JuliaDynamics - Open source and educative software for dynamical systems, nonlinear dynamics and chaos

George Datseris[1,2]

[1]Max Planck Institute for Dynamics and Self-Organization
[2]Georg-August-Universität Göttingen

February 28, 2019

George Datseris is a PhD student (candidate), supervised by Prof. Theo Geisel and Dr. Ragnar Fleischmann. He is the lead developer of the JuliaDynamics GitHub organization, whose development was done by George Datseris independently of the PhD Thesis project. Thus the lead developer is a grad student.

1 Introduction

With this document we would like to enter the competition of DSWeb 2019 "Tutorials on Dynamical Systems Software". Our entry represents the GitHub organization JuliaDynamics and more specifically the various tutorials we have produced to guide our users into not only being able to use the packages we provide, but also to be able to understand and study dynamical systems in depth.

The core packages we will be discussing in the rest of the application are:

1. **DynamicalSystems.jl**, which is a Julia software library for studying nonlinear dynamical systems and chaos.
2. `InteractiveChaos`, which is an extension to **DynamicalSystems.jl** that provides interactive applications for exploring chaos.
3. **DynamicalBilliards.jl**, which is a package for simulating billiard systems in two dimensions.

In the rest of the text we will shortly overview each package and point out how it contributes into this competition entry.

1.1 Versions

This application describes the following software versions (or latest):

1. **DynamicalSystems.jl** - v1.2.0
2. `InteractiveChaos` - v0.3.0
3. **DynamicalBilliards.jl** - v3.0.0

1.2 Installation

Provided that you have already installed the Julia language (version 1.0 or greater)[1] you only need to press `]` to access the package manager and then simply type

[1] https://julialang.org/downloads/

```
add DynamicalSystems InteractiveChaos DynamicalBilliards
add Makie
```

which will automatically install all the dependencies (i.e. other Julia packages) necessary. The final add command installs a plotting backend for the package `InteractiveChaos`.

1.3 Intended Audience

Intented audience of JuliaDynamics is anyone interested in dynamical systems, nonlinear dynamics or chaotic behavior, be they students, educators, researchers or simply curious individuals.

2 DynamicalSystems.jl

DynamicalSystems.jl [3] is a Julia [1] software library for the exploration of chaos and nonlinear dynamics that was built from the ground up based on the principles of generality, clarity and robustness. It is composed of multiple sub-packages and its functionality is too lengthy to discuss in detail in this paper. Notable features are calculations of the Lyapunov spectrum for any system, general delay coordinate embeddings, generalized entropies and dimensions, recurrence quantification analysis and much more. For the full list please visit the official documentation page.

DynamicalSystems.jl also has great applications in education. Because the syntax is intuitive and concise, scripts that produce informative demonstrations are short and understandable. Also, since the source code is clear and small, students can simply read it and understand how to e.g. apply and modify an algorithm for specialized applications. Lastly, we always discuss in detail the algorithm's use and always cite relevant scientific articles that introduce and apply the algorithm that we use in the source code.

For the above reasons, we believe that the official documentation page by itself is a quality tutorial on **DynamicalSystems.jl**. Besides the citations and explanations the documentation is also full of real world applications, examples, and even caveat demonstrations. A simple example is the documentation page where we explain a method to numerically compute the maximum Lyapunov exponent (see here). There we show an explicit example where naively choosing parameters for densely sampled data could lead to wrong results. Similar educative examples can be found all over the official documentation. We also point out that all code snippets featured in the documentation are true runnable examples and in fact the documentation is generated automatically by running those examples with every new commit pushed to the GitHub repository. This ensures robustness in the documentation.

Besides written text however, there are videos to guide users using **DynamicalSystems.jl**. Specifically, there is a 2-hour video tutorial hosted on the official YouTube channel of the Julia language. It can be found here. In similar spirit, this tutorial explains in depth how to use the code but also offers educative examples for all functionality introduced. In addition, significant effort is spent into explaining the introduced concepts from a scientific perspective (e.g. "what is a Lyapunov exponent and why is it useful?").

To summarize, **DynamicalSystems.jl** brings these points into our application for the DSWeb competition:

1. Its official documentation page.
2. A video tutorial hosted on the official Julia language YouTube channel.

3 InteractiveChaos

`InteractiveChaos` is a new Julia package (currently still in beta) that offers interactive applications for exploring chaotic systems. This package is based on the **DynamicalSystems.jl** framework, which means that all of its applications work for any possible dynamical system. For each application described in this section there is a short `.mp4` video that shows its basic features quickly. This video is directly below the application's documentation page, which we link in each paragraph separately. Besides this however some interactive applications are accompanied by a lengthy (typically 10-15 minutes) video explaining their use and pointing out a case of scientific relevance. We note that even though at the current date of entering in the DSWeb competition

`InteractiveChaos` has three applications, in the future it will be populated by more, as there is already work underway for two new applications.

The first application of `InteractiveChaos` is an interactive orbit diagram. The orbit diagram, also commonly known as bifurcation diagram, is a way to visualize the long term evolution of a discrete system. The application allows one to zoom into an orbit diagram or update other parameters like transient steps interactively. The application works for any discrete system and for any variable of the system and its documentation can be found here. In addition the application has a history, which allows users to move back through the steps that led them to their final position. We stress that the application is plotting a full, true orbit diagram, not only a shorthand that "looks like" an orbit diagram. All data are immediately accessible by the user. The video showcasing the application can be seen here.

The second application is an interactive Poincaré surface of section (PSOS) whose documentation is here. The PSOS is a technique to reduce a continuous dynamical system to a discrete system of one less dimensionality, by recording the state of the continuous system whenever it crosses a well-defined hyper-plane. For any continuous dynamical system the application allows one to interactively explore its PSOS. Clicking on any part of the plot will start a new initial condition, integrate it, compute its PSOS and add it to the shown scatter plot. Besides this some basic scaling sliders are available, to make it easy to zoom in and out of the plot. Lastly, it is possible to color-code any initial condition based on a given user defined function (for example one could color the points according to the value of their Lyapunov exponent, or the absolute value of the second coordinate, etc.). A video showcasing the application can be seen here.

The third application is an interactive "trajectory highlighter", whose documentation can be found here. This application takes as an input a vector of datasets and a vector of corresponding values. These could be anything, for example a vector of trajectories and their corresponding Lyapunov exponents, or a vector of PSOS and their corresponding GALI value or the value of the energy associated with each trajectory (GALI is a measure for the regularity of a trajectory, for more details see the documentation of the GALI function). The application will then produce a histogram of the values and color each associated dataset with the value corresponding to the histogram color. Then two plots will show up, one showing the datasets while the other showing the histogram. Clicking on any histogram bin will highlight all datasets associated with the bin's value. This will also *hide* all other datasets *not* associated with the click value. The same will happen if one clicks the plot of the datasets; the clicked dataset will be highlighted while the others will be hidden.

To summarize, `InteractiveChaos` brings these points into our application for the DSWeb competition:

1. An interactive application for exploring orbit diagrams of any discrete dynamical system.
2. An interactive application for exploring PSOS of continuous systems.
3. An interactive application for highlighting properties of trajectories and the trajectories themselves.
4. A video tutorial for the interactive orbit diagram, highlighting the period doubling route to chaos.
5. A video tutorial for the interactive PSOS, highlighting the "Hamiltonian route to chaos", i.e. the simultaneous application of the KAM and Poincaré-Birkhoff theorem.

4 DynamicalBilliards.jl

DynamicalBilliards.jl [2] is another Julia package, part of the JuliaDynamics GitHub organization, that offers computer code to simulate billiard systems in two dimensions. Its features are too many to discuss here, but for example include modular creation of a billiard, ray-splitting, magnetic propagation, Lyapunov exponents and more. The official documentation page can be found here, which also includes a detailed list of features.

The biggest strength of **DynamicalBilliards.jl** is the fact that the high level API is extremely concise yet powerful. With only 5 lines of code one can obtain an animation of particles moving around in a billiard and see how chaotic behavior emerges out of a parallel ray of particles. For example, this simple code snippet:

```
using DynamicalBilliards, PyPlot

bd = billiard_stadium()
```

```
cs = [(i/N, 0, 1 - i/N, 0.5) for i in 1:20]
ps = [Particle(1, 0.6 + 0.0005*i, 0) for i in 1:20]

animate_evolution(ps, bd, 7.0; colors = cs, tailtime = 1.5)
```

will produce an animation showing how chaos arises in a focusing billiard (here the Bunimovich stadium). To see the animation without running the code, please download this file.

The official documentation page of **DynamicalBilliards.jl** operates similarly with **DynamicalSystems.jl**. It is extensive, full of examples and runnable code, and is produced automatically with every new commit pushed to the GitHub repository. The documentation also has tutorials that target a specific process, like for example creating a billiard with arbitrary shape. Finally there is a detailed overview of the animation framework offered by **DynamicalBilliards.jl** (and showcased in the code snippet above). This framework allows one to not only see the animations but also conveniently save them as video files for later use.

Lastly, there is one more resource that is a tutorial for **DynamicalBilliards.jl**, namely an interactive article in the newly established platform "NextJournal". The article can be found here, but it is also available as a Jupyter notebook here. The article tries to guide the reader through the thought process and implementation of a code base that can evolve any particle in any billiard. It is an educative demonstration both for understanding how the **DynamicalBilliards.jl** package works but also how to take full advantage of Julia's Multiple Dispatch feature.

To summarize, **DynamicalBilliards.jl** brings these points into our application for the DSWeb competition:

1. Its official documentation page.
2. Its animation framework, which allows one to display (on the fly) videos of how particles (or even light rays) move in billiards.
3. The interactive article that demonstrates how one can implement a dynamical billiard software in an intuitive way.

5 Acknowledgements

The author would like to thank and acknowledge the following contributions

1. Lukas Hupe has significantly improved the animation functionality of **DynamicalBilliards.jl**.
2. Sebastian Micluța-Câmpeanu developed the third interactive application of `InteractiveChaos`, namely the "trajectory highlighter".

References

[1] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Shah. Julia: A Fresh Approach to Numerical Computing. *SIAM Review*, 59(1):65–98, jan 2017.

[2] George Datseris. DynamicalBilliards.jl: An easy-to-use, modular and extendable julia package for dynamical billiard systems in two dimensions. *The Journal of Open Source Software*, 2(19):458, nov 2017.

[3] George Datseris. DynamicalSystems.jl: A julia software library for chaos and nonlinear dynamics. *Journal of Open Source Software*, 3(23):598, mar 2018.