Contents lists available at ScienceDirect

SoftwareX

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

Original software publication

Update (2.0) to LyapXool: Eigenpairs and new classification methods

Carlos Argáez [a,*], Peter Giesl [b], Sigurdur Freyr Hafstein [a]

[a] Science Institute, University of Iceland, Dunhagi 3, 107 Reykjavík, Iceland
[b] Department of Mathematics, University of Sussex, United Kingdom

ARTICLE INFO

Article history:
Received 16 September 2020
Received in revised form 21 October 2020
Accepted 22 October 2020

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

ABSTRACT

This is an update to PII: S2352711019302109 LyapXool is a C++ program to compute complete Lyapunov functions and their orbital derivatives for any dynamical system expressed by an autonomous ordinary differential equation. Novel features and the improvements made in the second version are discussed. In particular, the gradient of the complete Lyapunov function and its Hessian, together with its eigenvalues and corresponding eigenvectors, are now computed by the software. This information can be used to analyse the attractivity and dimensions of the components of the chain-recurrent set. Further, the structure of the program is now object-oriented rather than procedural. The user friendliness of the program has been improved. This paper describes how the code is organised, how it can be used to compute complete Lyapunov functions and analyse chain-recurrent sets for dynamical systems, and it provides an interesting example of its application.

Code metadata

Current code version	v2
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-20-00047
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/V2
Support email for questions	carlos@hi.is

Software metadata

Current code version	v2
Permanent link to code/repository used for this code version	github.com/LyapXool/V2
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/V2
Support email for questions	carlos@hi.is

1. Motivation and significance

Solutions to 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 such systems often arise from applications. The general behaviour of solutions, depending on their initial values, is thus of fundamental interest. Numerical simulations of (1) using numerous different initial values are computationally expensive and do not necessarily deliver an exhaustive system description. For that reason, we consider an approach that characterises the behaviour of the system by a so-called complete Lyapunov function (CLF).

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

Definition 1.1 (*Complete Lyapunov Function*)**.** A complete Lyapunov function for (1) is a scalar-valued function $V : \mathbb{R}^n \to \mathbb{R}$, introduced in [1–4], that characterises the qualitative behaviour of the dynamical system. It is defined on the whole phase space of the system and it has the following properties:

- Non-increasing along solutions of the ODE
- It divides the phase space into two fundamentally different areas:
  - (G) the gradient-like flow, where V is strictly decreasing along solution trajectories, and
  - (C) the chain-recurrent set, where V is constant along solution trajectories.
- It separates the different chain-transitive components of the chain-recurrent set into different level-sets.

The areas (G) and (C) can be characterised by the sign of the orbital derivative $V'(\mathbf{x}) = \nabla V(\mathbf{x}) \cdot \mathbf{f}(\mathbf{x})$, the derivative along solutions: (G) $V'(\mathbf{x}) < 0$ in the area with gradient-like flow and (C) $V'(\mathbf{x}) = 0$ in the chain-recurrent set. In this paper we explain the improvements to the first version of the code from LyapXool [5], that was written to compute CLFs and to locate the chain-recurrent set for 2- and 3-dimensional systems. Further, we discuss several novel features. The method in this paper is inspired by the construction of classical Lyapunov functions. It is fast and works in any dimension.

Computing a CLF for a system to analyse its qualitative behaviour has advantages over numerically solving initial-value problems for a large number of starting points because it is computationally more efficient and the results can be rigorously verified [6]. As explained in [5,7–11], in our method we approximate the "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 in a large area. 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 $X \subset \mathbb{R}^n$ of collocation points. However, the computed function v will fail to fulfil the PDE at points in 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 used to locate the chain-recurrent set; we locate the chain-recurrent set by determining the area where $v'(\mathbf{x}) \not\approx -1$.

In this paper we describe the upgrades made to our first code [5] and we also introduce a new tool to compute the gradient and the Hessian of the Lyapunov function. As we have shown in [12], the chain-recurrent set can be determined as the area where the norm of the gradient is zero; the dimension and the attracting- and repelling directions of each of its connected components can be determined by the sign of the eigenvalues and the corresponding eigenvectors.

Our code uses mesh-free collocation methods based on RBFs [13]. As RBFs, we use Wendland functions, which are compactly supported and positive definite functions [14], constructed as polynomials on their compact support. More detailed discussion on the use of Wendland functions for the computation of Lyapunov functions can be found in [13].

For the collocation points we use a hexagonal grid that has been shown to minimise the condition number of the collocation matrix for a given fill distance [15]. The density of the collocation grid is controlled by a parameter α. More detailed information can be found in [7–11,13].

The approximation to the PDE is computed by solving a system of linear equations given by the collocation matrix, which is a quadratic matrix of size $N \times N$, where $N := |X|$ is the number of collocation points. We remove all equilibria from the set of collocation points X; not doing so would cause the collocation matrix to be singular. Once we have computed the approximation, we use a directional evaluation grid $Y_{\mathbf{x}_j}$ at each point $\mathbf{x}_j \in X$ to improve the approximation by iterations as in [5]. The directional evaluation grid was introduced in [9,11]. At each collocation point $\mathbf{x}_j \in X$ we use $2m$ evaluation points along the direction $\mathbf{f}(\mathbf{x}_j)$ of the flow, m upflow and m downflow. A parameter r fixes how far along the direction these $2m$ points are distributed. Typical values are $m = 10$ and $r = 0.49$.

2. Main differences between version 1 and 2

Compared to version 1 of LyapXool [5], the new version 2 includes the following list of added functionality:

- The capability to compute the CLF's gradient and to use it to locate the chain-recurrent set.
- The capability to compute the CLF's Hessian together with its eigenvalues and eigenvectors. This can be used to analyse the dimension and stability of the components of the chain-recurrent set.
- Finally, the method works for systems of arbitrary dimension n and not just for $n = 2$ and $n = 3$.

It also includes improvements of the code and its organisation.

- It is now structured using object-oriented programming, i.e. using classes. This makes the reuse and recycling of different pieces of code easier.
- The choice of the computations to be performed has been simplified.
- The definition of the ODE system to be used has been simplified.
- The generation of the optimal hexagonal collocation grid has been improved considerably by using the algorithm from [16].

Based on numerical experience we also modified some features.

- The localisation of the chain-recurrent set has been decoupled from the collocation points. The classification into (G) and (C) is now done for the points of a regular evaluation grid Ξ, which is independent of the collocation points.
- The circular/spherical evaluation grids for Ξ used in [5,7] have been removed, because we showed in [17] that the regular evaluation grid gives better results for locating the chain-recurrent set.
- Suboptimal iteration methods have been removed in accordance with the results from [9,18].

3. Complete Lyapunov function iteration and chain-recurrent set

As described in [5,7–11], a tolerance parameter $-1 < \gamma_- \leq 0$ is fixed by the user. If the chain-recurrent set is located using the orbital derivative v', every point $\mathbf{y} \in \Xi$ for which $v'(\mathbf{y}) > \gamma_-$ is marked to be in the chain-recurrent set. Similarly, if the gradient of v is used to located the chain-recurrent set as in [12], a tolerance parameter $0 \leq \gamma_+$ is fixed and every point $\mathbf{y} \in \Xi$ such that $\|\nabla v(\mathbf{y})\|_2 < \gamma_+$ is marked to be in the chain-recurrent set.

The approximation of the CLF can be improved iteratively by solving the discretised equation $v'(\mathbf{x}_j) = \min\{r_j, 0\}$ using the RBF method, where the r_j's are computed from the previous iteration. Unlike in [5] we have only implemented one strategy to compute the value r_j, using the average value of v' on $Y_{\mathbf{x}_j}$ because this was shown in [17] to deliver the best results.

The general procedure to compute a CLF for a system (1) is given by the following algorithm.

1. Compute the approximation v_i of a CLF for $i = 0$ by solving $v_i'(\mathbf{x}_j) = -1$, $\mathbf{x}_j \in X$, using the RBF method.
2. Compute r_j as the average of $v_i'(\mathbf{y})$, $\mathbf{y} \in Y_{\mathbf{x}_j}$.
3. Compute v_{i+1} by solving $v_{i+1}'(\mathbf{x}_j) = \min\{r_j, 0\}$, $\mathbf{x}_j \in X$, using the RBF method.
4. Set i to $i + 1$ and go back to step 2.

4. Software description

LyapXool V2 is written in C++. Like the first version [5], it requires the *Armadillo C++ library* for linear algebra and scientific computing [19,20] and uses *OpenMP* for multithreading several operations. The program is usually run from the command line on a unix-based operating system, but we have verified that it also runs on Windows machines provided the appropriate libraries are installed and linked. The code's execution generates three different files:

- *data.lpx*, into which all the parameters of the computation are written.
- Files containing the results of the computations, e.g. the values 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, cardinality of grids, etc., is documented.

4.1. Software architecture

The source code is divided into several separate files:

- `instructions.hpp`: all parameters and conditions carried out during calculations.
- `problem.cpp`: contains the entry point of the executable.
- `odesystem.cpp` and `odesystem.hpp`: specify the ODE system.
- `odetools.cpp` and `odetool.hpp`: functions to analyse critical points of the ODE.
- `rbf.cpp` and `rbf.hpp`: functions to generate the collocation points.
- `lyapunovfunction.cpp` and `lyapunovfunction.hpp`: compute the CLF, its orbital derivative and provide the classification of the chain-recurrent set. Moreover, they construct the gradient of the Lyapunov function, compute the Hessian and classify the eigenpairs.
- `wendland.cpp` and `wendland.hpp`: construction of the Wendland functions and derived functions needed for the computations.
- `generalities.cpp` and `generalities.hpp`: these files contain general functions for printing results and various information about the computations.

4.2. Software functionalities

Our software constructs complete Lyapunov functions (CLFs) for autonomous differential equations and determines the location of their chain-recurrent set, the dimensions of its different connected components and directions of attraction and repulsion.

5. Illustrative examples

Numerous systems have been analysed using this software, cf. e.g. [7–11]. Here we consider system (1) with right-hand side

$$\mathbf{f}(x,y) = \begin{pmatrix} -x(x^2+y^2-1/4)(x^2+y^2-1)-y \\ -y(x^2+y^2-1/4)(x^2+y^2-1)+x \end{pmatrix}. \qquad (2)$$

This system has an asymptotically stable equilibrium at the origin. Moreover, the system has two periodic circular orbits: an asymptotically stable periodic orbit at $\Omega_1 = \{(x,y) \in \mathbb{R}^2 \mid x^2+y^2 = 1\}$ and a repelling periodic orbit at $\Omega_2 = \{(x,y) \in \mathbb{R}^2 \mid x^2+y^2 = 1/4\}$. In Fig. 1 we show the results for this system with LyapXool-v2 with $\alpha = 0.015$, $\gamma_- = -0.5$, $m = 10$ and $r = 0.49$ in the domain $[-1.5, 1.5]^2 \in \mathbb{R}^2$. For the regular evaluation grid Ξ for the localisation of the chain-recurrent set we used $h = 10^{-3}$ as a scaling parameter. The figures were obtained after 10 iterations.

In Fig. 2 we plot the norm of the gradient and the chain-recurrent set obtained from the gradient.

6. How to use

- Write the expression for $\mathbf{f}(\mathbf{x})$ in (1) in `odesystem.cpp`.
- In `instructions.hpp` define the following parameters for the calculation:
  - `ode_dimension`: the dimension n of the ODE.
  - `geometrical_domain_limits`: bounds of the area $D \subset \mathbb{R}^n$ where the CLF is computed. As an example, and with $n = 2$, $\{-1.6, 1.8\}$ for the min and $\{3.6, 4.1\}$ for the max fixes the domain as $D = [-1.6, 3.6] \times [1.8, 4.1]$.
  - `alpha`: parameter to control the density of the hexagonal collocation grid X. Smaller $\alpha > 0$ delivers a denser collocation grid.
  - `normal`: whether or not to use the normalised approach [8], i.e. normalise the speed in which the trajectories of the system are traversed. Default value is `true` with $\delta = 10^{-8}$.
  - `printing`: whether or not to print the results.
  - `critval`: γ_- or γ_+ to be used to classify the chain-recurrent set, see above.
  - `points_directional`: the number of points, downflow and upflow, in each directional evaluation grid $Y_{\mathbf{x}_j}$, $\mathbf{x}_j \in X$. Total number of points in each $Y_{\mathbf{x}_j}$ is $2\times$ this number. Default value is 10.
  - `radius`: fixes the range in which the points of directional grid are distributed. The default value is 0.49.
  - `totaliterations`: total amount of iterations to be carried out.
  - `k,l,c`: Wendland function parameters, cf. e.g. [13,14] for a detailed discussion of these parameters. Default values are $k = 5$, $l = 3$ and $c = 1$.
  - `cart_grid_density`: Scaling parameter h for the regular evaluation grid Ξ, i.e. $\Xi = D \cap h\mathbb{Z}^n$. Smaller $h > 0$ give denser grids.
  - `OMP_NUM_THREADS`: Maximum number of threads to be generated simultaneously by the program. Default value is 4.
  - `outputf`: not to be modified.
- `choose_the_calculation`: select exactly one of the four different options:
  - `only_directional`: sets $\Xi = \bigcup_{\mathbf{x}_j \in X} Y_{\mathbf{x}_j}$. This is fast but does not deliver as good results as when Ξ is a dense regular grid.
  - `directional_and_cartesian`: sets $\Xi = D \cap h\mathbb{Z}^n$ as described above. The results using the directional grid are also reported.
  - `chain_recurrent_set_eigenvalues`: only computes the Hessian and its eigenpairs for points $\mathbf{y} \in \Xi$ in the chain-recurrent set.
  - `norm_chain_recurrent_set`: computes the chain-recurrent set using the norm of the gradient.
- Compile and link with OpenMP and Armadillo. See file `makeLyapXoolV2`.
- Run

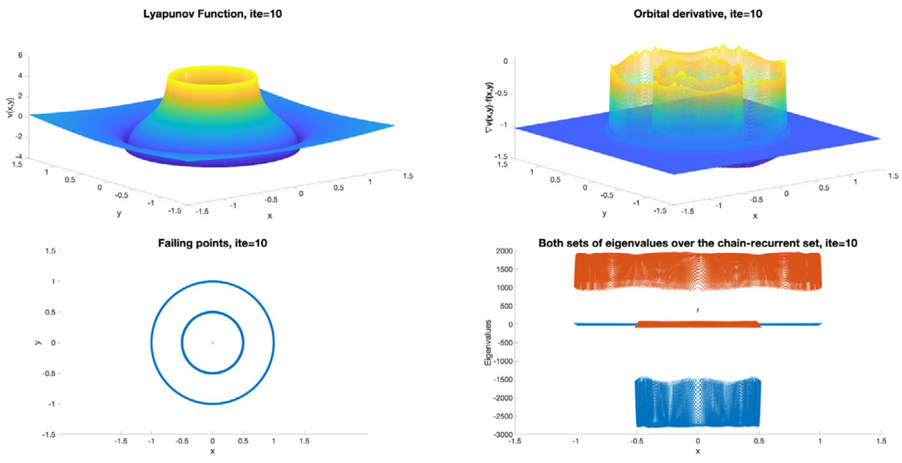

Fig. 1. Upper left: A CLF obtained with LyapXool-v2. Upper right: its orbital derivative. Lower left: Chain-recurrent set obtained using the orbital derivative. Lower right: Eigenvalues of the Hessian of the CLF on the chain-recurrent set. The outer orbit, in blue, is 1-dimensional, because one eigenvalue is zero, and that it is attractive as the other is positive. Likewise, the inner orbit, in red, is 1-dimensional, because one eigenvalue is zero, and it is repelling because the other eigenvalue is negative. The equilibrium at the origin is 0-dimensional and attractive as it has two positive eigenvalues.

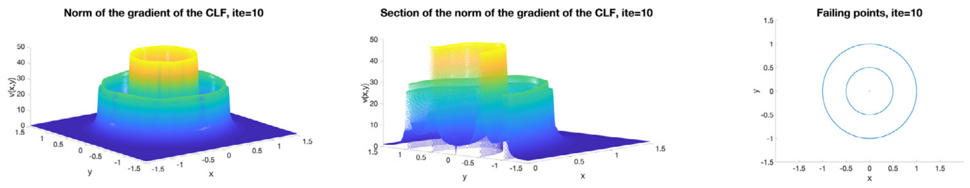

Fig. 2. Left: Norm of the gradient $\|\nabla v(\mathbf{x},\mathbf{y})\|_2$ of the CLF after iteration 10. Middle: Detail of the previous figure displaying $x > 0$. Right: Chain-recurrent set determined by the gradient of v.

7. Impact

This software computes complete Lyapunov functions (CLF) for dynamical system in arbitrary dimension whose dynamics are given by an autonomous ODE. It locates the chain-recurrent set, which determines the qualitative dynamics of the system, and analyses its connected components. Hence, it can be used both to analyse given dynamical systems and to derive appropriate models. We expect it to be used in applied sciences where ordinary differential equations are used, 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 qualitative behaviour of a dynamical system without solving the ODE with many initial values.

8. Conclusions

We have published the C++ code for the computation of complete Lyapunov functions for dynamical systems given by autonomous ordinary differential equations. The code is a major revision of the code in [5] with added functionality and improved algorithms and structure. The code is based on the theory explained in [7–11,13].

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.

Acknowledgements

The first author in this paper is supported by the Icelandic Research Fund (Rannís) grant number 163074-052, Complete Lyapunov functions: Efficient numerical computation. First author thanks Dr. Jean-Claude Berthet for his very good advices on programming.

References

[1] Conley C. Isolated invariant sets and the morse index. In: AMS, CBMS regional conference series. 1978, p. 38. http://dx.doi.org/10.1090/cbms/038.
[2] Conley C. The gradient structure of a flow I. Ergodic Theory Dynam Systems 1988;8:11–26. http://dx.doi.org/10.1017/S0143385700009305.
[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. Proc Amer Math Soc 1998;126:245–56.
[5] Argáez C, Berthet J-C, Björnsson H, Giesl P, Hafstein SF. LyapXool - a program to compute complete Lyapunov functions. SoftwareX 2019;10. http://dx.doi.org/10.1016/j.softx.2019.100325.
[6] Giesl P, Hafstein S. Computation and verification of Lyapunov functions. SIAM J Appl Dyn 2015;14:1663–98. http://dx.doi.org/10.1137/140988802.
[7] 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. http://dx.doi.org/10.5220/0006440601340144.
[8] 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. http://dx.doi.org/10.1007/978-3-319-96598-7_1.
[9] Argáez C, Giesl P, Hafstein S. Terative 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. http://dx.doi.org/10.5220/0006835402110222.
[10] 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. http://dx.doi.org/10.5220/0006944305600568.
[11] 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. http://dx.doi.org/10.1109/CDC.2018.8619088.

[12] Argáez C, Giesl P, Hafstein SF. Complete Lyapunov functions: Determination of the chain-recurrent set using the gradient. In: Obaidat M, Ören T, H. Szczerbicka, editors. Simulation and modeling methodologies, technologies and applications. SIMULTECH. Advances in intelligent systems and computing, vol. 1260, Cham: Springer; 2019, http://dx.doi.org/10.1007/978-3-030-55867-3_6.

[13] Giesl P. Construction of global lyapunov functions using radial basis functions. Lecture notes in mathematics, vol. 1904, Berlin Heidelberg: Springer-Verlag; 2007, https://link.springer.com/book/10.1007%2F978-3-540-69909-5.

[14] Wendl H. Error estimates for interpolation by compactly supported Radial Basis Functions of minimal degree. J Approx Theory 1998;93:258–72. http://dx.doi.org/10.1006/jath.1997.3137.

[15] Iske A. Perfect centre placement for radial basis function methods. Technical report TUM M9809, Technische Universität München; 1998.

[16] Bjornsson H, Hafstein S. Advanced algorithm for interpolation with wendland functions. In: Informatics in control, automation and robotics (LNEE). Springer; 2021, [in press].

[17] Argáez C, Giesl P, Hafstein S. Statistical analyses of an iterative algorithm class for dynamical systems, [submitted for publication].

[18] Argáez C, Giesl P, Hafstein S. Iterative construction of complete Lyapunov functions: Analysis of algorithm efficiency. In: Obaidat M, Ören T, Rango F, editors. Simulation and modeling methodologies, technologies and applications. SIMULTECH 2018. Advances in intelligent systems and computing, Advances in intelligent systems and computing, 2020;vol. 947.http://dx.doi.org/10.1007/978-3-030-35944-7_5,

[19] Sanderson C, Curtin R. Armadillo: a template-based C++ library for linear algebra. J Open Source Softw 2016;1:26. http://dx.doi.org/10.21105/joss.00026.

[20] Sanderson C, Curtin R. A user-friendly hybrid sparse matrix class in C++. Lecture Notes in Comput Sci 2018;10931:422–30. http://dx.doi.org/10.1007/978-3-319-96418-8_50.