Spyci Documentation

Release 1.0.2

Gonçalo Magno

Jan 21, 2022

Contents:

1 Spyci **1**
- 1.1 Getting Started . . . 1
- 1.2 Installation . . . 1

2 Installation **7**
- 2.1 Stable release . . . 7
- 2.2 From sources . . . 7

3 Usage **9**

4 spyci **11**
- 4.1 spyci package . . . 11

5 Contributing **13**
- 5.1 Types of Contributions . . . 13
- 5.2 Get Started! . . . 14
- 5.3 Pull Request Guidelines . . . 15
- 5.4 Tips . . . 15
- 5.5 Deploying . . . 15

6 Credits **17**
- 6.1 Development Lead . . . 17
- 6.2 Contributors . . . 17

7 History **19**
- 7.1 0.6.2 (2020-06-02) . . . 19

8 Indices and tables **21**

Python Module Index **23**

Index **25**

CHAPTER 1

Spyci

A tiny Python package to parse and plot spice raw data files.

- Free software: MIT license
- Documentation: https://spyci.readthedocs.io.

1.1 Getting Started

These instructions will get you a copy of the package up and running on your local machine.

Note: at the moment only ascii raw spice format is supported!

1.2 Installation

1.2.1 From PyPI

```
$ pip install spyci  # it is recommended to this in a virtual environment
```

1.2.2 From the github repo

```
$ pip install git+https://github.com/gmagno/spyci.git
```

or

```
$ git clone git@github.com:gmagno/spyci.git
$ cd spyci/
$ python setup.py install
```

1.2.3 Usage

From python run:

```
>>> from spyci import spyci
>>> data = spyci.load_raw("/path/to/rawspice.raw")  # see 'Data structure' section␣
↪below
```

Or just use the CLI:

```
$ spyci -r /path/to/rawspice.raw vin vout
```

for more details use:

```
$ spyci -h
usage: spyci [-h] [-v] [-r RAW_FILE] [-l] [-f] [-o OUT_IMAGE] ...

Spyci (spyci v0.6.1) -- parses ngspice raw data files and
plots the specified variables.
For full documentation check the repo: https://github.com/gmagno/spyci

positional arguments:
  VARS                  List of variables to plot

optional arguments:
  -h, --help            show this help message and exit
  -v, --version         shows spyci version
  -r RAW_FILE, --raw-file RAW_FILE
                        path to raw file to be parsed
  -l, --list-variables  lists variables that can be plotted
  -f, --out-formats     lists supported output image formats
  -o OUT_IMAGE, --out-image OUT_IMAGE
                        path to output image file, use -f, to list supported
                        formats

                                                  /##
                                                 |__/
          /#######  /######  /##   /##  /####### /##
         /##_____/ /##__  ##| ##  | ## /##_____/| ##
        |  ###### | ##  \ ##| ##  | ##| ##      | ##
         \____  ##| ##  | ##| ##  | ##| ##      | ##
         /#######/| #######/|  #######|  #######| ##
        |_______/ | ##____/  \____  ## \_______/|__/
                  | ##       /##  | ##
                  | ##      |  ######/
                  |__/       \______/

return:
    The return value of spyci is 0 if the raw file is successfully
    parsed and plotted.

examples:
```

(continues on next page)

(continued from previous page)

```
    # Run without arguments will attempt to load rawspice.raw from cwd
    # and plot all variables
    $ spyci

    # List variables that can be plotted
    $ spyci -l
    Variables:

    idx  name        type
    -----  ----------  -------
        1  i(l1)       current
        2  n1          voltage
        3  vi          voltage
        4  vo          voltage
        5  i(vsource)  current

    # Load 'some/location/sim.raw' and plot variables 'i(l1)' and 'vo'
    $ spyci -r some/location/sim.raw "i(l1)" vo

    # Indices can be used insted of variable names, this is equivalent
    # to the previous example
    $ spyci -r some/location/sim.raw 1 4

    # Save your plot to the file system
    $ spyci -o myplot.png 1 4

    # Different image formats are supported, just use the correct
    # extension, {.png, .svg, .pdf, ...}. For a list of supported
    # formats run with -f flag
    $ spyci -f
    Supported output image file formats:

    ext    format
    -----  -------------------------
    raw    Raw RGBA bitmap
    rgba   Raw RGBA bitmap
    pgf    PGF code for LaTeX
    svgz   Scalable Vector Graphics
    svg    Scalable Vector Graphics
    ps     Postscript
    png    Portable Network Graphics
    eps    Encapsulated Postscript
    pdf    Portable Document Format

copyright:
    Copyright © 2020 Gonçalo Magno <goncalo@gmagno.dev>
    This software is licensed under the MIT License.
```

1.2.4 Data structure

A properly parsed raw spice file by *load_raw()* returns a dictionary with the following structure:

```
{
    "title": <str>,
    "date:": <str>,
```

(continues on next page)

(continued from previous page)

```
    "plotname:": <str>,
    "flags:": <str>,
    "no_vars:": <str>,
    "no_points:": <str>,
    "vars": [
        { "idx": <int>, "name": <str>, "type": <str> },
        { "idx": <int>, "name": <str>, "type": <str> }
        ...
        { "idx": <int>, "name": <str>, "type": <str> }
    ]
    "values": {
        "var1": <numpy.ndarray>,
        "var2": <numpy.ndarray>,
        ...
        "varN": <numpy.ndarray>
    }
}
```

Where values *values* is a numpy structured array with the actual data.

1.2.5 Examples

The following examples make use of ngspice to run the spice simulations, so please ensure it is installed. On ubuntu that would be:

```
$ sudo apt install ngspice
```

1.2.6 Inverting amplifier with an opamp LM741

Check the directory *examples/amplifier/* for details on the cirtcuit and the simulation files.

The schematic:

Run the simulation with:

```
$ cd examples/amplifier
$ ngspice -r rawspice.raw -o output.log main.cir
$ spyci vout vin
```

which will fire ngspice generating output.log and rawspice.raw files and also plots the voltages *vin* and *vout*.

1.2.7 Second order low pass filter with an opamp LM741

Check the directory *examples/lp_filter/* for details on the cirtcuit and the simulation files.

The schematic:

Run the simulation with:

```
$ cd examples/lp_filter
$ ngspice -r rawspice.raw -o output.log main.cir
$ spyci vout vin
```

which will fire ngspice generating output.log and rawspice.raw files and also plots the gain *vout*/*vin* in dB.

1.2.8 License

This project is licensed under the MIT License - see the LICENSE file for details

CHAPTER 2

Installation

2.1 Stable release

To install Spyci, run this command in your terminal:

```
$ pip install spyci
```

This is the preferred method to install Spyci, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for Spyci can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/gmagno/spyci
```

Or download the tarball:

```
$ curl -OJL https://github.com/gmagno/spyci/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

To use Spyci in a project:

```
import spyci
```

CHAPTER 4

spyci

4.1 spyci package

4.1.1 Submodules

4.1.2 spyci.cli module

4.1.3 spyci.spyci module

4.1.4 Module contents

Top-level package for Spice Raw Parser.

CHAPTER 5

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/gmagno/spyci/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

5.1.4 Write Documentation

Spyci could always use more documentation, whether as part of the official Spyci docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/gmagno/spyci/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *spyci* for local development.

1. Fork the *spyci* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/spyci.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv spyci
$ cd spyci/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 spyci tests
$ python setup.py test or pytest
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.5, 3.6, 3.7 and 3.8, and for PyPy. Check https://travis-ci.com/gmagno/spyci/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ pytest tests.test_spyci
```

5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bump2version patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER 6

Credits

6.1 Development Lead

- Gonçalo Magno <goncalo@gmagno.dev>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

History

7.1 0.6.2 (2020-06-02)

- Fix project.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

spyci, 11

Index

S

`spyci` (*module*), 11