SoftwareX 10 (2019) 100325

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

LyapXool – a program to compute complete Lyapunov functions

Carlos Argáez[a,*], Jean-Claude Berthet[b], Hjörtur Björnsson[a], Peter Giesl[c], Sigurdur Freyr Hafstein[a]

[a] *Science Institute, University of Iceland, Dunhagi 3, 107 Reykjavík, Iceland*
[b] *Vatnaskil, Sídumúli 28, 108 Reykjavík, Iceland*
[c] *Department of Mathematics, University of Sussex, United Kingdom*

A R T I C L E   I N F O

Article history:
Received 12 June 2019
Received in revised form 13 August 2019
Accepted 30 August 2019

Keywords:
Complete Lyapunov functions
C++
Scientific computing
Dynamical systems
Autonomous ordinary differential equations

A B S T R A C T

LyapXool is a C++ program to compute complete Lyapunov functions and their orbital derivatives for any two- or three-dimensional dynamical system expressed by an autonomous ordinary differential equation. The program is user-friendly and determines the chain-recurrent set. This paper describes how the code is organized, how it can be used and provides an interesting example of its application.

Code metadata: General information about this code.

Current code version	v1
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX_2019_207
Legal Code License	GNU General Public License v3.0
Code versioning system used	None
Software code languages, tools, and services used	C++, Armadillo, OpenMP
Compilation requirements, operating environments & dependencies	Unix-like systems
Link to developer documentation/manual	github.com/LyapXool/V1
Support email for questions	carlos@hi.is

1. Motivation and significance

Solutions of an autonomous differential equations of the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}), \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^n$, $n \in \mathbb{N}$, describe general time-changing phenomena and often arise from applications. The general behavior of solutions, depending on their initial condition, is thus of fundamental interest. Numerical simulations of (1) under various different initial conditions are computationally expensive. For that reason, we consider an approach that characterizes the behavior of the system by a so-called complete Lyapunov function.

A *complete Lyapunov function* $V : \mathbb{R}^n \to \mathbb{R}$, introduced in [1–4], is defined on the whole phase space. It does not increase along solutions of the differential equation (1) and thus provides information about stability through analyzing the function's minima and maxima. Moreover, it divides the phase space $\mathbb{R}^n$ into two disjoint areas: The area of the gradient-like flow, where the function strictly decreases along solutions and thus solutions flow through, and the chain-recurrent set, where the function is constant along solutions and where infinitesimal perturbations can make the flow recurrent. These two areas can be characterized by the sign of the orbital derivative $V'(\mathbf{x}) = \nabla V(\mathbf{x}) \cdot \mathbf{f}(\mathbf{x})$, the

* Corresponding author.
E-mail address: carlos@hi.is (C. Argáez).

https://doi.org/10.1016/j.softx.2019.100325

derivative along solutions: $V'(\mathbf{x}) < 0$ in the area with gradient-like flow and $V'(\mathbf{x}) = 0$ in the chain-recurrent set. In this paper we explain the computational code we have written to compute complete Lyapunov functions and to classify the chain-recurrent set. Some authors of this papers have explained the mathematical algorithm before in [5–9], which is based on an algorithm to compute classical Lyapunov functions, see [10].

There are other methods to analyze the general behavior of dynamical systems. The direct simulation of solutions with many different initial conditions is costly and can only give limited information about the general behavior of the system. Other, more sophisticated methods include determining the boundaries of basins of attraction of attractors by computing invariant manifolds [11]. The cell mapping approach [12] or set oriented methods [13] divide the phase space into cells and compute the dynamics between those cells [14].

Other methods to compute complete Lyapunov functions [15–17] start by considering the discrete-time system given by the time-T map, subdivide the phase space into cells and compute the dynamics between them through an induced multivalued map using the computer package GAIO. A complete Lyapunov function is then obtained using graph algorithms [9]. However, this approach has the disadvantage of requiring a large number of cells even in low dimensions.

In [18], a complete Lyapunov is constructed as a continuous piecewise affine (CPA) function, affine on each simplex of a fixed simplicial complex. However, this method assumes that information about local attractors is available, while the proposed method in this paper does not require any information about the system under consideration.

The method in this paper is inspired by the construction of classical Lyapunov functions. It is fast and works well in higher dimensions. In [19], the method described in [15] is compared to the RBF method for equilibria for one particular example.

Computing a complete Lyapunov function does not require solving the dynamical system for particular initial conditions, instead, the general idea is to approximate a "solution" to the ill-posed partial differential equation (PDE) $V'(\mathbf{x}) = -1$ in order to find a function which has a negative orbital derivative. An approximation v is computed using Radial Basis Functions (RBFs), a mesh-free collocation technique, such that $v'(\mathbf{x}) = -1$ is fulfilled at all points $\mathbf{x}$ in a finite set of collocation points X. However, the computed function v will fail to fulfill the PDE at points of the chain-recurrent set, such as an equilibrium or a periodic orbit, since for some $\mathbf{x}$ in the chain-recurrent set we must have $v'(\mathbf{x}) \geq 0$. This is the key component of our general algorithm to locate the chain-recurrent set; we determine the chain-recurrent set by localizing the area where $v'(\mathbf{x}) \not\approx -1$.

Our code uses mesh-free collocation methods, based on RBFs [10]. As RBFs, we use Wendland functions, which are compactly supported and positive definite functions [20], constructed as polynomials on their compact support. General descriptions about Wendland functions are found in [10].

As collocation points $X = \{\mathbf{x}_1, \dots, \mathbf{x}_N\} \subset \mathbb{R}^n$, we use a subset of a hexagonal grid. This has been shown to minimize the condition number of the collocation matrix for a fixed fill distance [21]. The approximant is found by solving a system of linear equations, given by the collocation matrix, which is a quadratic matrix of size N. We remove all equilibria from the set of collocation points X; not doing so would cause the collocation matrix to be singular. The density of the collocation grid is controlled by a parameter α. More detailed information can be found in [5–10].

Once we know v, we use an evaluation grid $Y_{\mathbf{x}_j}$ around each collocation point $\mathbf{x}_j$ to validate the approximation. LyapXool is coded with two different options for evaluation grids. The first one consists of m equidistant points over μ concentric circles around each collocation point. This grid is extended to spheres in 3D. The second one is a directional grid introduced in [7,9], which places all evaluation points aligned to the flow of the ODE system; at each collocation point, we place $2m$ evaluation points along the direction of the flow with the collocation point in the middle.

1.1. Complete Lyapunov function and chain-recurrent set

A tolerance parameter $-1 < \gamma \leq 0$ is defined and every collocation point $\mathbf{x}_j$, such that there exists a $\mathbf{y} \in Y_{\mathbf{x}_j}$ with $v'(\mathbf{y}) > \gamma$, is marked to be in the chain-recurrent set ($\mathbf{x}_j \in X^0$). The other points are considered to be in the gradient-like flow ($\mathbf{x}_j \in X^-$). After that, the approximation of the complete Lyapunov function can be improved iteratively by solving $v'(\mathbf{x}_j) = r_j$, where r_j is computed from the previous iteration. We have implemented three different strategies to compute the value r_j, cf. [5,6,9] for more information:

A. We define $r_j = -1$ if $\mathbf{x}_j \in X^-$ and $r_j = 0$ if $\mathbf{x}_j \in X^0$
B. r_j is a smooth interpolation of A.
C. r_j is the average of the values $v'(\mathbf{x})$, $\mathbf{x} \in Y_{\mathbf{x}_j}$

The general procedure is given by the following Algorithm.

1. Compute the approximation v_i of the complete Lyapunov function for $i = 0$ by solving $v_i'(\mathbf{x}_j) = -1$ at the collocation points
2. Approximate X^0 using $v_i'(\mathbf{y})$, $\mathbf{y} \in Y_{\mathbf{x}_j}$, for each collocation point $\mathbf{x}_j$. If $v_i'(\mathbf{y}) > \gamma$ for any $\mathbf{y} \in Y_{\mathbf{x}_j}$, then $\mathbf{x}_j \in X^0$, else $\mathbf{x}_j \in X^-$, where $\gamma \leq 0$ is a predefined tolerance parameter
3. Compute r_j using $v_i'(\mathbf{x})$, $\mathbf{x} \in Y_{\mathbf{x}_j}$
4. Compute v_{i+1} by solving $v_{i+1}'(\mathbf{x}_j) = r_j$ for $j = 1, \dots, N$

Set i to $i + 1$ and repeat steps 2. to 4.

2. Software description

LyapXool is written in C++. It requires the *Armadillo C++ library* for linear algebra and scientific computing [22,23] as well as *OpenMP* for multithreading several operations. The program is usually run from the command line in a unix-based operating system. The code's execution generates three different files:

- *data.lpx*, where all the input parameters are written
- files with user-chosen extension. The files contain, e.g., the coordinates of $v_i(\mathbf{x})$ and $v_i'(\mathbf{x})$, the points in the chain-recurrent set, etc.
- *output.lpx*, where information about running times is documented

2.1. Software architecture

The source code is divided into a set of modules, each in a separate file:

- problem.cpp: contains the entry point of the executable
- instructions.hpp: all parameters and conditions carried out during calculations
- generalities.cpp and generalities.hpp: compute the dynamical system's Jacobian, its eigenvalues at given points. Classify critical points as well as include printing functions
- odesystem.cpp and odesystem.hpp: specify the ODEs
- RBF.cpp and RBF.hpp: compute the complete Lyapunov function, its orbital derivative, the collocation grid, the first r_j

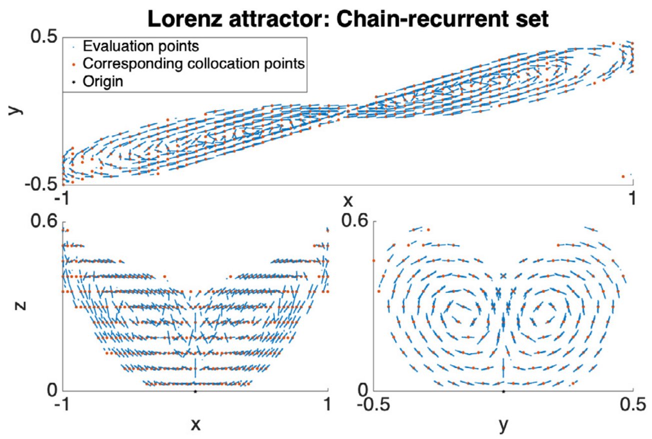

Fig. 1. Three different projections of the computed chain-recurrent set for the Lorenz attractor. Upper: Projection on the xy plane. Lower left: Projection on the xz plane. Lower right: Projection on the yz plane. The z axis in the lower right figure is the same as in the lower left figure.

- chainrecurrentset.cpp and chainrecurrentset.hpp: provide the classification of the chain-recurrent set, evaluation grid and r_j for further iterations
- wendland.cpp and wendland.hpp: construct the Wendland function

2.2. Software functionalities

Our software constructs complete Lyapunov functions and determines the chain-recurrent set.

3. Illustrative examples

Numerous systems have been analyzed using this software [5–9]. In Fig. 1 we show the results for the (scaled) Lorenz attractor's chain-recurrent set obtained with LyapXool with $\alpha = 0.0665$, $\gamma = -0.5$, $r = 0.49$, in the domain $[-1.0, 1.0]^3 \in \mathbb{R}^3$, with the directional evaluation grid and $m = 10$. These figures were obtained after 11 iterations.

4. How to use

- Write the expression for $\mathbf{f}(\mathbf{x})$ in (1), i.e. the dynamical system, in odesystem.cpp and give it a name
- Define that name in *namespace* in instructions.hpp. In this file also define the following parameters for the calculation: domain, critical values, total number of evaluation points per collocation point, type of grid, etc. Finally define the extension into which the results will be printed
- Compile and link with OpenMP and Armadillo
- Run

5. Impact

This software computes complete Lyapunov functions (CLF) for any 2D or 3D dynamical system, whose dynamics are given by an autonomous ODE. It classifies the chain-recurrent set and determines the qualitative dynamics of the system. Hence, it can be used both to analyze given dynamical systems and to derive appropriate models. We expect it to be used in applied sciences using ordinary differential equations, such as biology, computational chemistry, physics, pharmacology, etc. The tool provides a fast and reliable algorithm to construct a CLF to give information on the behavior of any 2D or 3D dynamical system without solving the ODE with many initial conditions.

In [24], some of the authors showed that the cost of computing a complete Lyapunov function with our method using I iterations is of order $O(N^3 + IN^2\ nm)$, where N is the total number of collocation points, n is the dimension of the problem and m is the total number of points to be evaluated in the evaluation grid per collocation point.

6. Future work

We will further develop this program, using experience form different test systems, and make it more sophisticated to allow it to classify the connected components of the chain-recurrent set and their stability. In the future, we will extend our code to handle general n-dimensional systems, and it can also be generalized to discrete-time dynamical systems. Further, we will investigate if it can be extended to compute Lyapunov–Krasovskii functionals for time-delay systems.

7. Conclusions

We have written a C++ code that computes complete Lyapunov functions for any 2- or 3-dimensional dynamical system given by an autonomous ordinary differential equation. This code is based on the theory explained in [10] and [5–9].

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The first author in this paper is supported by the Icelandic Research Fund (Rannís), Iceland grant number 163074-052, Complete Lyapunov functions: Efficient numerical computation. The third author in this paper is supported by the Icelandic Research Fund (Rannís), Iceland grant number 152429-051, Lyapunov Methods and Stochastic Stability.

References

[1] Conley C. Isolated invariant sets and the morse index. CBMS regional conference series, vol. 38, American Mathematical Society; 1978.
[2] Conley C. The gradient structure of a flow i. Ergodic Theory Dynam Systems 1988;8:11–26.
[3] Hurley M. Chain recurrence, semiflows, and gradients. J Dynam Differential Equations 1995;7(3):437–56.
[4] Hurley M. Lyapunov functions and attractors in arbitrary metric spaces. In: Proceedings of the American mathematical society, Vol. 126. 1998, p. 245–56.
[5] Argáez C, Giesl P, Hafstein S. Analysing dynamical systems towards computing complete Lyapunov functions. In: Proceedings of the 7th international conference on simulation and modeling methodologies, technologies and applications (SIMULTECH). Madrid, Spain; 2017, p. 134–44.
[6] Argáez C, Giesl P, Hafstein S. Computational approach for complete Lyapunov functions. In: Awrejcewicz J, editor. Dynamical systems in theoretical perspective. Springer proceedings in mathematics and statistics, vol. 248, Springer; 2018, p. 1–11.
[7] Argáez C, Giesl P, Hafstein S. Iterative construction of complete Lyapunov functions. In: Proceedings of the 8th international conference on simulation and modeling methodologies, technologies and applications (SIMULTECH). Porto, Portugal; 2018, p. 211–22.
[8] Giesl P, Argáez C, Hafstein S, Wendland H. Construction of a complete Lyapunov function using quadratic programming. In: Proceedings of the 15th international conference on informatics in control, automation and robotics (ICINCO). Porto, Portugal; 2018, p. 560–8.

[9] Argáez C, Giesl P, Hafstein S. Computation of complete Lyapunov functions for three-dimensional systems. In: Proceedings of the 57rd IEEE conference on decision and control (CDC). Miami Beach, FL, USA; 2018, p. 4059–64.
[10] Giesl P. Construction of global Lyapunov functions using radial basis functions. Lecture notes in mathematics, vol. 1904, Berlin Heidelberg: Springer-Verlag; 2007.
[11] Krauskopf B, Osinga H, Doedel EJ, Henderson M, Guckenheimer J, Vladimirsky A, Dellnitz M, Junge O. A survey of methods for computing (un)stable manifolds of vector fields. Int J Bifurcationand Chaos Appl Sci Eng 2005;15(3):763–91.
[12] Hsu CS. Cell-to-cell mapping applied mathematical sciences, Vol. 64. New York: Springer-Verlag; 1987.
[13] Dellnitz M, Junge O. Set oriented numerical methods for dynamical systems. In: Handbook of dynamical systems, Vol. 2. Amsterdam: North-Holland; 2002, p. 221–64.
[14] Osipenko G. Dynamical systems, graphs, and algorithms. Lecture notes in mathematics, vol. 1889, Berlin: Springer; 2007.
[15] Ban H, Kalies W. A computational approach to Conley's decomposition theorem. J Comput Nonlinear Dyn 2006;1:312–9.
[16] Goullet A, Harker S, Mischaikow K, Kalies W, Kasti D. Efficient computation of Lyapunov functions for Morse decompositions. Discrete Contin Dyn Syst Ser B 2015;20:2419–51.
[17] Kalies W, Mischaikow K, VanderVorst R. An algorithmic approach to chain recurrence. Found Comput Math 2005;5:409–49.
[18] Björnsson J, Giesl P, Hafstein S, Kellett C, Li H. Computation of Lyapunov functions for systems with multiple attractors. Discrete Contin Dyn Syst 2015;9:4019–39.
[19] Björnsson J, Giesl P, Hafstein S. Algorithmic verification of approximations to complete Lyapunov functions. In: Proceedings of the 21st international symposium on mathematical theory of networks and systems, Vol. 0180. 2014, p. 1181–8.
[20] Wendland H. Error estimates for interpolation by compactly supported Radial Basis Functions of minimal degree. J Approx Theory 1998;93:258–72.
[21] Iske A. Perfect Centre Placement for Radial Basis Function Methods, Technical Report TUM M9809, Technische Universität München, 1998.
[22] Sanderson C, Curtin R. Armadillo: a template-based C++ library for linear algebra. J Open Source Softw 2016;1:26.
[23] Sanderson C, Curtin R. A user-friendly hybrid sparse matrix class in C++. Lecture notes in computer science (LNCS), vol. 10931, 2018, p. 422–30.
[24] Argáez C, Giesl P, Hafstein S. Iterative construction of complete Lyapunov functions: Analysis of algorithm efficiency. Springer; 2019, in press.