolbox architecture so that each reachability method is implemented in its own function, and all methods can be accessed through a unique interface function. This interface function serves as a hub taking as inputs the definition of the reachability problem to be solved (time range and intervals of initial states and inputs), checking if the user-provided system satisfies the requirements of the requested reachability method, and finally calling this method and returning the resulting reachable set over-approximation.

This architecture greatly benefits the toolbox extensibility, since adding a new method only requires to implement this method in a separate function, then add a call for this function and the tests for its requirements in the main interface function. To allow its use by non-expert users, it is also possible to call this interface function without specifying which reachability method to call. In such case, the interface function checks the requirements of all implemented methods and calls the most suitable one whose requirements are satisfied by the user-provided system. In addition, although we offer an easy access for the user to modify the internal parameters of the reachability methods if desired, all those parameters are predefined with default values so that non-expert users can still use the toolbox without knowledge of how each method and its internal parameters work.

The toolbox is implemented in MATLAB, and is thus platform independent and does not require an installation. More details on requirements, setup and use of the toolbox are provided in the documentation available in the online repository at the link in the introduction of this appendix.

## B.2 Guided Use through an Example

Several examples are already implemented in TIRA (folder `Examples/`) to test the toolbox. In this section, we instead provide a didactic description of how to introduce and solve a new reachability problem in TIRA. For this, we consider the tunnel diode oscillator from Sect. 8.1.

The first step is to provide the system description by filling the skeleton function `System_description.m`.

```
%% In file System_description.m

function dx = System_description(t,x,p)
% Constant parameters
C = 1e-12; L = 1e-6; R = 200; V_in = 0.3;
% Polynomial current-voltage relationship
i_D = @(v_D) 1e-3/.3*(803.712*v_D^5 - 1086.288*v_D^4 + 551.088*
    v_D^3 - 124.548*v_D^2 + 10.656*v_D);
% Continuous-time dynamics
dx = [(-i_D(x(1))+x(2))/C; ...
      (-x(1)-R*x(2)+V_in)/L];
```

Depending on which reachability methods we want to apply to this system, we might need to provide additional information on the system to satisfy the methods' requirements. For this example, we first provide bounds on the Jacobian matrices in file `Input_files/UP_Jacobian_Bounds.m` to satisfy Assumption 4.1 (with the help of Appendix A) to use the method based on continuous-time mixed monotonicity in Sect. 4.1.

```
%% In file Input_files/UP_Jacobian_Bounds.m

function [J_x_low,J_x_up,J_p_low,J_p_up] = UP_Jacobian_Bounds(
    t_init,t_final,x_low,x_up,p_low,p_up)
% Constant parameters
C = 1e-12; L = 1e-6; R = 200;
% State Jacobian bounds
J_x_low = [-inf   1/C; ...
           -1/L   -R/L];
J_x_up = [.0074/C   1/C; ...
          -1/L      -R/L];
% Input Jacobian bounds
J_p_low = zeros(2); J_p_up = zeros(2);
```

We then provide in file Input_files/UP_Contraction_Matrix.m a contraction matrix satisfying Assumption 6.3 to use the growth bound method from Sect. 6.3.

```
%% In file Input_files/UP_Contraction_Matrix.m

function C = UP_Contraction_Matrix(t_init,t_final,x_low,x_up,
    p_low,p_up)
% Constant parameters
C = 1e-12; L = 1e-6; R = 200;
% Componentwise bounds on the state Jacobian matrix
C = [.0074/C  1/C; ...
     1/L      -R/L];
```

Another important file is Solver_parameters.m, in which all internal parameters of the toolbox are predefined and can be modified by the user. In particular, it can be useful to select a specific sub-method for one reachability analysis (e.g. how bounds on the sensitivity matrices are computed from Sects. 5.2–5.4, or how the growth bound function is created from Sects. 6.2–6.4). In this example, we modify the default parameters to ensure that the sensitivity bounds for the sampled-data mixed-monotonicity approach (Assumption 5.1 in Sect. 5.1) are computed based on the sampling method in Sect. 5.3 with 500 sample points and no falsification iteration. Note that the code below only includes the parameters in Solver_parameters.m whose default value was modified.

```
%% In file Solver_parameters.m

%% Sampled-data mixed monotonicity: method and parameter choice

% How are bounds on the sensitivity matrices obtained?
  % 1: Read sensitivity bounds from user-provided files
  % 2: Interval Arithmetics
  % 0: Sampling and falsification
parameters.sensitivity_bounds_method = 0;

%% Sampling and falsification: internal parameters

% Number of sample pairs in [x_low,x_up]*[p_low,p_up]
parameters.sampling_pairs = 500;
% Max number of falsification iterations
parameters.falsification_max_iter = 0;
```

The final steps are in the file MAIN_CALL.m where we first define the reachability problem to be solved (intervals of initial states and inputs, and time range) and then call the toolbox for each desired reachability method. In this example, we call the three methods mentioned above: the continuous-time mixed monotonicity from Sect. 4.1; the growth bound method from Sect. 6.1; and the sampled-data mixed-monotonicity method from Sect. 5.1 with the sampling-based sub-method from Sect. 5.3 as specified above in Solver_parameters.m. Although the numerical example in Sect. 8.1 approximates the reachable tube of this system by computing 500 reachable set over-approximations, the example code below only computes the reachable set over-approximations at the final time 15 ns.

```
%% In file MAIN_CALL.m

% Interval of initial states
x_low = [0.45; 0.6e-3];
x_up = [0.50; 0.6e-3];

% Interval of allowed input values
p_low = [0; 0];
p_up = [0; 0];

% Time interval
t_init = 0;
t_final = 15e-9;

% Toolbox call for continuous-time mixed-monotonicity method
[MM_low,MM_up] = TIRA([t_init,t_final],x_low,x_up,p_low,p_up,3);
% Toolbox call for growth bound method
[GB_low,GB_up] = TIRA([t_init,t_final],x_low,x_up,p_low,p_up,2);
% Toolbox call for sampled-data mixed-monotonicity method
[SD_low,SD_up] = TIRA([t_init,t_final],x_low,x_up,p_low,p_up,4);
```

## References

1. Meyer, P.J., Devonport, A., Arcak, M.: TIRA: toolbox for interval reachability analysis. In: Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, pp. 224–229. ACM (2019)