plasmaboundaries

Release 0.0.3

May 10, 2021

Contents

This code computes and plots analytical solutions of the Grad-Shafranov (GS) equation for studying plasma equilibrium, stability and transport in fusion reactors based on the work of A. Cerfon and J. Freidberg[1].

[1] *"One size fits all" analytical solutions to the Grad-Shafranov equation*, Physics of Plasmas 17 (2010) https://doi.org/10.1063/1.3328818

CHAPTER 1

Magnetic Flux

plasmaboundaries.magnetic_flux.**derivatives**(*f*, *order*)

Computes the derivatives of function. Does not computes xy or yx derivatives.

Parameters

- **f** (*callable f(x, y, c_i, pkg)*) – function
- **order** (*int*) – order of differenciation

Returns (fx^order, fy^order)

Return type (sympy.Add, sympy.Add)

plasmaboundaries.magnetic_flux.**psi**(*X*, *Y*, *c_i*, *A*, *config*, *pkg='numpy'*)

Computes the value of magnetic flux at point (X, Y) according to coefficients ci.

Parameters

- **X** (*float or numpy.array*) – x coordinate
- **Y** (*float or numpy.array*) – y coordinate
- **c_i** (*list*) – list of floats, the ci coefficients
- **A** (*float*) – plasma parameter
- **config** (*str*) – shape of the plasma 'non-null', 'single-null', 'double-null'.
- **pkg** (*str, optional*) – if set to 'numpy' (resp. 'sympy'), numpy (resp. sympy) objects will be used. Defaults to 'numpy'.

Raises ValueError – If argument pkg is not in ['numpy', 'np', 'sympy', 'sp']

Returns value(s) of magnetic flux

Return type float or numpy.array or sympy.Add

CHAPTER 2

Plasmaboundaries Model

plasmaboundaries.model.**compute_N_i**(*params*)
Computes N_1, N_2 and N_3 coefficients based on plasma parameters

Parameters **params** (*dict*) – contains the plasma parameters (aspect_ratio, elongation, triangularity, A)

Returns (N_1, N_2, N_3)

Return type (float, float, float)

plasmaboundaries.model.**compute_psi**(*params*, *config='non-null'*, *return_coeffs=False*)
Computes the magnetic flux fonction

Parameters

- **params** (*dict*) – contains the plasma parameters (aspect_ratio, elongation, triangularity, A)
- **config** (*str, optional*) – shape of the plasma "non-null", "single-null", "double-null". Defaults to "non-null".
- **return_coeffs** (*bool, optional*) – If True, will also return the coefficients c_i. Defaults to False.

Returns

Magnetic flux fonction and coefficients c_i (only if return_coeffs is True)

Return type (callable) or (callable, list)

plasmaboundaries.model.**constraints**(*p*, *params*, *config*)
Creates set of constraints for parametric GS solution for any plasma configuration.

Parameters

- **p** (*list*) – c_i coefficients (floats)
- **params** (*dict*) – contains the plasma parameters (aspect_ratio, elongation, triangularity, A)

- **config** (*str*) – shape of the plasma 'non-null', 'single-null', 'double-null'.

Returns set of constraints

Return type list

plasmaboundaries.model.**get_separatrix_coordinates**(*params*, *config*, *step=0.01*)

Creates a list of points describing the separatrix

Parameters

- **params** (*dict*) – contains the plasma parameters (aspect_ratio, elongation, triangularity, A)
- **config** (*str*) – shape of the plasma "non-null", "single-null", "double-null".
- **step** (*float, optional*) – Resolution of the domain. Defaults to 0.01.

Raises ValueError – If no separatrix is found within the points of interest

Returns list of points coordinates

Return type numpy.array

plasmaboundaries.model.**test_points**(*aspect_ratio*, *elongation*, *triangularity*)

Compute the coordinates of inner and outer equatorial points and high point based on plasma geometrical parameters.

Parameters

- **aspect_ratio** (*float*) – minor radius / major radius
- **elongation** (*float*) – plasma elongation
- **triangularity** (*float*) – plasma triangularity

Returns

points (x, y) coordinates

Return type ((float, float), (float, float), (float, float))

plasmaboundaries.model.**val_from_sp**(*expression*)

Transforms a sympy expression to a callable function f(x, y)

Parameters **expression** (*sympy.Add*) – sympy expression to be converted which has symbols 'x' and 'y' in it.

CHAPTER 3

Installation

You can install plasma-boundaries using Pip by running:

```
pip install plasmaboundaries
```

Alternatively you can clone the repository:

```
git clone https://github.com/RemiTheWarrior/plasma-boundaries
```

Install the dependencies

```
pip install -r requirements.txt
```

CHAPTER 4

Usage

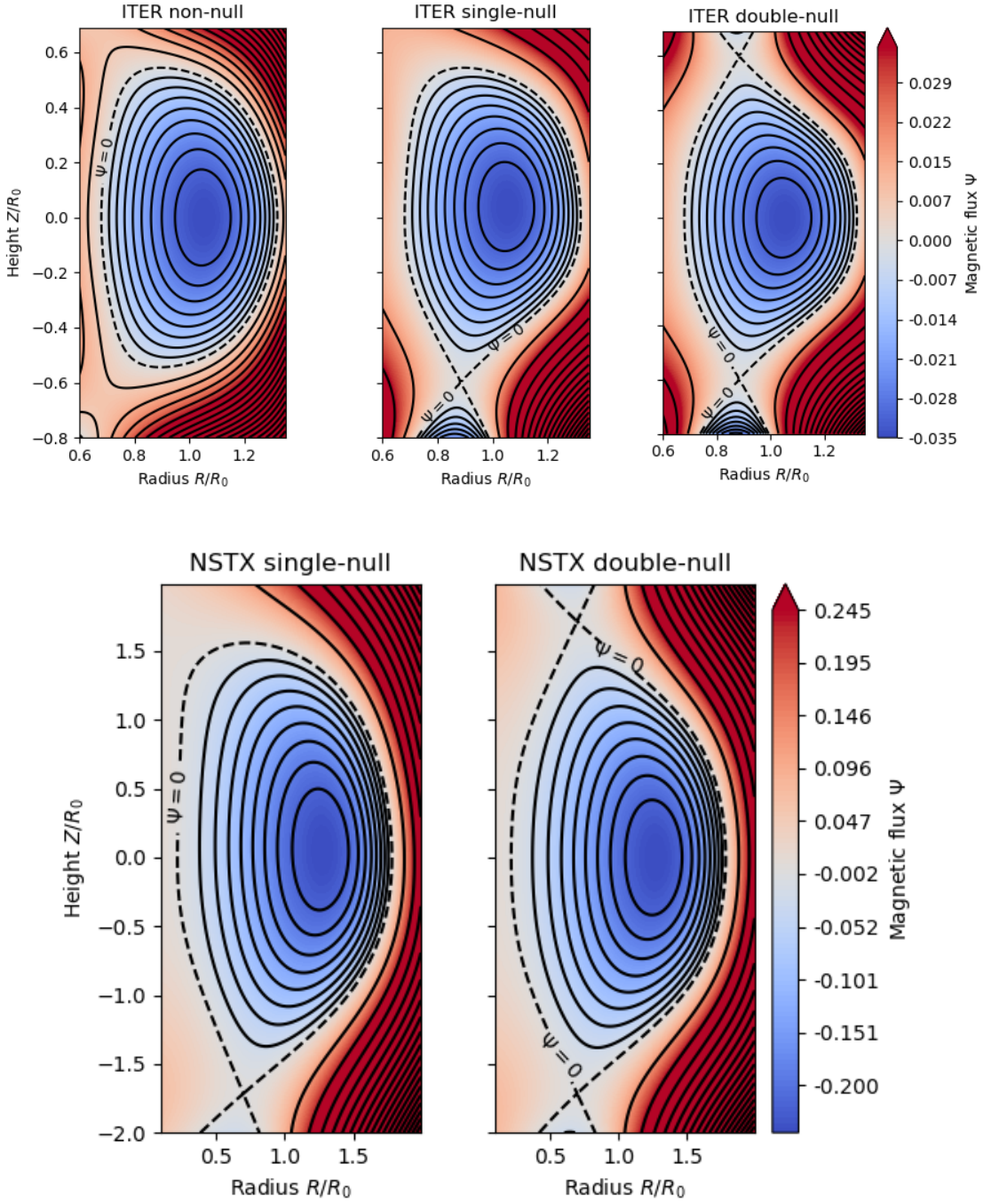
First compute the magnetic flux Ψ from plasma-boundaries based on a specific set of parameters. In this example, the built-in ITER plasma parameters will be used:

```
import plasmaboundaries

# plasma parameters
params = plasmaboundaries.ITER

# compute magnetic flux psi(R, z)
psi = plasmaboundaries.compute_psi(params, config='double-null')
```

The magnetic flux can now be calculated for any coordinates and ploted with matplotlib:

```
print(psi(1.0, 0))

# plot the results
import matplotlib.pyplot as plt
import numpy as np

rmin, rmax = 0.6, 1.4
zmin, zmax = -0.6, 0.6
r = np.arange(rmin, rmax, step=0.01)
z = np.arange(zmin, zmax, step=0.01)
R, Z = np.meshgrid(r, z)
PSI = psi(R, Z)  # compute magnetic flux

levels = np.linspace(PSI.min(), 0, num=25)
CS = plt.contourf(R, Z, PSI, levels=levels, vmax=0)
plt.contour(R, Z, PSI, levels=[0], colors="black") # display the separatrix

plt.colorbar(CS, label="Magnetic flux $\Psi$")
plt.xlabel('Radius $R/R_0$')
plt.ylabel('Height $z/R_0$')
plt.gca().set_aspect("equal")
plt.show()
```

In *compute_psi*, the argument *config* can also be set to *'single-null'* or *'non-null'* for other plasma shapes.



CHAPTER 5

Custom plasma parameters

Parameters can also be defined by creating the parameters dictionary:

```
params = {
    "A": -0.155,
    "aspect_ratio": 0.32,
    "elongation": 1.7,
    "triangularity": 0.33,
}
```

CHAPTER 6

Run the tests

You can run the tests with:

```
pytest tests/
```

CHAPTER 7

References

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index