Electromagnetics Documentation

Release 0.1.0

ayub-khan

Jul 18, 2019

Contents:

CHAPTER 1

Electromagnetics

Electromagnetics is a suit of algorithms used in electromagnetics problems.

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

1.1 Features

- TODO

1.2 Credits

This package was created with Cookiecutter .

CHAPTER 2

Installation

2.1 Stable release

To install Electromagnetics, run this command in your terminal:

```
$ pip install electromagnetics
```

This is the preferred method to install Electromagnetics, 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 Electromagnetics can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/ayub-khan/electromagnetics
```

Or download the tarball:

```
$ curl  -OL https://github.com/ayub-khan/electromagnetics/tarball/master
```

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

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

CHAPTER 3

Usage

To use Electromagnetics in a project:

```
import electromagnetics
```

CHAPTER 4

electromagnetics

4.1 electromagnetics package

4.1.1 Subpackages

electromagnetics.tests package

Submodules

electromagnetics.tests.test_charge module

electromagnetics.tests.test_coulombs_law module

Module contents

4.1.2 Submodules

4.1.3 electromagnetics.charge module

Charge : Charge is a property of matter that creates force.

`electromagnetics.charge.`**`negative_charge`**(*weight*)
This calculates the total amount of negative charge in a item

Parameters **`weight`** – weight in kgs

Returns charge: charge in coulomb

`electromagnetics.charge.`**`positive_charge`**(*weight*)
This calculates the total amount of positive in a item

Parameters **`weight`** – weight in kgs

Returns charge: charge in coulomb

4.1.4 electromagnetics.constants module

Constants : This will contain all the constants used in Electromagnetics.

4.1.5 electromagnetics.coulombs_law module

Coulomb's law : a law stating that like charges repel and opposite charges attract, with a force proportional to the product of the charges and inversely proportional to

the square of the distance between them.

electromagnetics.coulombs_law.**accumulative_force_on_charge_by_other**(*main_charge*, *charges_dict_with_distance*)

This returns the accumulative force on charge one by other charges.

Parameters

- **main_charge** – Charge on which we are measuring force. (unit : C)
- **charges_dict_with_distance** – list of charges with their distance and charge example= [

  {charge: amount, distance: amount, unit_vector: 1}, {charge: amount, distance: amount, unit_vector: -1}, {charge: amount, distance: amount, unit_vector: 1},]

Returns Total Accumulative force on main charge. (unit : N)

Example :

accumulative_force_on_charge_by_other(10e-6, [

{'charge':10e-6,'distance':.15,'unit_vector':-1}, {'charge':10e-6,'distance':.6,'unit_vector':1}, {'charge':10e-6,'distance':math.sqrt(.6*.6+.15*.15),'unit_vector':-1}

]

)

electromagnetics.coulombs_law.**two_charges_vector_force**(*charge_1*, *charge_2*, *unit_vector*, *distance*)

This will calculate the amount of repulsion or attraction force between two charges. This returns the vector force from charge_1 to charge_2

Parameters

- **charge_1** – Charge 1 (unit : C)
- **charge_2** – Charge 2 (unit : C)
- **unit_vector** – Direction of force
- **distance** – Distance between both charges (unit : m)

Returns Total amount of repulsion or attraction force between charges. (unit : N)

Example : two_charges_vector_force(10e-6, 10e-6 ,-1, .13)

electromagnetics.coulombs_law.**two_charges_vector_force_magnitude_only**(*charge_1*, *charge_2*, *distance*)

This will calculate the amount of repulsion or attraction force between two charges.

Parameters

- **charge_1** – Charge 1 (unit : C)
- **charge_2** – Charge 2 (unit : C)
- **distance** – Distance between both charges (unit : m)

Returns Total amount of repulsion or attraction force between charges. (unit : N)

Example: two_charges_vector_force(10e-6, 10e-6, .10)

4.1.6 Module contents

Top-level package for Electromagnetics.

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/ayub-khan/electromagnetics/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

Electromagnetics could always use more documentation, whether as part of the official Electromagnetics 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/ayub-khan/electromagnetics/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 *electromagnetics* for local development.

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

```
$ git clone git@github.com:your_name_here/electromagnetics.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 electromagnetics
$ cd electromagnetics/
$ 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 electromagnetics tests
$ python setup.py test or py.test
$ 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 2.7, 3.4, 3.5 and 3.6, and for PyPy. Check https://travis-ci.org/ayub-khan/electromagnetics/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ py.test tests.test_electromagnetics
```

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:

```
$ bumpversion 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

- ayub-khan <muhammadayubkhan6@gmail.com>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

History

7.1 0.1.0 (2019-02-12)

- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

e

Index