Robust INner and Outer reachability

RINO User Manual

Eric Goubault and Sylvie Putot

LIX, Ecole Polytechnique, CNRS and Institut Polytechnique de Paris,
91128 Palaiseau, France
name.surname@polytechnique.edu

May 3, 2022

Abstract

We present the C++ RINO library, available on `https://github.com/cosynus-lix/RINO/`, for the computation of inner and outer approximations of reachable sets for uncertain discrete-time or continous-time dynamical systems, with (possibly time-varying) disturbances and control inputs, where some of the control inputs can be specified as outputs of a neural network.

For continuous-time systems, it relies on Taylor expansions in time and affine arithmetic (i.e. zonotopes) in space based reachability analysis to compute outer envelopes of all possible trajectories of an uncertain system. Additionally, it uses a generalized mean-value theorem to deduce inner tubes, that contain only states guaranteed to be reached. It also studies robust versions of these tubes, when there can be control inputs and perturbations to the system. Finally, the control can be specified as the output of a neural network which inputs are the system state.

1 Introduction and references

RINO implements the following:

- Forward inner and outer-approximated reachability of non-linear differential systems [GP17]. The reachability algorithm relies on Taylor expansions in time and affine arithmetic (i.e. zonotopes) in space for computing over or outer-approximating tubes. Under or inner-approximating tubes are deduced by application of a generalized mean-value theorem to the flow of the system. This supposes to compute an over-approximation of the solution flow and its Jacobian with respect to uncertain inputs and initial conditions.
- Forward inner and outer-approximated reachability of non-linear delay differential systems with constant delay [GPS18]. Using the classical method of steps, the problem is reduced to the reachability analysis of a sequence of non-linear differential systems.

- Robust inner and outer approximations of differential systems with possibly time-varying disturbances [GP19]: the above reachability analysis is extended to the case with both disturbances and control inputs.
- In [GP17, GPS18, GP19], inner-approximations are computed for one-dimensional projections. RINO also implements vector-valued inner-approximations [GP20]. (in practice, 2 and 3-dimensional projections).
- In [GP17, GPS18, GP19, GP20], the under-approximation relies on a mean-value theorem, which may be imprecise in some cases. In [GP21], higher-order inner-approximations are proposed. They are implemented in RINO in the case of discrete-time dynamical systems.
- In [GP17, GPS18, GP19, GP20, GP21], the control inputs are specified either in a range (for constant or piecewise constant inputs) or as solution of a differential system (for differentiable time-varying inputs). Some of the control inputs can also be specified as the output of a neural network taking as input the system state [GP22]. This constitutes neural network controlled systems. In RINO, the underlying dynamical system can be either discrete-time or continuous time. For the time being, the activation functions have to be differentiable functions (typically sigmoid and hyperbolic tangent).

2 Installation

2.1 Using docker

Get the RINO directory and run

```
$ docker build .
```

An image `shaxyz...` is built which you can run by

```
$ docker run -it --name rino shaxyz....
```

You can then execute RINO from directory /home/RINO as described in Section 3.

2.2 Building from sources

- You need g++, LAPACK and BLAS installed. Python visualization was tested with Python 3.8.8.
- Install the FILIB++ Interval Library, available from `http://www2.math.uni-wuppertal.de/wrswt/software/filib.html` (we used Version 3.0.2), and set variable $FILIBHOME
- Get and unzip the FADBAD++ automatic diffentiation package, available from `http://www.fadbad.com/fadbad.html` (we used FADBAD++ 2.1), and set variable $FADBADHOME. Copy files fadbad.h and fadiff.h from RINO/FADBAD_Modified/ into your FADBAD++ distribution (we modified these files to add differentiation of activation functions).
- A slightly modified of the third party package for Affine Arithmetic aaflib-0.1 (`http://aaflib.sourceforge.net`) has been included in the current directory. Future plans include separating more cleanly the initial version and our modifications... Go to directory aaflib-0.1 within the current package and compile by "make static".
- Returning to the main (RINO) directory, you can now compile by "make" and obtain the "main" executable.

The installation has been mostly tested on MacOS, but should also work on Ubuntu.

3 Running the reachability analysis

For now, the dynamics of systems on which to perform reachability analysis are defined as C++ code and given a fixed id used to run their analysis:

- for ODEs and DDEs in ‘‘‘ode_def.h‘‘‘ (system and constant parameters) and ‘‘‘ode_def.cpp‘‘‘ (parameters, initial conditions and input ranges)
- for discrete-time systems in ‘‘‘discrete_system.h‘‘‘ and ‘‘‘discrete_system.cpp‘‘‘

Running an example is then performed at command line, in directory /home/RINO, by

```
$ ./rino [-systype system_type -syschoice system_id] [-nnfile-sfx nnfile.sfx]
[-configfile cfgfile.txt]
```

where

- `system_type` is either `ode` (for a system of Ordinary Differential Equations) or `dde` (for a system of Delay Differential Equations) or `discrete` (for a discrete-time dynamical system)
- `system_id` is an integer specifying the predefined system identifier (matching variable syschoice in file ode_def.h for ODEs and DDEs and discrete_system.h for discrete-time systems)
- `nnfile.sfx` contains a neural network in the Sherlock `sfx` format (`https://github.com/souradeep-111/sherlock/blob/master/sherlock-network-format.pdf`)
- `cfgfile.txt` specifies analysis parameters, inputs, initial conditions of the system.

Note that default values for parameters, inputs and initial conditions of the system are set in the code. If a configuration file is used, the configuration file values override those present in the code.

At command line, either the system type and choice should be specified, or a configuration file containing this information should be provided. If both are provided, the configuration file information overrides command-line options. Finally, the name of file containing the neural network, when relevant, can be provided either at command-line or in the configuration file.

The parameters which can be set in the configuration file are described in Section 4. The commands for running the different examples presented in our work are given in Section 3.1.

3.1 Running existing systems

3.1.1 Continuous-time differential systems (ODEs)

- The Brusselator example [GP17] (the system is an ODE and is given syschoice identifiant equal to 2) is run by:

  ```
  $./rino -systype ode -syschoice 2
  ```

  or if you want to use a configuration file to modify the parameter and initial conditions, by:

  ```
  $./rino -configfile Examples/ConfigFiles/cfg_ode_2.txt
  ```

  In what follows, we will use the following aggregate notation to indicate these two alternatives:

  ```
  $./rino -systype ode -syschoice 2 [-configfile Examples/ConfigFiles/cfg_ode_2.txt]
  ```

- The self-driving car example [GP19] is run by

  ```
  $./rino -systype ode -syschoice 6 [-configfile Examples/ConfigFiles/cfg_ode_6.txt]
  ```

3.1.2 Continuous-time delay differential systems with constant delays (DDEs)

- The running example of [GPS18] is run by

  ```
  $ ./rino -systype dde -syschoice 1 [-configfile Examples/ConfigFiles/cfg_dde_1.txt]
  ```

- Example 10 of [GPS18] is run by

  ```
  $./rino -systype dde -syschoice 3 [-configfile Examples/ConfigFiles/cfg_dde_3.txt]
  ```

- Example 9 (self-driving car with uncertain PID coefficients) of [GPS18] is run by

  ```
  $./rino -systype dde -syschoice 8 [-configfile Examples/ConfigFiles/cfg_dde_8.txt]
  ```

- The platoon examples of [GPS18] are run, for 5 vehicles by

  ```
  $./rino -systype dde -syschoice 10 [-configfile Examples/ConfigFiles/cfg_dde_10.txt]
  ```

  or for 10 vehicles by:

  ```
  $./rino -systype dde -syschoice 11 [-configfile Examples/ConfigFiles/cfg_dde_11.txt]
  ```

3.1.3 Discrete-time dynamical systems

3.1.4 Neural network controlled dynamical systems (continuous-time or discrete-time)

3.2 Modifying / adding one's own example

4 Parameters and Configuration File

4.1 Sample configuration file: parameters common to all system types

4.2 Parameters specific to ODEs (when systype is ode)

4.3 Parameters specific to DDEs (when systype is dde)

4.4 Parameters specific to discrete-time systems (when systype is discrete)

4.5 Parameters specific to neural network controlled dynamical systems (systype can be either ode or discrete)

5 Visualizing results

Analysis output files After running an example, all results are in the subdirectory 'output'. They are provided in the following files :

- `sumup.txt`: summary of configuration, running time and ranges at the final state of the analysis (part of this information can also be found with more significant digits in `sumup.yaml`)
- `samplesreachset.yaml`: sampled trajectories (used to assess accuracy of reachability results)
- `approxreachset.yaml`: over and under-approximated reachset (projected, robust, joint ranges) and accuracy measures (eta, gamma) at each time step

Running the visualization script A python visualization file `Visu_output.py` is available in the GUI directory. It can be run from the analyzer (if variable create-png is set to 1 in the configuration file) but you can also run it separately, provided the above data files are present in the output subdirectory of RINO. For example, for an interactive analysis (prints the figures on screen, otherwise the files are simply saved in the output directory) and to produce figures only for variables x[1] and x[2]), it is run by:

```
$ cd GUI; python3 Visu_output.py --interactive=1 --printvar=-1-2; cd ..
```

When the script is run by analyzer, the options set above in command line can be set in the configuration file by:

```
interactive-visualization = 1
variables-to-display = 1 2
```

One-dimensional projections For k ranging from 1 to system dimension, the following results files display the projected ranges on dimension k as function of time:

- `xk_max.png` (e.g. `x1_max.png`) and `xk_max_sample.png`: the maximal inner and outer-approximations, with and without sampled trajectories
- `xk.png`, `xk_sample.png`: additionally to the maximal inner and outer-approximations, the robust approximations when relevant, with and without sampled trajectories

Global views are provided: `xi_max.png` and `xi_subplots_min_max.png` display the one-dimensional projected reachable sets for all variables on one graph.

Two and three-dimensional projections For any couple (k,l) we also display 2-dimensional projections:

- `xkxl.png`: maximal (and when relevant robust [GP19]) inner and outer-approximations of the joint or vector-valued range (xk,xl) as skewed boxes. (see e.g. [GP20])
- `xkxl_sample.png`: sampled trajectories for (xk,xl)

- `xkxl_approx_sample.png`: on the same graph the inner and outer-approximations of the joint range (xk,xl) as skewed boxes and sampled trajectories
- `xkxl_box_sample.png`: same as above but the approximations are printed as boxes (useful in a few cases where the skewed boxes have a bad behavior)
- `xkxl_finalstate.png`: box and skewed box inner and outer-approximations, robust when relevant, and sampled points at the final state of the analysis

Three-dimensional projections when relevant are also printed, only the corners of boxes are printed for more lisibility.

Error measures We display the following error measures (the closer to 1 the better) as functions of time (or iterations for discrete-time systems): `eta.png`, `gamma.png`: error measures (η_o = (width of sampled set)/(width of outer-approx) ; η_i = (width of inner-approx)/(width of sampled set); γ = (width of inner-approx)/(width of outer-approx)

6 Examples

7 License

References

[GP17] E. Goubault and S. Putot. Forward inner-approximated reachability of non-linear continuous systems. In *HSCC*. ACM, 2017.

[GP19] Eric Goubault and Sylvie Putot. Inner and outer reachability for the verification of control systems. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2019, Montreal, QC, Canada, April 16-18, 2019*, pages 11–22. ACM, 2019.

[GP20] E. Goubault and S. Putot. Robust under-approximations and application to reachability of non-linear control systems with disturbances. *IEEE Control Systems Letters*, 4(4):928–933, 2020.

[GP21] Eric Goubault and Sylvie Putot. Tractable higher-order under-approximating AE extensions for non-linear systems. In Raphaël M. Jungers, Necmiye Ozay, and Alessandro Abate, editors, *ADHS 2021*, volume 54 of *IFAC-PapersOnLine*, pages 235–240. Elsevier, 2021.

[GP22] Eric Goubault and Sylvie Putot. Rino: Robust inner and outer approximated reachability of neural networks controlled systems. In *Submitted as tool paper to CAV 2022*, 2022.

[GPS18] Eric Goubault, Sylvie Putot, and Lorenz Sahlmann. Inner and Outer Approximating Flowpipes for Delay Differential Equations. In *Proceedings of the 30th International Conference on Computer Aided Verification, Part II*, pages 523–541. Springer, 2018.