MR SPRINT Documentation

Release 1.2.2

Daniel Cosmo Pizetta, Victor Hugo de Mello Pessoa

Jul 04, 2018

Contents


MR SPRINT

CHAPTER 1

Welcome!

Magnetic resonance experiment simulator and visualization tool

MRSPRINT is a visual magnetic resonance simulator where you can simulate a magnetic resonance experiment - spectroscopy or imaging. Its main goal is to be an education tool that assists the student/staff to understand, interpret and explore magnetic resonance phenomena.

This tool is totally free (see license), but if you are making use of it, you need to cite us using both citations. This is very important for us.

- Article: Coming soon!
- Software: Coming soon!

If you are using this piece of software to generate images, gif's, movies, etc., or using images available on this site, please, also reference us using those citations.

1.1 What can you see?

- Precession: spins precessing in static magnetic field;
- Resonance: resonance when a RF pulse is applied;
- Contrast: T1, T2, and density of spins;
- Field inhomogeneity: isochromates can be shown with their dispersion;
- Gradient: magnetic field gradient in action, its intensity and effect over the frequency;
- Evolution: over magnetization with intensity, frequency and phase and the pulse sequence;
- FID: free induction decay, the signal;
- Echo: spin or gradient echo (rephasing/dephasing).

1.2 Future planned features

- K-space visualization;
- Multi-nuclei samples/experiments;

- T2* as sample parameter to easily setup field inhomogeneity;
- Graphical sequence editor;
- Chemical interactions;
- View on coordinate laboratory system;
- Flow (spins not fixed in positions).

1.3 Documentation

Go to documentation on ReadTheDocs! It is available on Read The Docs in HTML, EPUB, and PDF.

1.4 Download binaries - click-and-run

Binaries are for those do not wish to install any Python things. We recommend them to the ones without any programming experience. Download from links below.

- Portable Windows Binaries: coming soon!
- Portable Linux Binaries: coming soon!
- Portable Mac Binaries: coming soon!

Sou you can just download, decompress, click-and-run.

1.5 Installing from PyPI - stable, end-user

To install, do

```
$ pip install mrsprint
```

This will install all necessary dependencies then the code.

To run from terminal

```
$ mrsprint
```

1.6 Dependencies

- NumPy: Numerical mathematical library;
- SciPy: Scientific library;
- NMRGlue: NMR processing library;
- H5Py: Storing and managing data files;
- PyQtGraph: Data visualization library;
- PyQt/Pyside: Graphical framework.

Dependencies are automatically installed when using the method above.

CHAPTER 2

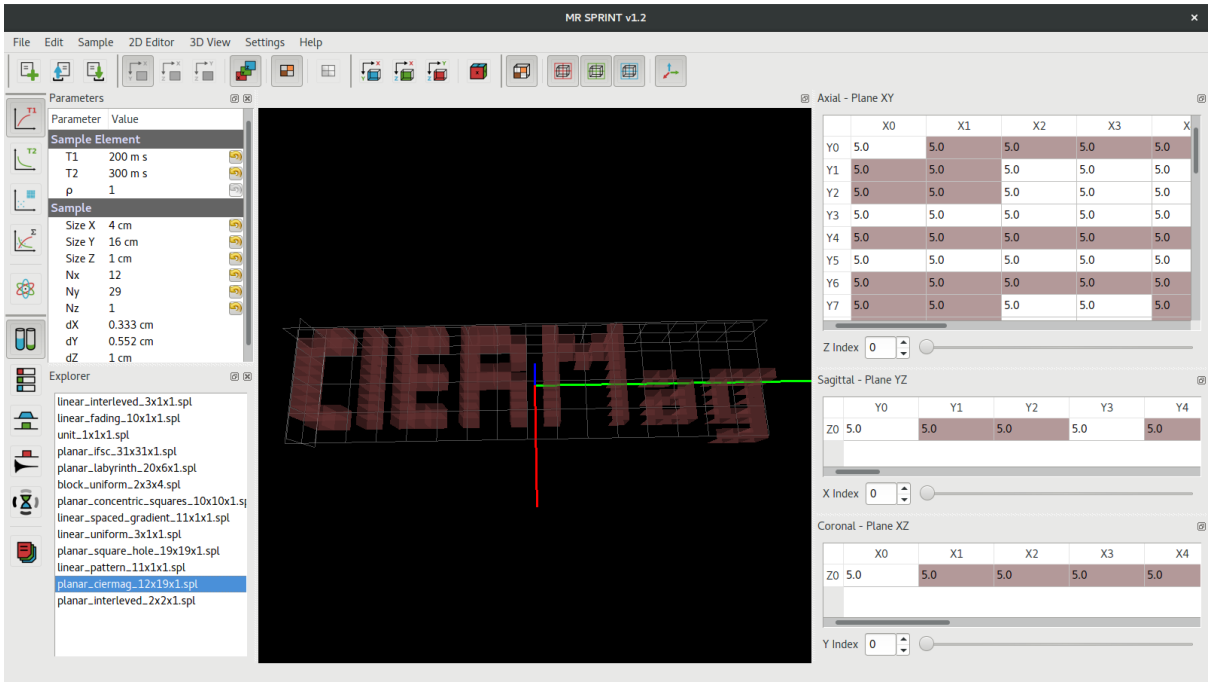
Overview

2.1 Main screens

Context editors are specialized to edit each step of experiment, see below.

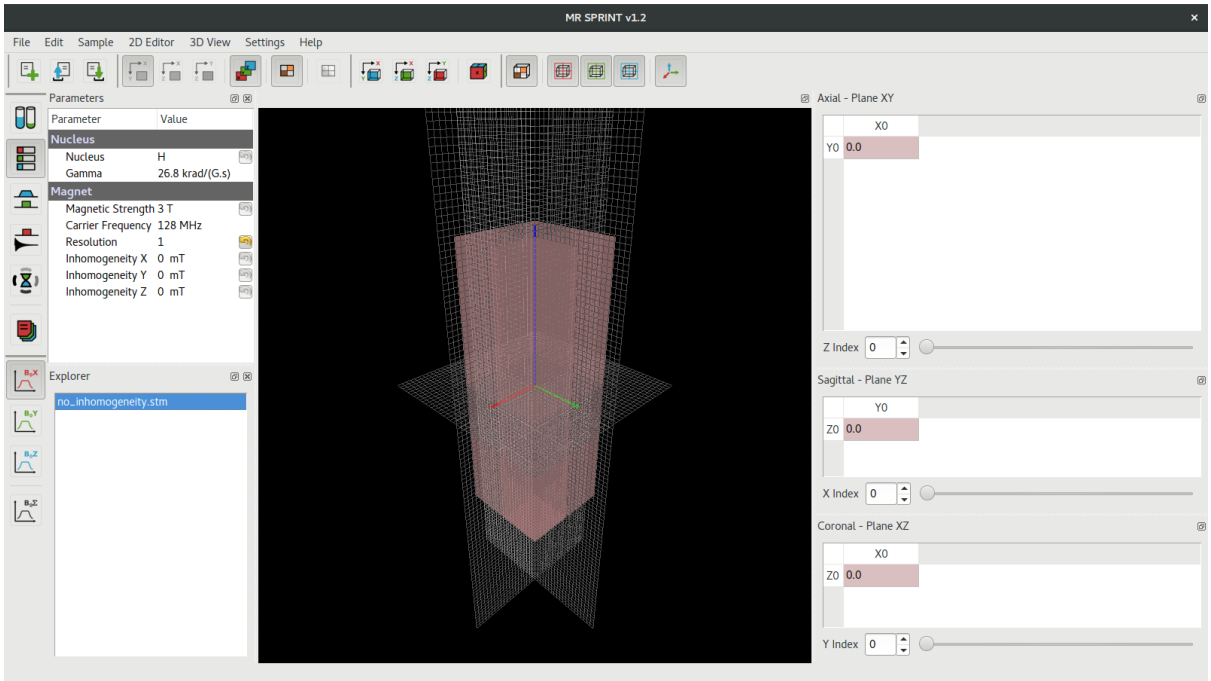
2.1.1 Sample

You can open and/or create samples to run with MR experiment. Each sample element have three characteristics: t1,t2 and density of spins.



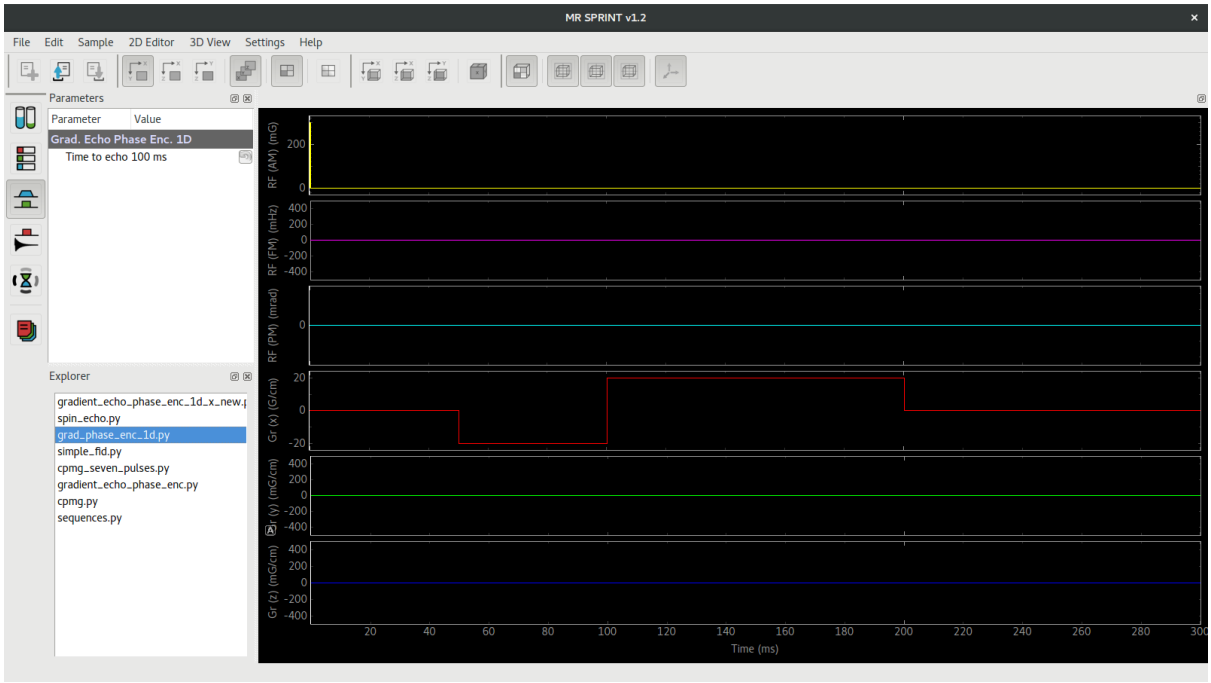
2.1.2 System

Here you can set static magnetic field and add inhomogeneity to it.



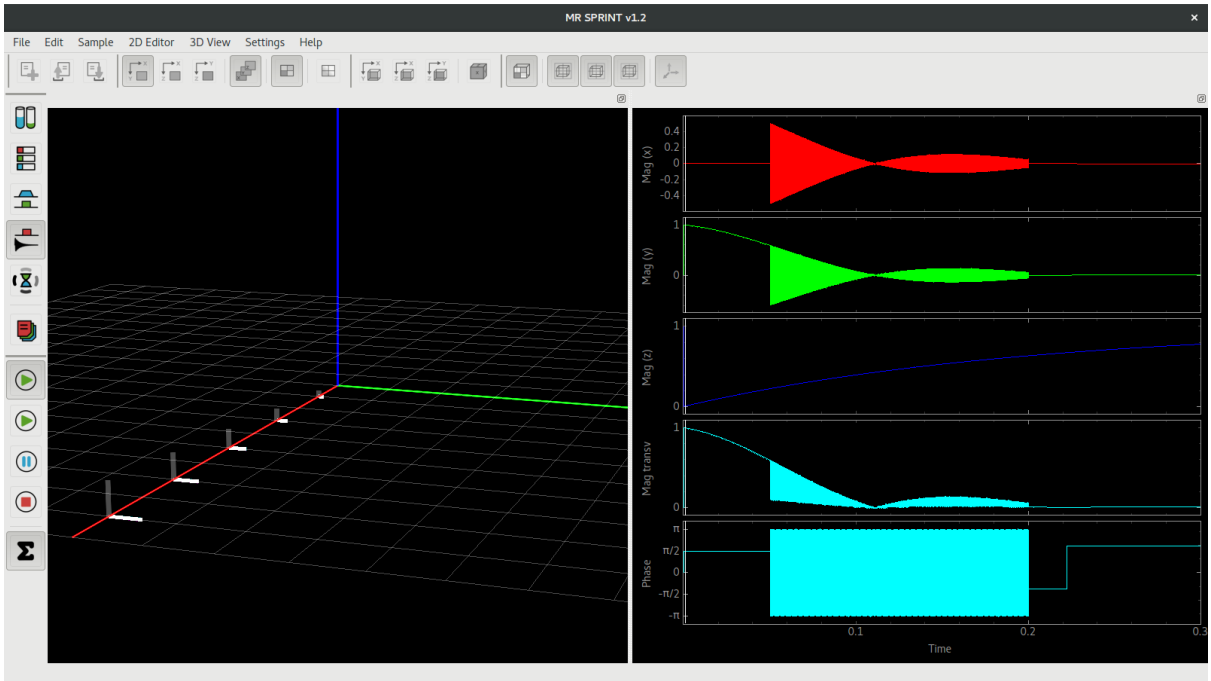
2.1.3 Sequence

You can choose a pulse sequence for your experiment, that includes the RF and gradient pulses. The sequence is programmed in Python at this moment.



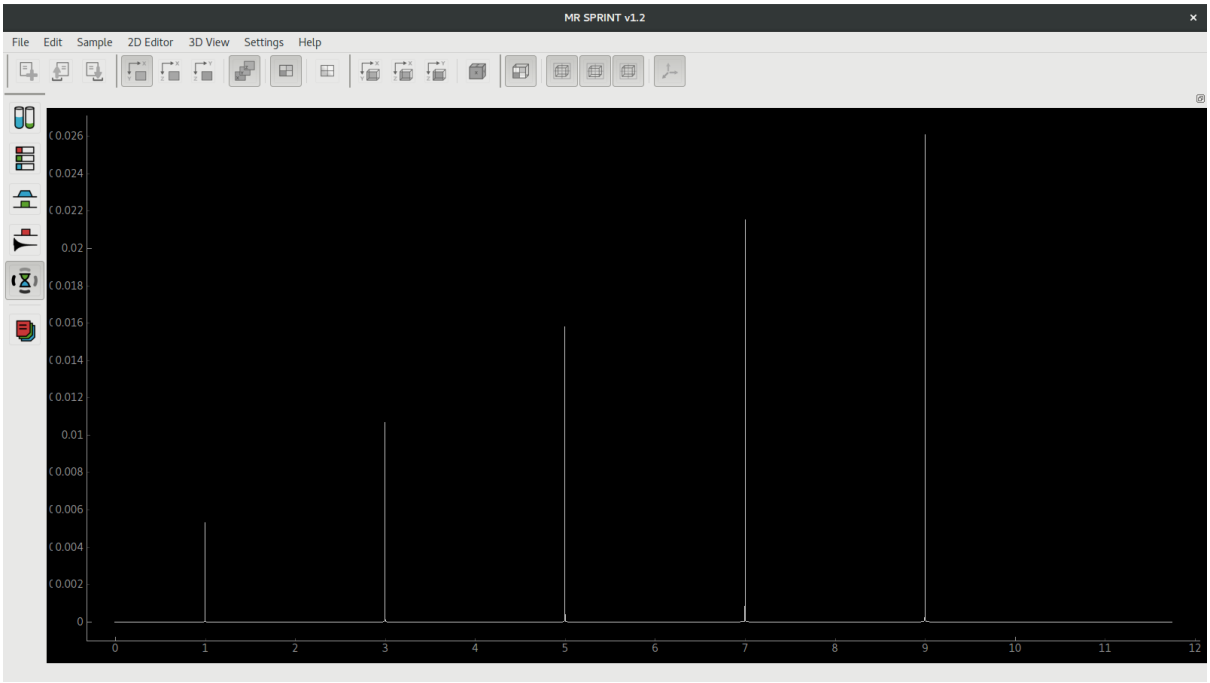
2.1.4 Simulator

At this point you can set simulation mode and other details about simulation , e.g. time resolution.



2.1.5 Processing

Finally you can process your data to plot the spectrum, imaging or other features.



2.2 2D Editor

This editor provides a table that represents the current selected slice of the 3D view. In this table you can edit the values of each element property(ies). Colors also help you to pre visualize the intensity of chosen value.

2.3 3D View

3D view is used to show contexts objects such as sample, magnet field inhomogeneity, and the evolution of magnetization. You can move the cam to adjust the perspective.

2.4 Data

Data can be saved using HDF5 files. For each context we adopted an extension to distinguish from each other. HDF5 files can be easily modified with other external tool if necessary.

CHAPTER 3

History

3.1 A brief history of this universe

Being short, when difficulties arises at understanding magnetic resonance phenomena, we should search tools that could help us.

At that time we have found nice explanations on Brian site.

As we are fan of Python, we discovered this implementation from Neji.

Yet, it was difficult to change parameters and create new graphics. So, this work starts. From that, many ideas have shining, and, here we are.

The main developer/maintainer is the PhD in computational physics Daniel C. Pizetta, at University of São Paulo, São Carlos, Brazil.

A lot of the code was developed by the bachelor, in physics and biomolecular sciences, Victor H.M. Pessoa in his final paper. He created many new functionalities, implemented all file interactions and connections between graphical interface and core code.

From now on, Clara Vidor is continuing the development, and we hope that will still new features to be released soon.

We also have a head professor PhD Fernando Fernandes Paiva that provides a technical support in magnetic resonance issues.

CHAPTER 4

Reference

4.1 examples

4.2 mrsprint

4.2.1 mrsprint package

Subpackages

mrsprint.gui package

Submodules

mrsprint.gui.mrsprint_rc module

mrsprint.gui.mw_gradient module

mrsprint.gui.mw_mrsprint module

mrsprint.gui.mw_settings module

Module contents

Package for GUI related classes and objects.

Authors:

- Victor Hugo de Mello Pessoa <victor.pessoa@usp.br>
- Daniel Cosmo Pizetta <daniel.pizetta@usp.br>

Since: 2017/01/09

mrsprint.sequence package

Submodules

mrsprint.sequence.protocol module

mrsprint.sequence.sequence module

Module contents

Package for sequence related classes and objects.

Authors:

- Daniel Cosmo Pizetta <daniel.pizetta@usp.br>

Authors: 2015/11/01

Todo: Fix these modules.

---

mrsprint.simulator package

Submodules

mrsprint.simulator.old_plot module

mrsprint.simulator.plot module

mrsprint.simulator.simulator module

Module contents

Package for simulator related classes and objects.

Authors:

- Daniel Cosmo Pizetta <daniel.pizetta@usp.br>

Since: 2018/06/07

Todo: Maybe remove old_plot module.

`mrsprint.simulator.`**`calculate_t2_star`**(*t2*, *freq_shift*)

Returns the value for t2*, considering the range of frequencies.

Parameters

- **t2** (`float [s]`) – T2 value in seconds.
- **freq_shift** (`float [Hz]`) – Frequency shift from resonance - symetric between zero (in resonance).

Returns T2 star value.

Return type float [s]

Todo: Confirm if it is the right way to calculate or if it has a weight function for t2star Pass freq_shift as a number?

mrsprint.simulator.**create_positions**(*size=(1, 1, 1)*, *step=(1.0, 1.0, 1.0)*, *offset=(0.0, 0.0, 0.0)*, *dtype=<class 'numpy.float32'>*)

Creates array of positions.

Parameters

- **size** (tuple(int)) – Size in x, y and z. The minimum number is one for each axis.
- **step** (tuple(float)) – Step for each axis.
- **offset** (tuple(float)) – Offset for each axis. If zero, the final vector begins in zero and ends in size plus offset.

Returns

Position array in this format [[gr_x_plotx gr_y_plotx gr_z_plotx . . .]
[gr_x_ploty gr_y_ploty gr_z_ploty . . .][gr_x_plotz gr_y_plotz gr_z_plotz . . .]], where p is the number of position (from offset to size plus offset).

Return type np.array

Todo: Set physical unit for each args.

mrsprint.simulator.**frequency_shift**(*freq_shift*, *freq_step=1.0*, *offset=0.0*, *symetric=True*, *dtype=<class 'numpy.float32'>*)

Generates an array of frequency shift, from -frequency_shift to +frequency_shift, between offset.

Parameters

- **freq_shift** (float [Hz]) – Maximum frequency to shift.
- **freq_step** (float [Hz]) – Spacing between values.
- **offset** (float [Hz]) – Offset frequency.
- **symetric** (bool) – If true, generates the array between offset (-maximum shift, maximum shift), otherwise from offset to maximum frequency. Default is True.
- **dtype** (np.dtype) – Data type for the array. Default is np.float32.

Returns Array of frequency shift value.

Return type np.array [Hz]

Todo: Use the key symetric for something.

mrsprint.simulator.**reduce_magnetization_in_frequency**(*mx*, *my*, *mz*, *freq_shift*, *fsa_size*)

Reduces the magnetization vector by summing frequency components.

Parameters

- **mx** (np.array) – Array of magnetization x without reduction in frequency.
- **my** (np.array) – Array of magnetization y without reduction in frequency.
- **mz** (np.array) – Array of magnetization z without reduction in frequency.
- **freq_shift** (np.array) – Array of frequency shift.
- **fsa_size** (int) – Size of array of frequency shift.

Returns (mx, my, mz) arrays of magnetization reduced

Return type tuple

Todo: Shape size different of 2. Better solution for freq_shift array. Set physical unit for each args.

---

mrsprint.simulator.**reduce_magnetization_in_position**(*mx*, *my*, *mz*, *position*, *freq_shift*)
Reduces the magnetization vector by summing position components.

Parameters

- **mx** (np.array) – Array of x magnetization without reduction in frequency.
- **my** (np.array) – Array of y magnetization without reduction in frequency.
- **mz** (np.array) – Array of z magnetization without reduction in frequency.
- **position** (np.array) – Array of positions.
- **freq_shift** (np.array) – Array of frequency shift.

Returns (mx, my, mz) arrays of magnetization reduced.

Return type tuple

Todo: Test shape size different of 2. Set physical unit for each args.

---

mrsprint.simulator.**transform_cart_to_pol**(*x*, *y*)
Returns a transformed catesian vector into polar coordinates.

Parameters

- **x** (np.array) – X coordinate.
- **y** (np.array) – Y coordinate.

Returns (rho, phi) vectors formed by radial and angular coordinates.

Return type tuple

mrsprint.simulator.**transform_pol_to_cart**(*rho*, *phi*)
Returns a transformed polar vector into cartesian coordinates.

Parameters

- **rho** (np.array) – Radial coordinate.
- **phi** (np.array) – Angular coordinate.

Returns (x, y) vectors formed by x and y coordinates

Return type tuple

mrsprint.subject package

Submodules

mrsprint.subject.sample module

Module contents

Package for subject related classes and objects.

Authors:

- Daniel Cosmo Pizetta <daniel.pizetta@usp.br>

Since: 2017/10/01

mrsprint.system package

Submodules

mrsprint.system.gradient module

mrsprint.system.magnet module

mrsprint.system.rf module

Module contents

Package for system related classes and objects.

Authors:

- Daniel Cosmo Pizetta <daniel.pizetta@usp.br>

Since: 2015/11/01

Submodules

mrsprint.globals module

Global values.

Authors:

- Victor Hugo de Mello Pessoa <victor.pessoa@usp.br>
- Daniel Cosmo Pizetta <daniel.pizetta@usp.br>

Since: 2015/06/01

mrsprint.mainwindow module

mrsprint.settings module

Module contents

Magnetic resonance experiment simulator and visualization tool.

Authors:

- Daniel Cosmo Pizetta <daniel.pizetta@usp.br>
- Victor Hugo de Mello Pessoa <victor.pessoa@usp.br>

Since: 2015/07/01

4.3 scripts

4.3.1 generate_qrc module

4.3.2 get_version module

4.3.3 process_icons module

4.3.4 process_ui module

CHAPTER 5

Downloads

Here you will find documents and files to download.

5.1 Download binaries - click-and-run

Binaries are for those do not wish to install any Python things. We recommend them to the ones without any programming experience. Download from links below.

- Portable Windows Binaries: coming soon!
- Portable Linux Binaries: coming soon!
- Portable Mac Binaries: coming soon!

Sou you can just download, decompress, click-and-run.

5.2 Documentation

Links for latest available documentation.

CHAPTER 6

Changelog

6.1 v1.2

- Add file explorer
- Add about
- Add docs to ReadTheDocs
- Improved and new icons, and logo
- Bug fixes

6.2 v1.1

- Fix docs, examples

6.3 v1.0

- First public API

6.4 v0.6

- New structure, revised files and screenshots
- Fix doc style
- Change nucleous to nucleus
- Fix pyqtgraph imports

6.5 v0.5

- Add new extensions for HDF5 files for contexts
- More examples of samples
- Add simulation context
- Bug fixes

6.6 v0.4

- Add sample examples
- Bug fixes
- Improvements on 2D editor and 3D plot

6.7 v0.3

- UI Improvements
- Add git-lab CI
- Add pylint
- Open and save samples in HDF5
- Bug fixes

6.8 v0.2

- Removing unnecessary things
- Improvements in code
- Docs with Sphinx

6.9 v0.1

- First version with binaries for Windows and Linux - Pyinstaller
- Add tox
- Add scripts for pyinstaller, ui, icons
- 2D editor and 3D view
- Main window and toolbars

CHAPTER 7

Authors

- Daniel Cosmo Pizetta
- Victor Hugo de Mello Pessoa

CHAPTER 8

License

8.1 Code - The MIT License

8.2 Images - Creative Commons Attribution International 4.0

8.2.1 Using Creative Commons Public Licenses

Creative Commons public licenses provide a standard set of terms and conditions that creators and other rights holders may use to share original works of authorship and other material subject to copyright and

certain other rights specified in the public license below. The following considerations are for informational purposes only, are not exhaustive, and do not form part of our licenses.

- **Considerations for licensors:** Our public licenses are intended for use by those authorized to give the public permission to use material in ways otherwise restricted by copyright and certain other rights. Our licenses are irrevocable. Licensors should read and understand the terms and conditions of the license they choose before applying it. Licensors should also secure all rights necessary before applying our licenses so that the public can reuse the material as expected. Licensors should clearly mark any material not subject to the license. This includes other CC-licensed material, or material used under an exception or limitation to copyright. More considerations for licensors.
- **Considerations for the public:** By using one of our public licenses, a licensor grants the public permission to use the licensed material under specified terms and conditions. If the licensor's permission is not necessary for any reason–for example, because of any applicable exception or limitation to copyright–then that use is not regulated by the license. Our licenses grant only permissions under copyright and certain other rights that a licensor has authority to grant. Use of the licensed material may still be restricted for other reasons, including because others have copyright or other rights in the material. A licensor may make special requests, such as asking that all changes be marked or described. Although not required by our licenses, you are encouraged to respect those requests where reasonable. More considerations for the public.

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

Python Module Index

m

Index