COMPASS
INSTITUTE OF PLASMA PHYSICS ASCR

PLEQUE

–

PLasma EQUilibrium Enjoyment in Python

Lukáš Kripner[1), 2)], Matěj Tomeš[1), 2)], Ondřej Grover [2)],
Ondřej Ficker [2)], Jakub Urban [2)], and others

Katedra fyzik povrchů a plazmatu
Matematickofyzikální fakulta, Univerzita Karlova, Praha

COMPASS tokamak
Institute of Plasma Physics of the CAS, Prague

Lukáš Kripner, Matěj Tomeš, Ondřej Grover, Ondřej Ficker, Jakub Urban, and others

PLEQUE

—

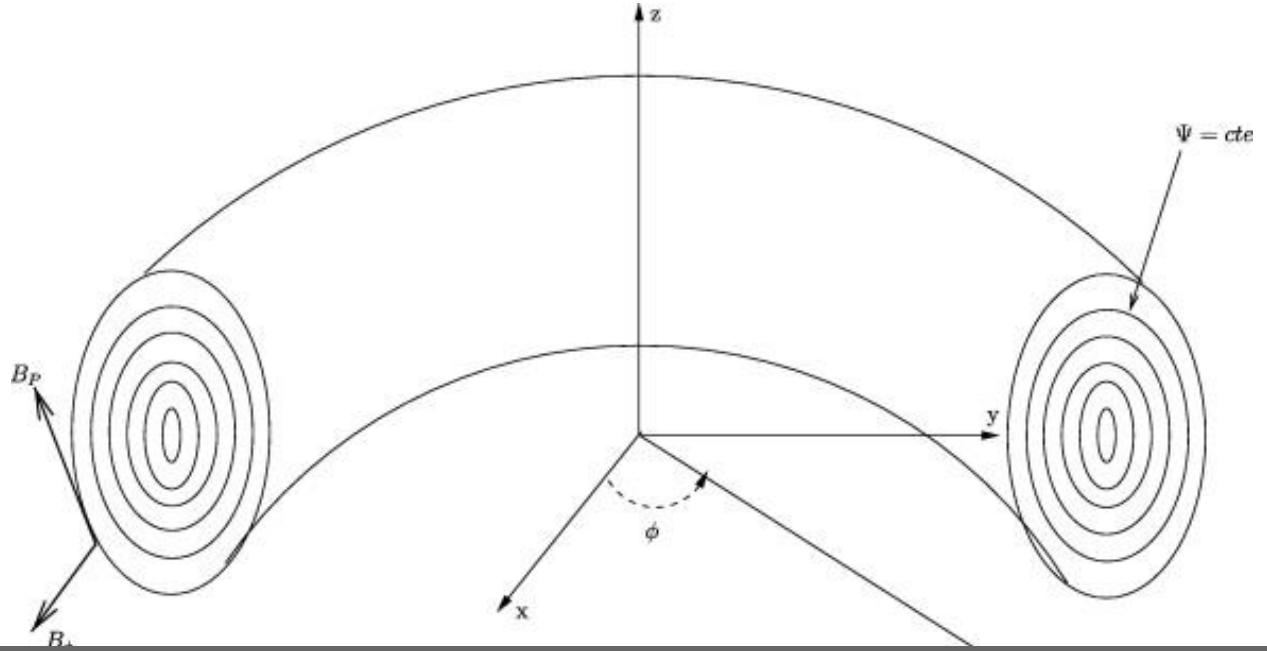
PLasma **EQU**ilibrium **E**njoyment

→ Python module for ***simple*** manipulation with tokamak plasma equilibrium

Plasma equilibrium

$$\boldsymbol{J} \times \boldsymbol{B} = \nabla P$$

- The equilibrium of a magnetically confined plasma is defined by the magnetohydrodynamic (MHD) equations.
- Equilibrium requires external poloidal magnetic field.
- For stationary conditions (compared to the Alfvén time scale) and the axial symmetry, the Grad-Shafranov equation can be derived:



Grad-Shafranov equation:

$$\Delta^* \psi = -\mu_0 R^2 \frac{\mathrm{d}p}{\mathrm{d}\psi} - \frac{f\,\mathrm{d}f}{\mathrm{d}\psi}$$

$$\Delta^* = R \frac{\partial}{\partial R}\left(\frac{1}{R}\frac{\partial}{\partial R}\right) + \frac{\partial^2}{\partial Z^2}$$

ψ = poloidal magnetic flux
p = plasma pressure
$f = RB_\phi$

How to obtain an equilibrium?

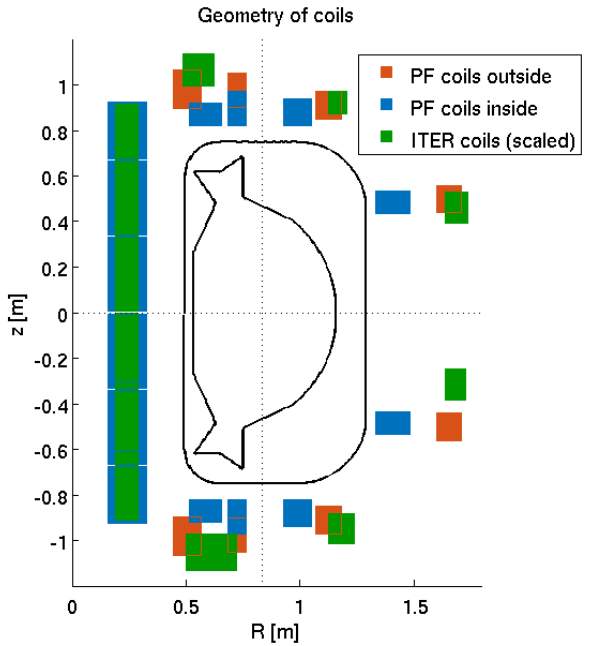
- **Fix/free boundary codes (CHEESE, HELENA, FREEBIE, FIESTA, etc.)**

$$\Delta^*\psi = -\mu_0 R^2 \frac{\mathrm{d}p}{\mathrm{d}\psi} - \frac{f\,\mathrm{d}f}{\mathrm{d}\psi}$$

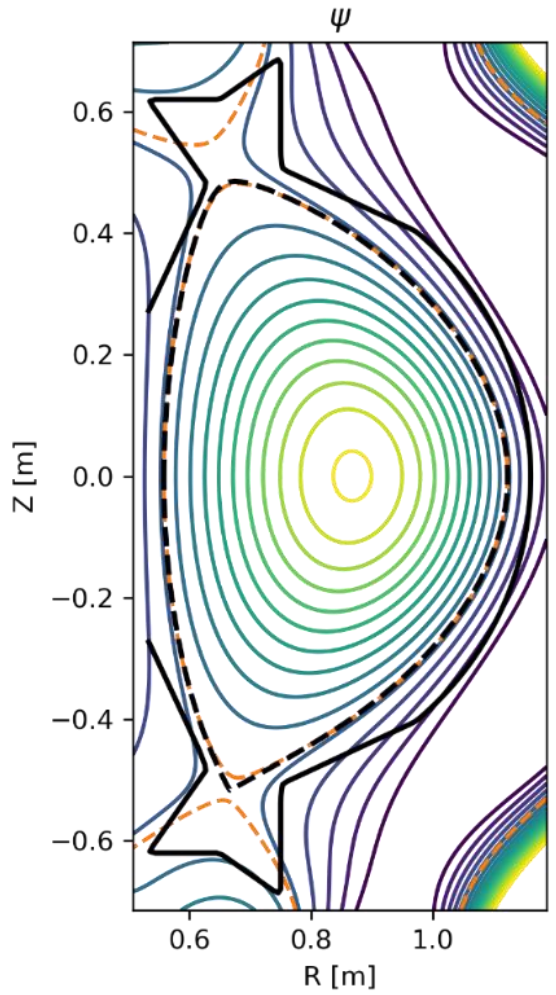
$$\Delta^* = R\frac{\partial}{\partial R}\left(\frac{1}{R}\frac{\partial}{\partial R}\right) + \frac{\partial^2}{\partial Z^2}$$

- Coil currents
- Current profile
- Pressure profile

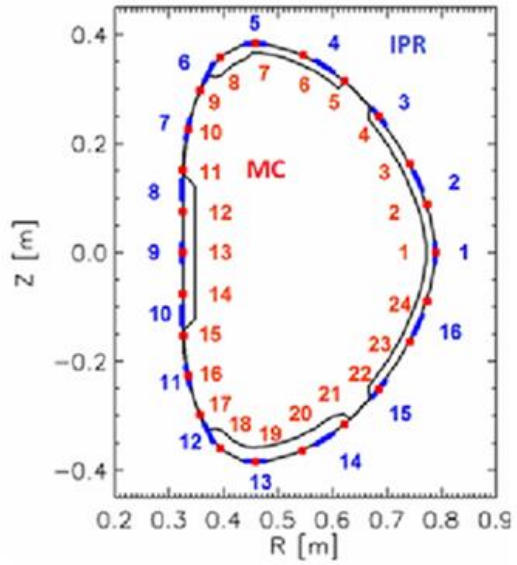
+



=



- **Equilibrium codes for experiment (EFIT, ..)**



- Guessed profiles
- Fitting measured coils
- Fitting measured plasma current
- Fitting other diagnostics

→

It is actually pretty useful

$\psi(R,Z)$
$p(\psi)$
$f(\psi)$

$\boldsymbol{B} = (B_\mathrm{R}, B_\mathrm{Z}, B_\phi)$
j_θ
j_ϕ
V_surf
$p(R,Z)$
q
A_surf
$f(R,Z)$
I_plasma

$B_R = -\frac{1}{R}\frac{\partial\psi}{\partial Z}$ $B_Z = \frac{1}{R}\frac{\partial\psi}{\partial R}$

$j_\phi = -(Rp' + \frac{1}{\mu_0} f f')$ $j_\theta = \frac{1}{R\mu_0} f' |\nabla\psi|$ $B\phi = \mathrm{Rf}$

$f = RB_\phi$

$q = \frac{\mathrm{d}\Phi}{\mathrm{d}\psi} = \frac{gV'}{(2\pi)^2}\langle R^{-2}\rangle$ $\langle a\rangle = \frac{2\pi}{V'}\oint a\frac{Rd\ell}{|\nabla\psi|}$

$I_p = \frac{V'}{2\pi\mu_0}\left\langle\frac{|\nabla\psi|^2}{R^2}\right\rangle$

Copy/passed from: Jardin, S.: Computational Methods in Plasma Physics

PLEQUE input/output

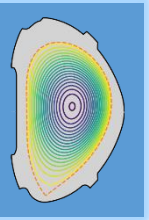
$\psi(R_i, Z_j), f(\psi_{N,k}), p(\psi_{N,k})$
first wall

COMPASS CDB /
JET DATABASE/
ASDEX-U (planned in v0.1.0)

g-file

OMAS
(planned in v0.1.0)

PLEQUE



OMAS
(Ordered Multidimensional Array Structure)

g-file (on request, will be in v0.1.0)

…

HDC

IMAS

ITM

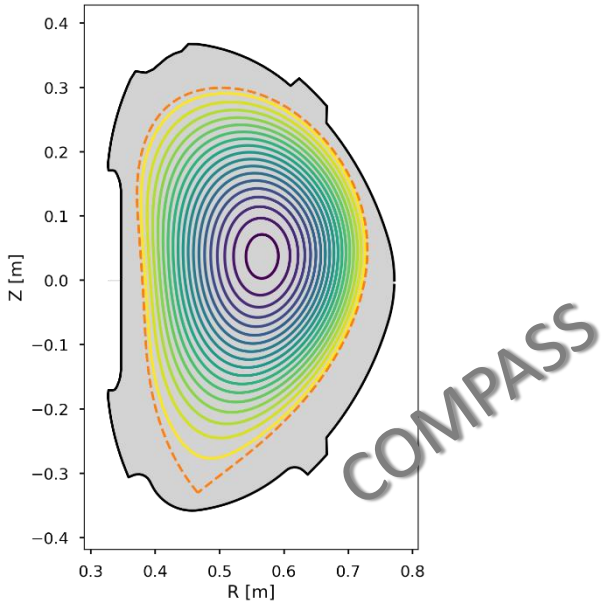
JSON

NetCDF

OMFIT

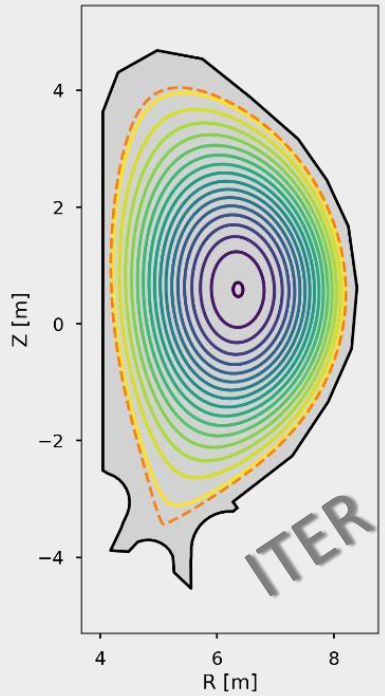
CDB

```
# CDB
from pleque.io import compass
#                    shot, time
eq = compass.cdb(17829, 1100)

# from COMPASS HDF5:
eq = compass.read_efithdf5(
'/compass/CC18_CDB_data/17829/EFITXX/EFITXX.1.h5'
, time=1100)
```



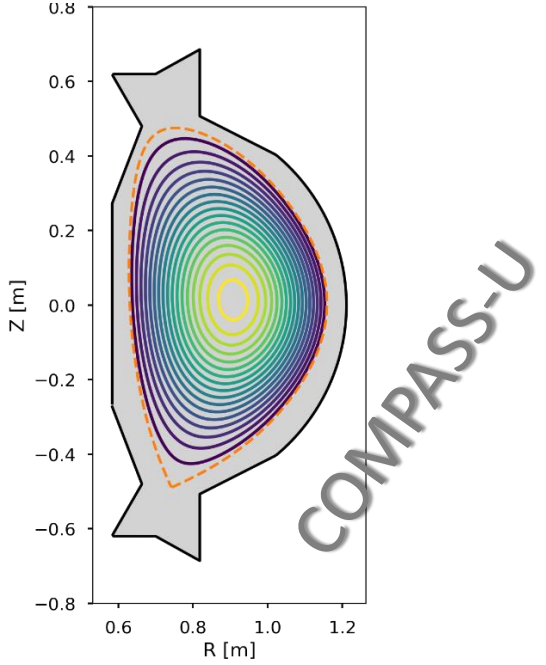
g-file

```
from pleque.io.readers import read_geqdsk

eq = read_geqdsk(
'/path_to_your_eqdsk_file/shot.eqdsk')
```



Fiesta g-file

```
from pleque.io import compass

fiesta_file = '/compass/Shared/Common/COMPASS-
UPGRADE/RP1 ,\\
  'Design/Equilibria/v3.1/baseline_eqdsk'
eq =
     compass.read_fiesta_equilibrium(
         fiesta_file)
```



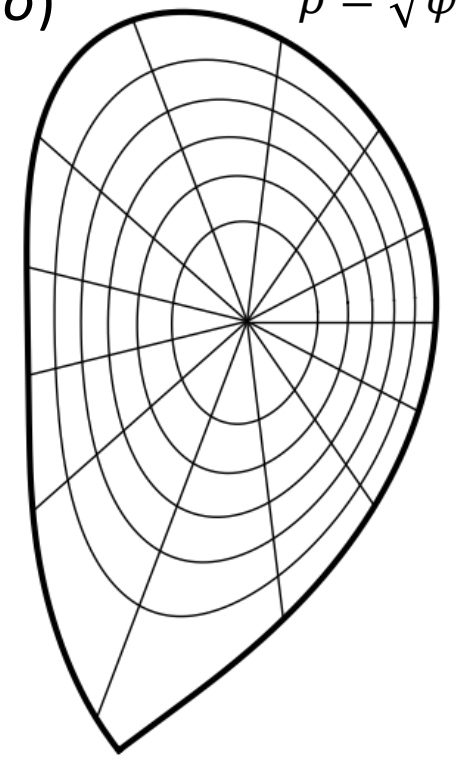
Coordinate systems

1D

- (ψ)
- (ψ_N)
- (σ)

$$\psi_N = \frac{\psi - \psi_{ax}}{\psi_{ax} - \psi_{lcsf}}$$

$$\rho = \sqrt{\psi_N}$$

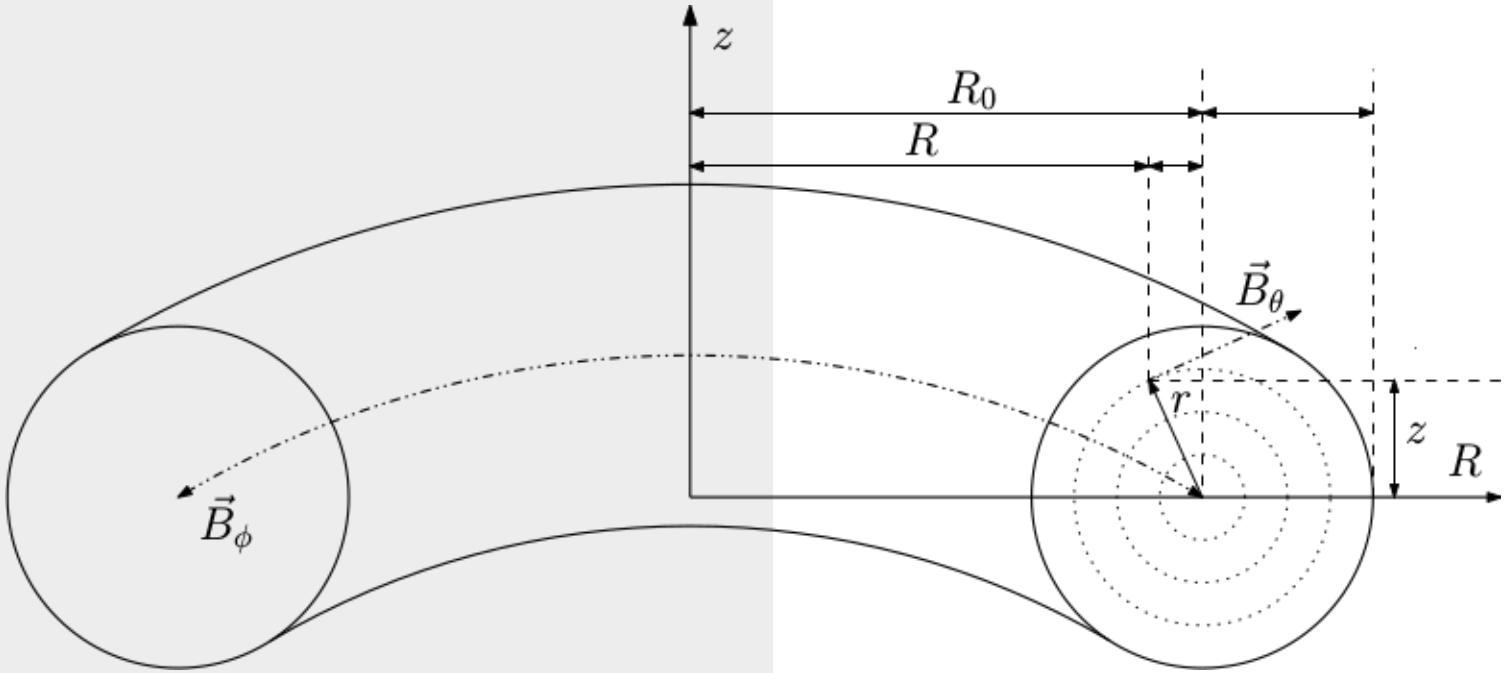


2D

- (R, Z) – tokamak axis
- (r, ϑ) – magnetic axis

3D

- (R, Z, ϕ)
- (X, Y, Z)



Create coordinates:

```
# most of the functions has a special head:
def coordinates(self, *coordinates, coord_type=None, grid=False, **coords):
```

```
# 2d
coord = eq.coordinates(R=1, Z=2) # one point
coord = eq.coordinates([1, 2, 3], [3, 5, 6])      # implicitly (R, Z)
coord = eq.coordinates(R=[1, 2, 3], Z=[3, 5, 6]) # explicitly define the type
coord = eq.coordinates(([1, 3], [2, 5], [3, 6])) # as set of points

coord = eq.coordinates(r=[0.2, 0.3, 0.3, 0.2], theta=[0, pi/2, pi, 3/2*pi])
```

```
# 1d
coord = eq.coordinates(psi=[0.4, 0.35, 0.3, 0.2, 0.15])
coord = eq.coordinates(psi_n=linspace(0, 1, 10))
```

```
# 3d case
coord = eq.coordinates(X=linspace(1, 5, 11), Y=zeros(11), Z=zeros(11)
```

Accepted coordinates

Coordinates

Accepted coordinates types

1D - coordinates

Coordinate	Code	Note
ψ_N	`psi_n`	Default 1D coordinate
ψ	`psi`	
ρ	`rho`	$\rho = \sqrt{\psi_n}$

2D - coordintares

Coordinate	Code	Note
(R, Z)	`R, Z`	Default 2D coordinate
(r, θ)	`r, theta`	Polar coordinates with respect to magnetic axis

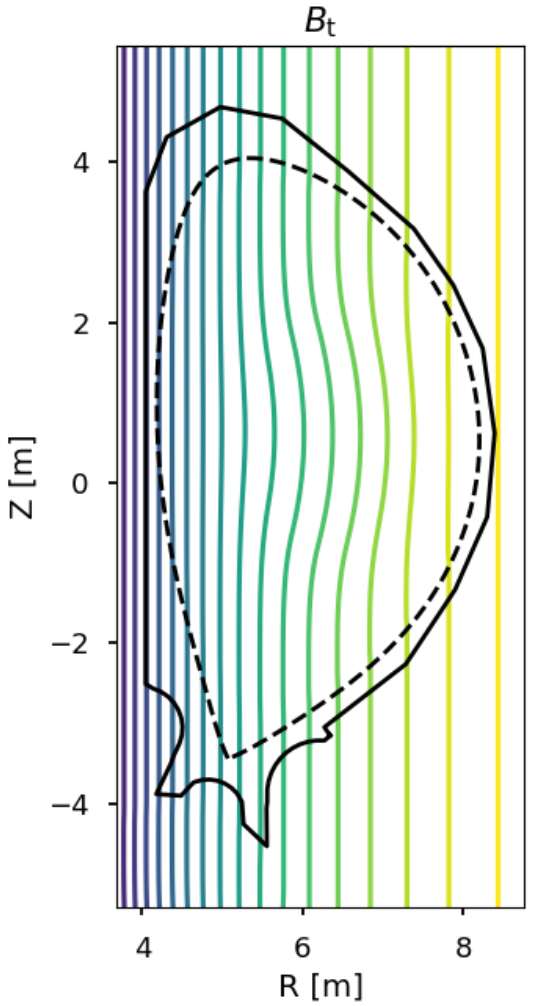
3D - coordinates

Coordinate	Code	Note
(R, Z, ϕ)	`R, Z, phi`	Default 3D coordinate
(X, Y, Z)	`X, Y, Z`	

```
coord = eq.coordinates(X=linspace(1, 5, 11), Y=zeros(11), Z=zeros(11)) # define 3D Coordinate object
```

```
r_mid = coord.r_mid      r = coord.r
R = coord.R              psi = coord.psi
X = coords.X             ...
```

Magnetic field

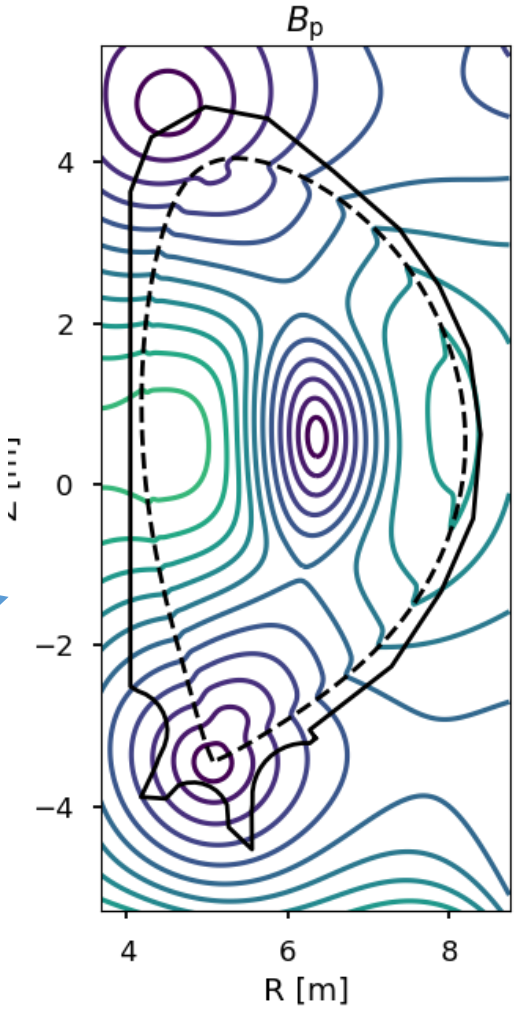



```
coord = eq.coordinates(
                linspace(4, 8.2),
                linspace(-5, 5),
                grid=True)

# this return 2D array
Bt = eq.B_tor(coord)
```

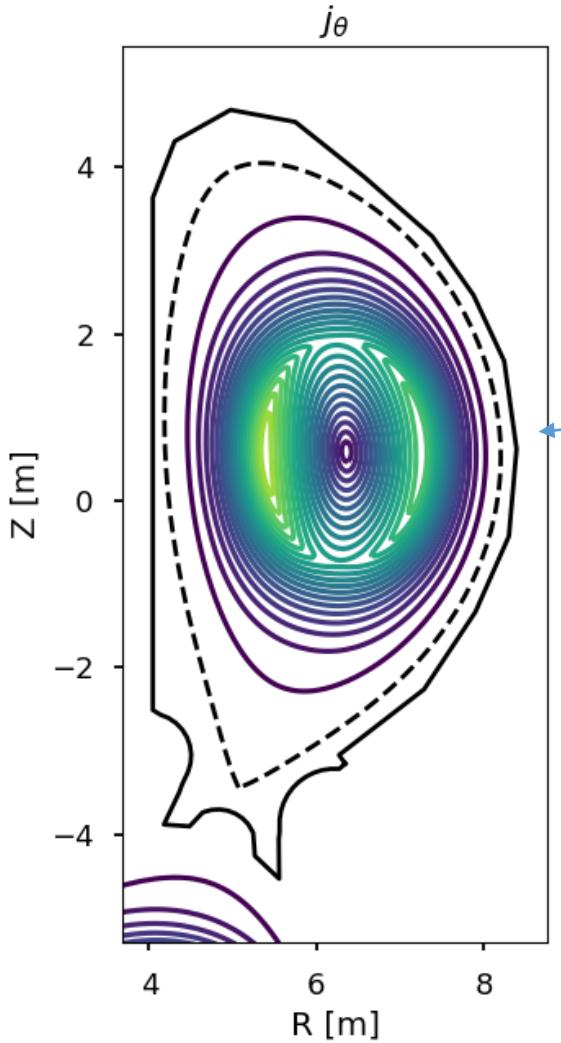
$$B_\phi = \frac{f}{R}$$

```
Bp = eq.B_pol(linspace(4, 8.2),
                linspace(-5, 5),
                grid=True)
```

$$\frac{1}{R}|\nabla\psi|$$



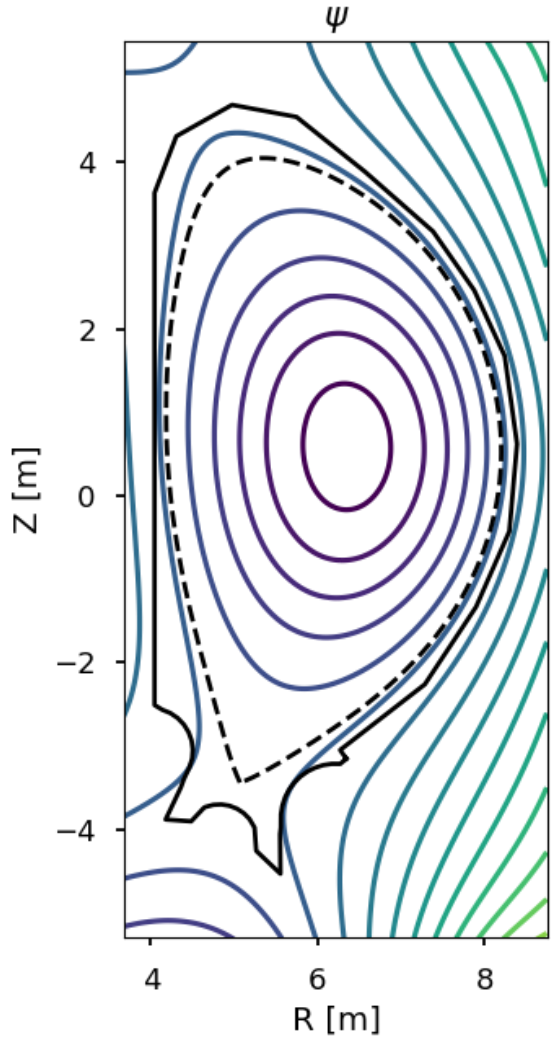
Plasma current




```
# default Coordinates grid
grid = eq.grid()
j_pol = eq.j_pol(grid)
```

$$j_\theta = \frac{1}{R\mu_0} f' |\nabla\psi|$$

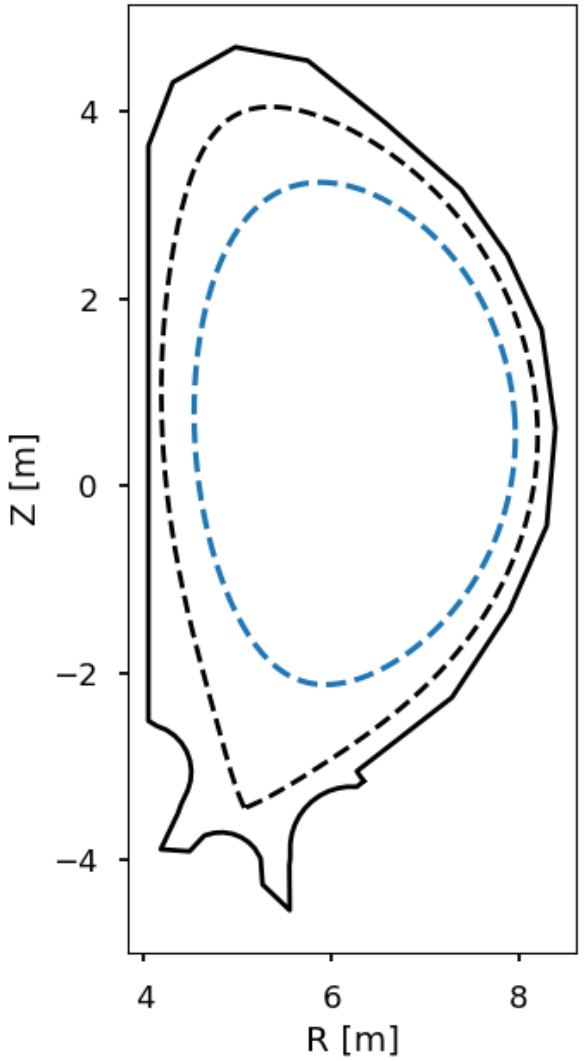
```
psi = grid.psi
```



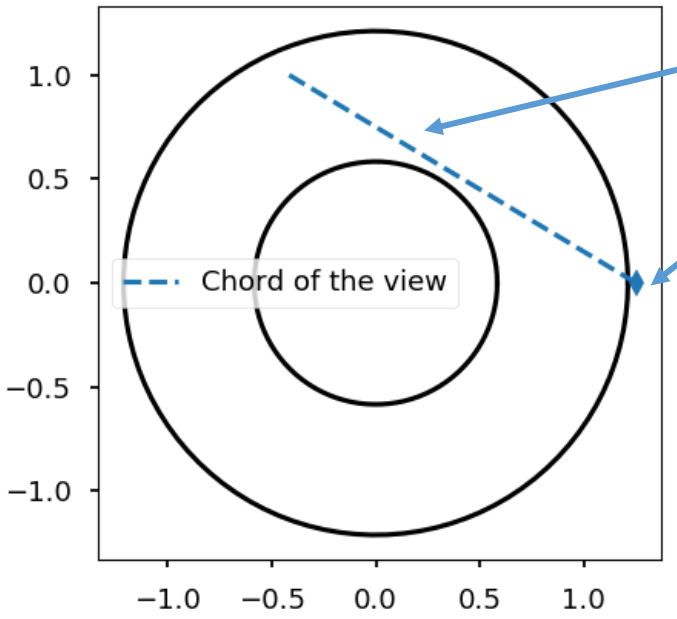
FluxSurfaces

```
surf = eq.flux_surface(psi_n=0.8)[0]

    dict(q           = surf.eval_q,
         length      = surf.length,
         area        = surf.area,
         volume      = surf.volume,
         current = surf.tor_current/1e6)
```

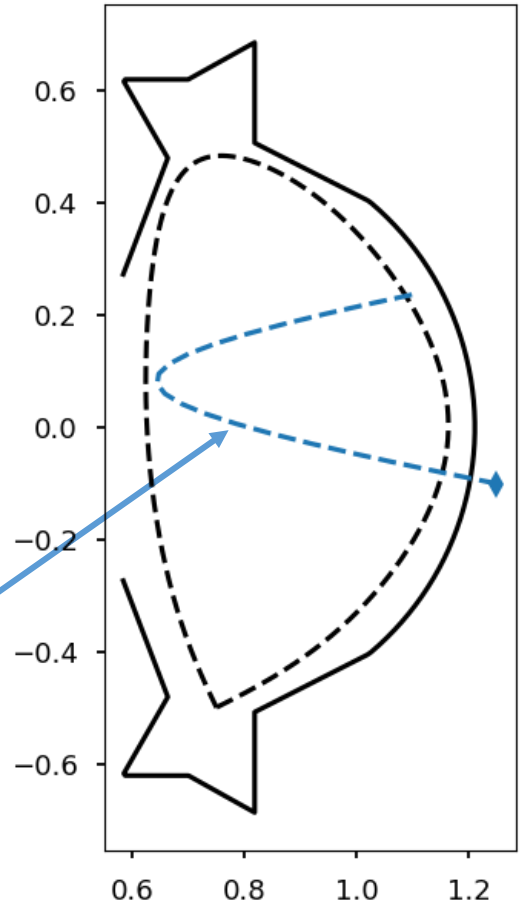
```
{'q': array([-1.99901063]),
 'length': 13.973753170736375,
 'area': 14.311638142524348,
 'volume': 554.01117642953955,
 'current': 13.000279670184989}
```



3D coordinates

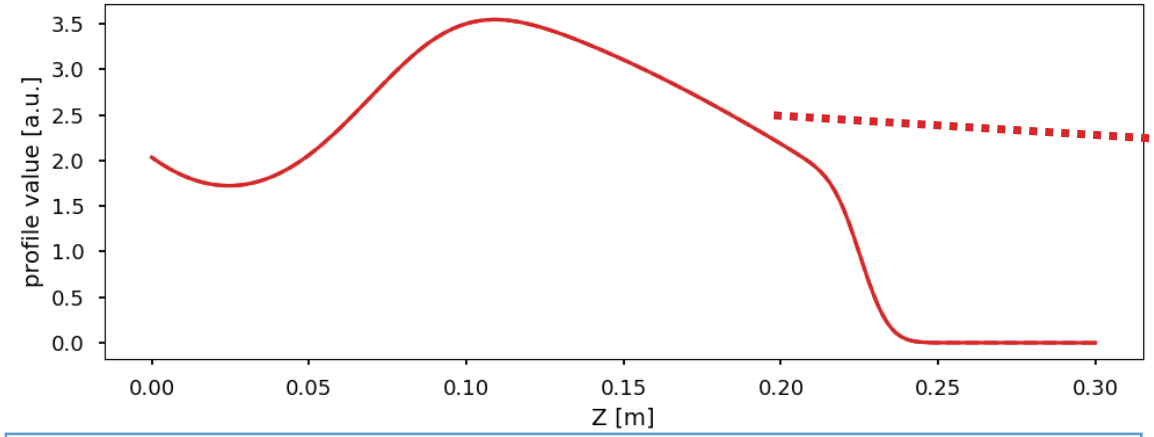



```
#                       ( x,    y,    z)
direction = array((-1, 0.6, 0.2))
direction /= norm(direction)

position = array((1.25, 0, -0.1))

camera_view = eq.coordinates(
        position+direction[newaxis,:]*
        linspace(0, 2.0, 20)[:, newaxis],
        coord_type=('X', 'Y', 'Z'))


plot(camera_view.X, camera_view.Y, 'x--',
        label='Chord of the view')
...

plot(camera_view.R, camera_view.Z, 'x--')
```

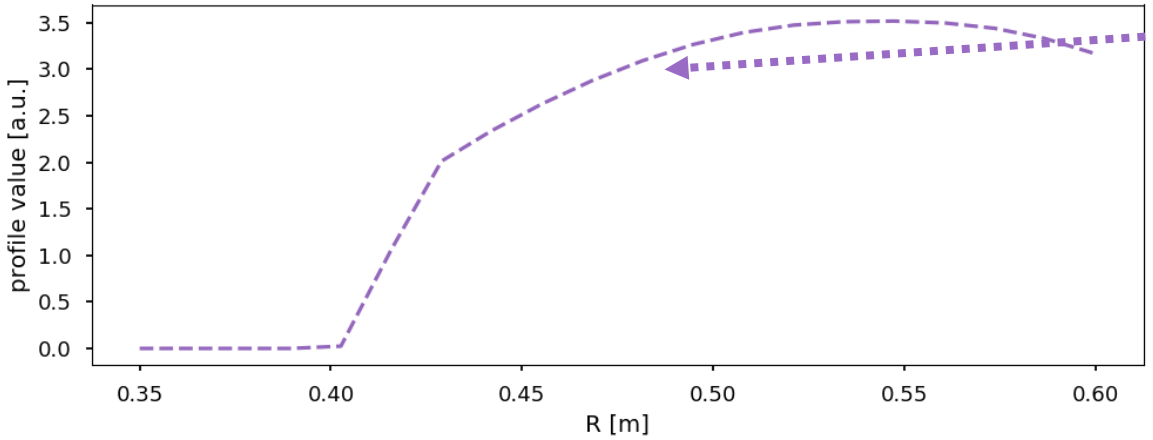


Side note: there is module `pleque.spatran` which can handle work with diagnostics much better (**responsible person: Matej**, to be presented)

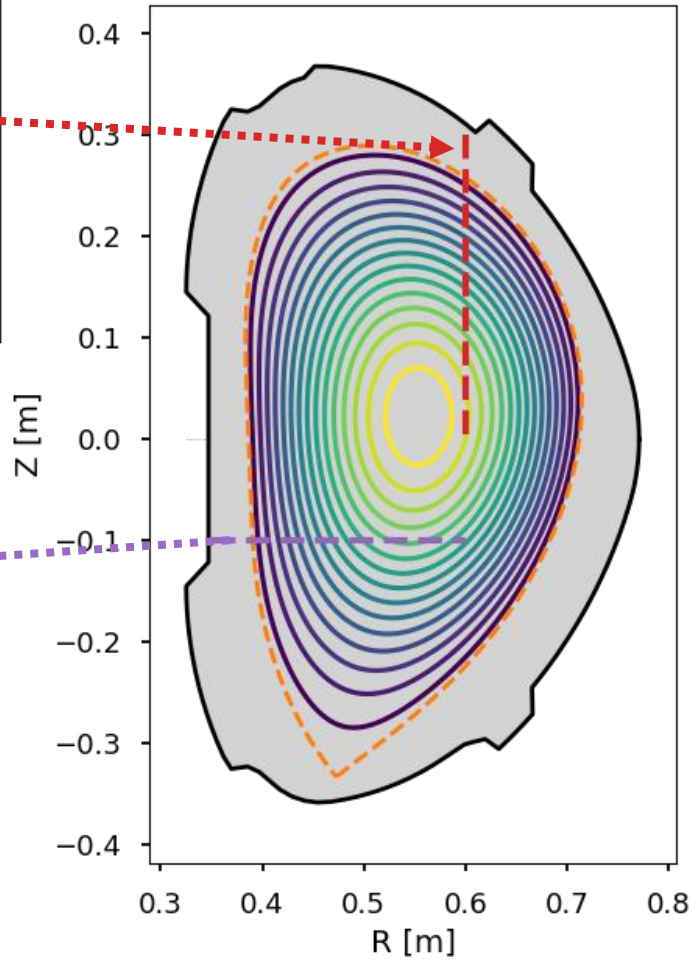
Profiles

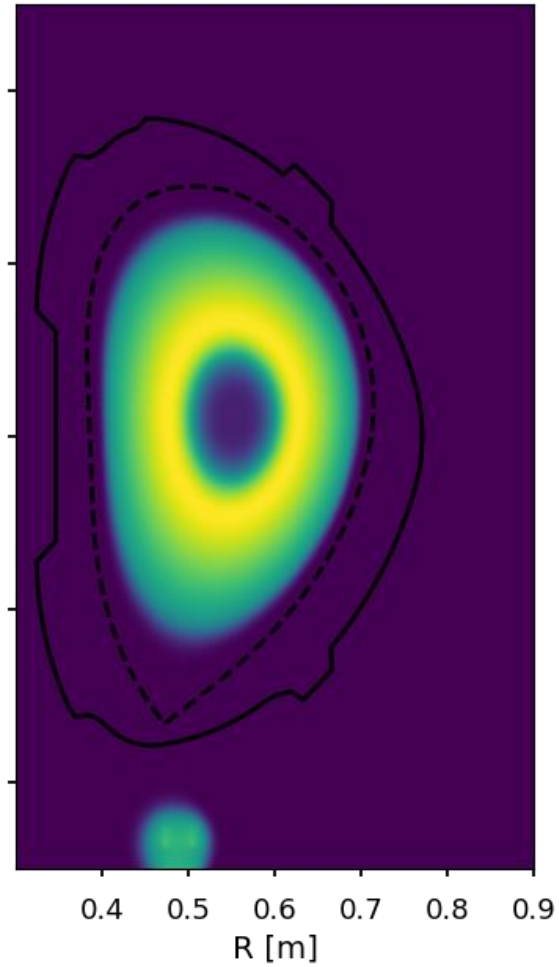



```
eq.fluxfuncs.add_flux_func('test_prof',
                           chord_prof, chord1)
```



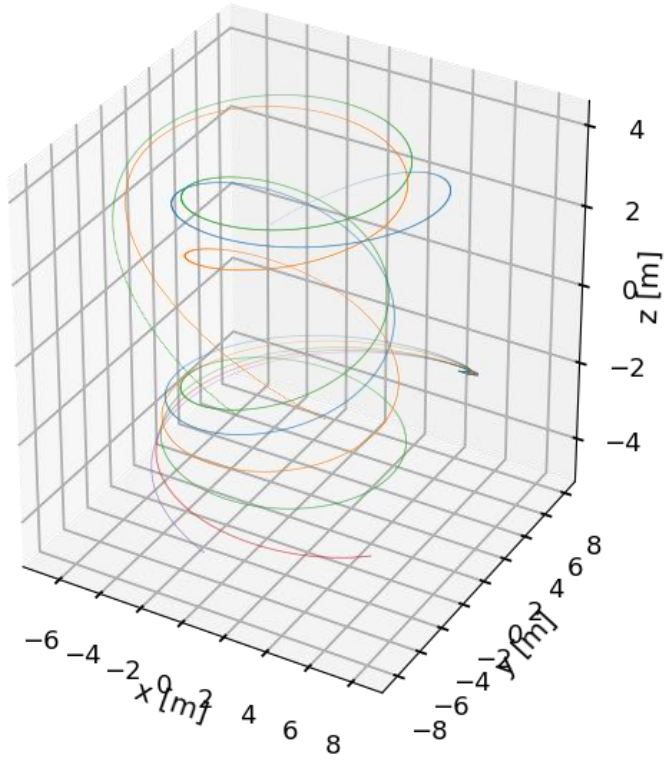

```
plt.plot(chord2.R, eq.fluxfuncs.test_prof(chord2),
                                   '--', color='C4')
```

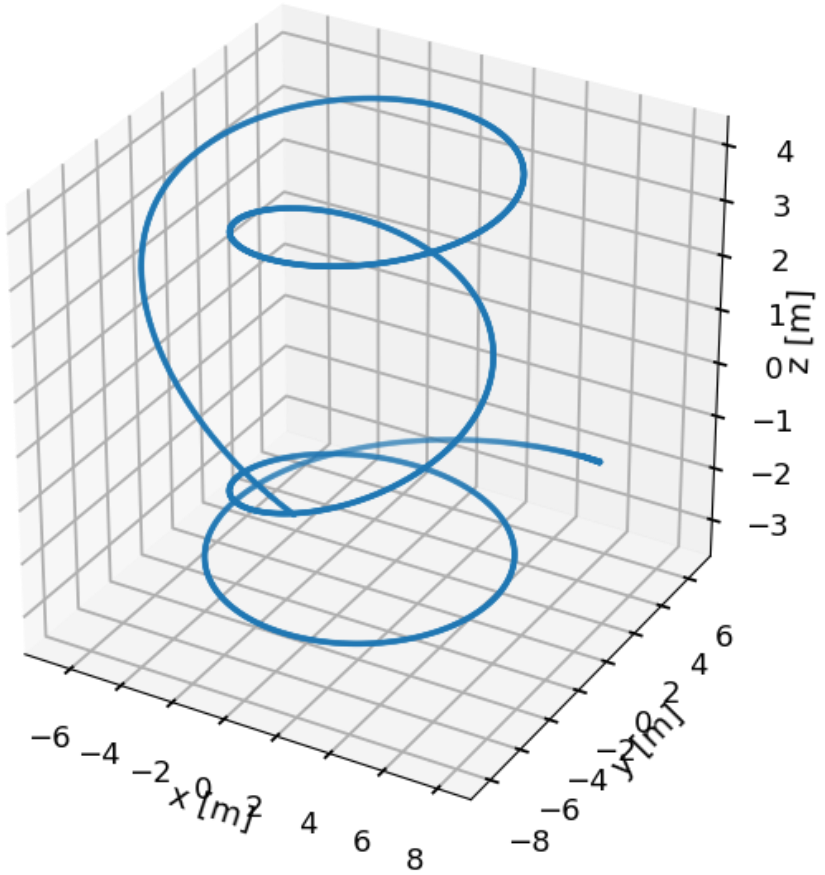


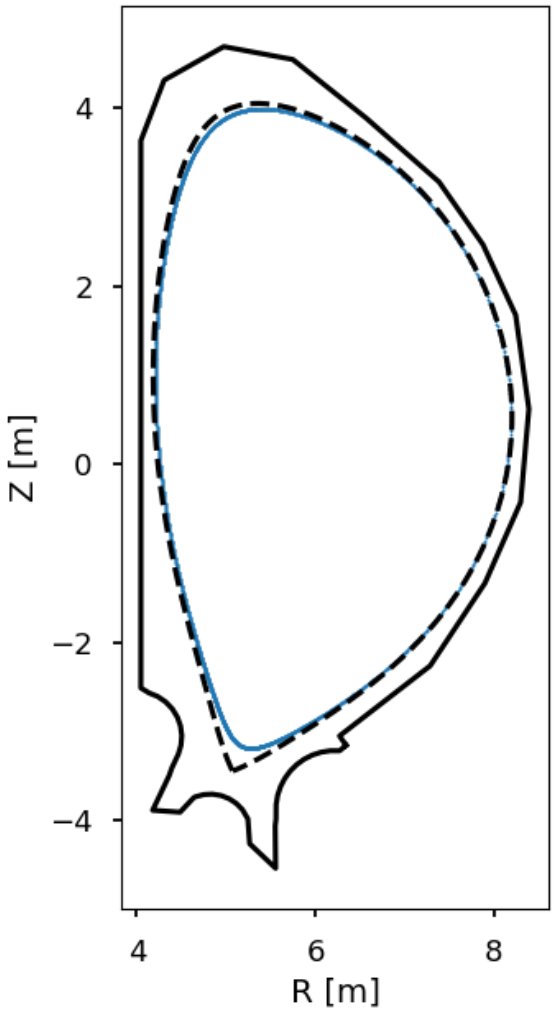



```
grid = eq.grid()
ax.pcolormesh(grid.R, grid.Z, eq.fluxfuncs.test_prof(grid))
```

Field line tracing

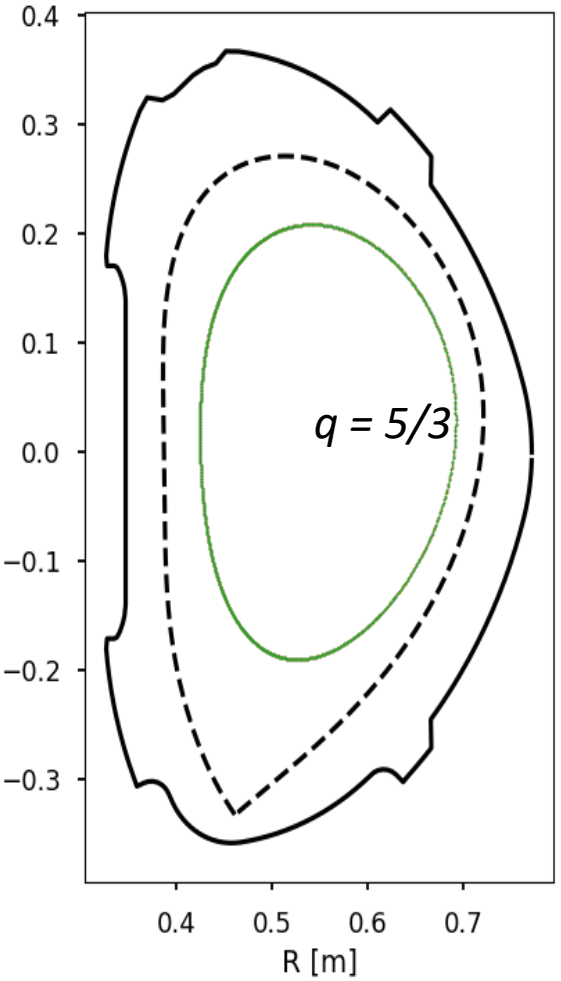


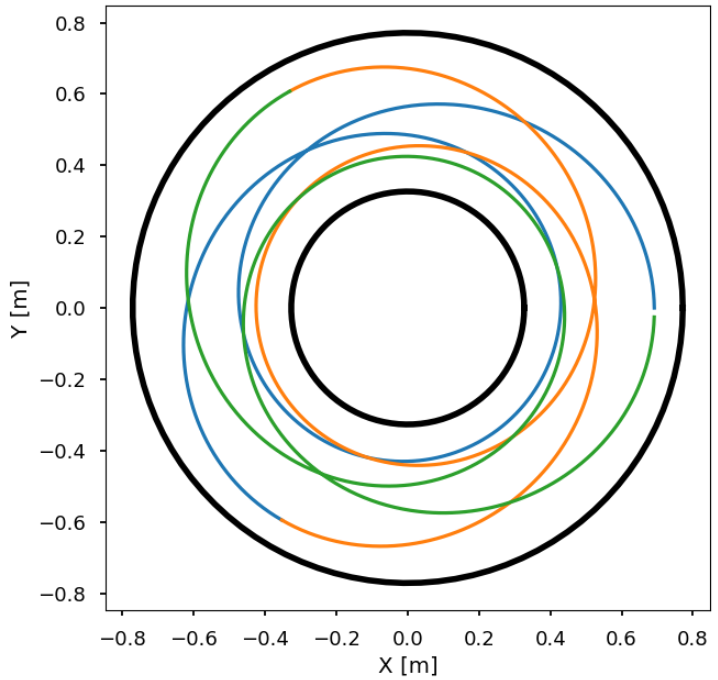


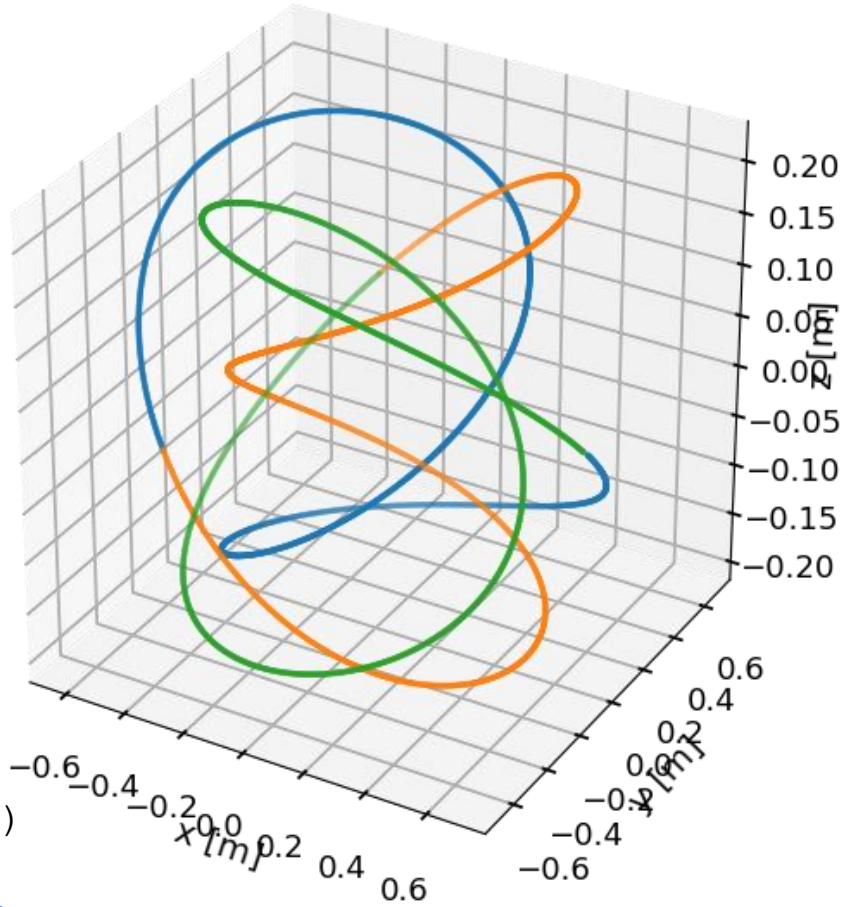



```
rs = np.linspace(8, 8.3, N)
zs = np.zeros_like(rs)

traces = eq.trace_field_line(R=rs, Z=zs)

for fl in traces:
        scatter(fl.X, fl.Y, fl.Z, s=0.3, marker='.')
```

Straight Field Line Coordinates θ^* [experimental]



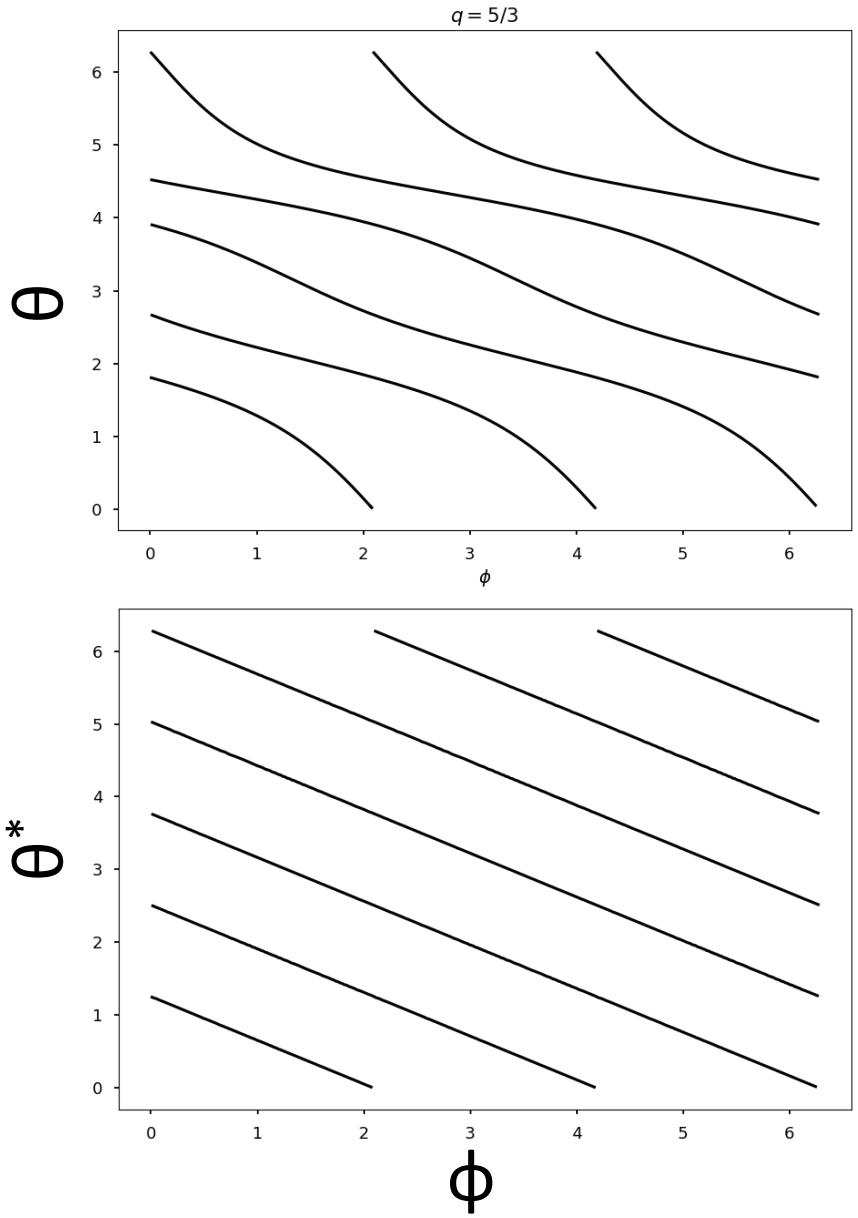


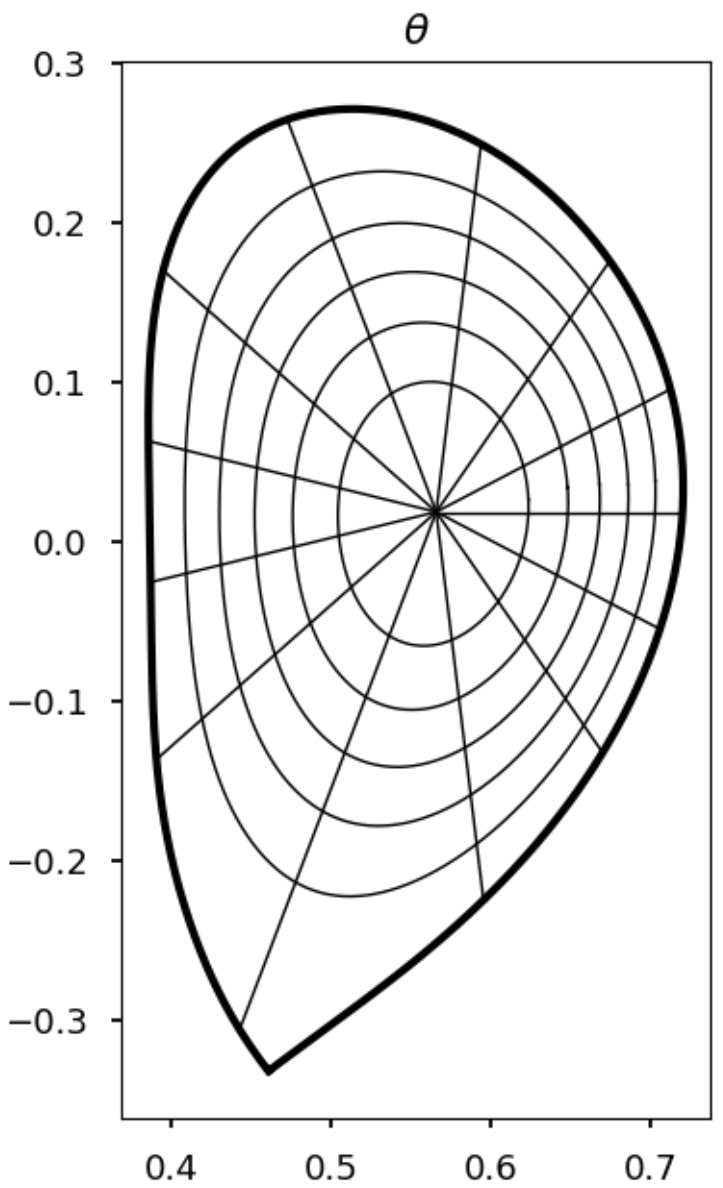


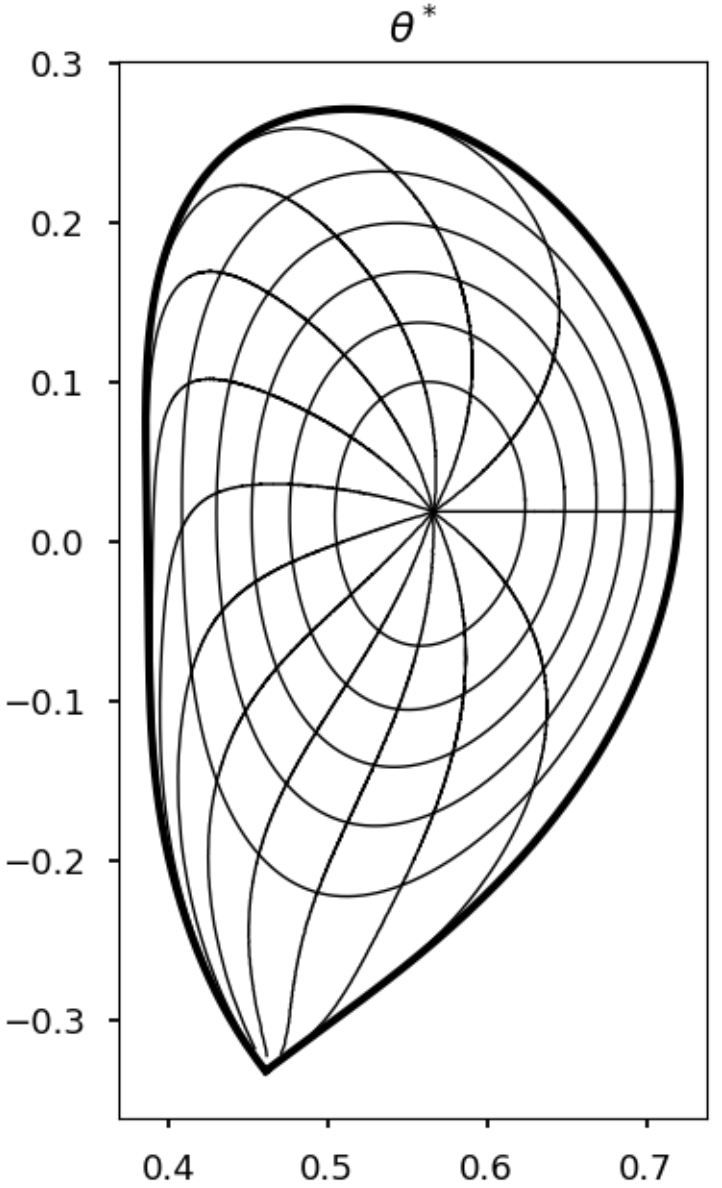

```
# brentq = scipy minimization function
from scipy.optimize import brentq
psi_onq = brentq(lambda psi_n:
        np.abs(eq.q(psi_n)) - 5/3, 0, 0.95)

surf = eq._flux_surface(psi_n = psi_onq)[0]

theta_star = surf.straight_fieldline_theta
```

COMPASS
INSTITUTE OF PLASMA PHYSICS ASCR

Straight Field Line Coordinates θ* [experimental]





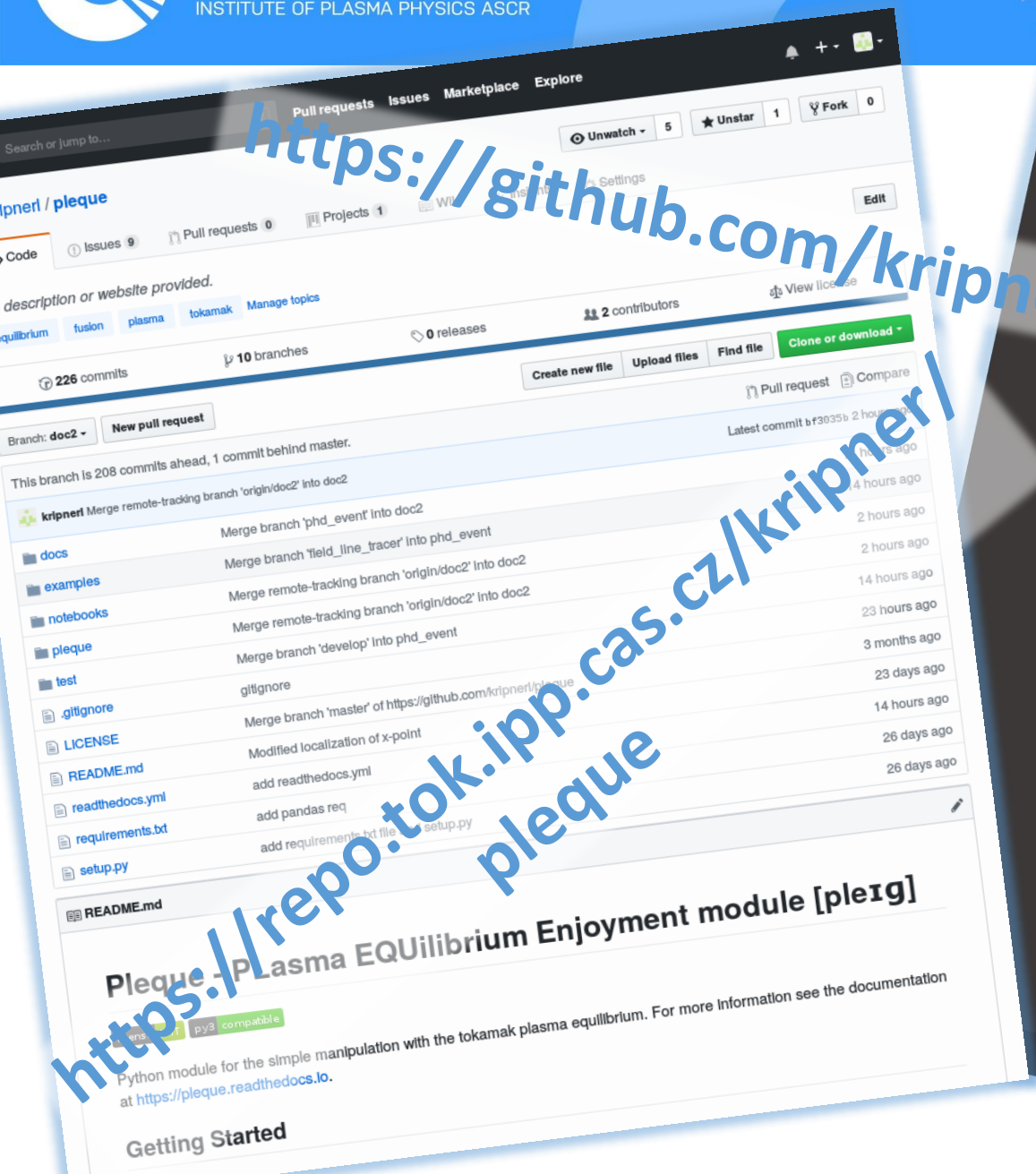


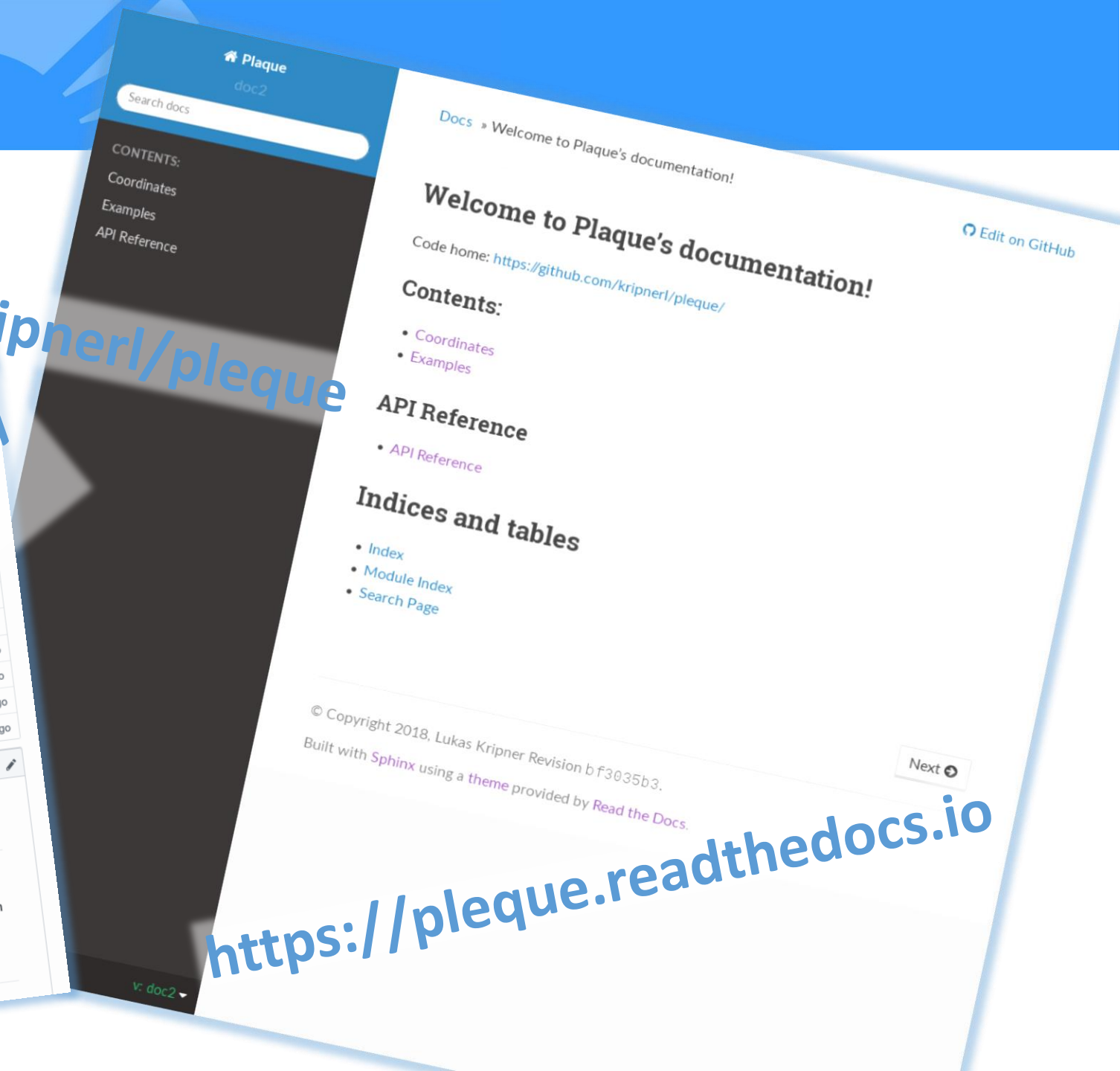
COMPASS
INSTITUTE OF PLASMA PHYSICS ASCR

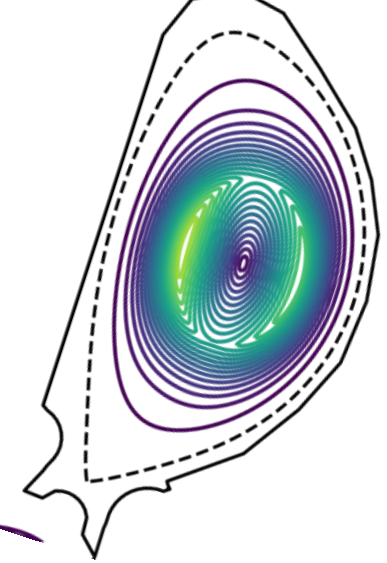
https://github.com/kripnerl/pleque

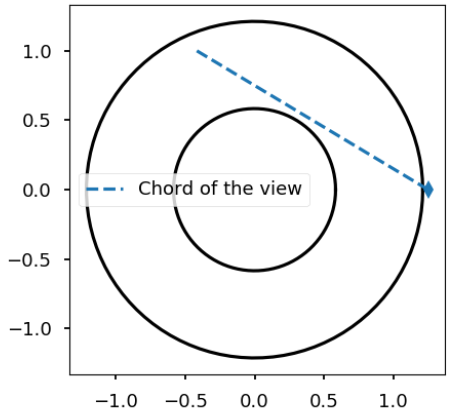
https://repo.tok.ipp.cas.cz/kripnerl/pleque

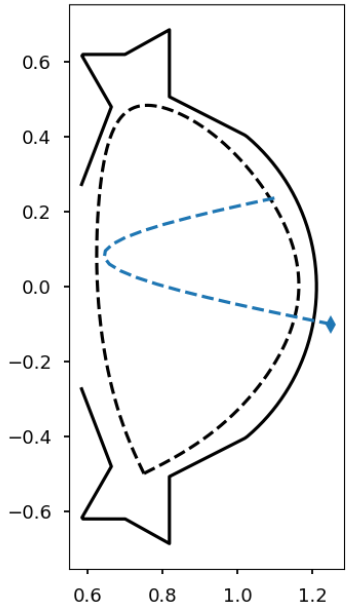
https://pleque.readthedocs.io

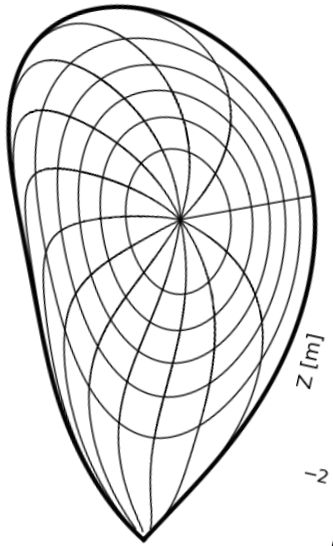


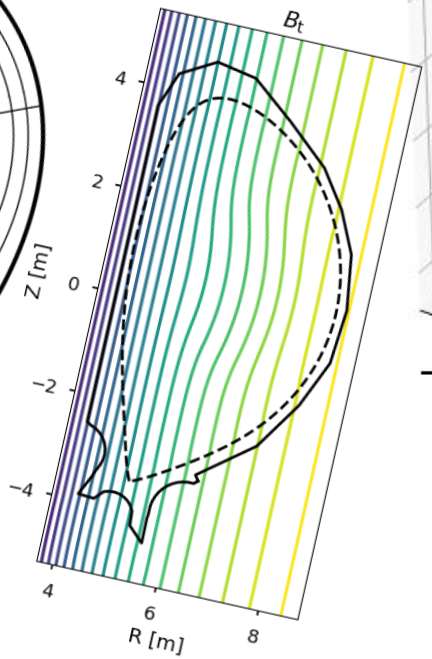


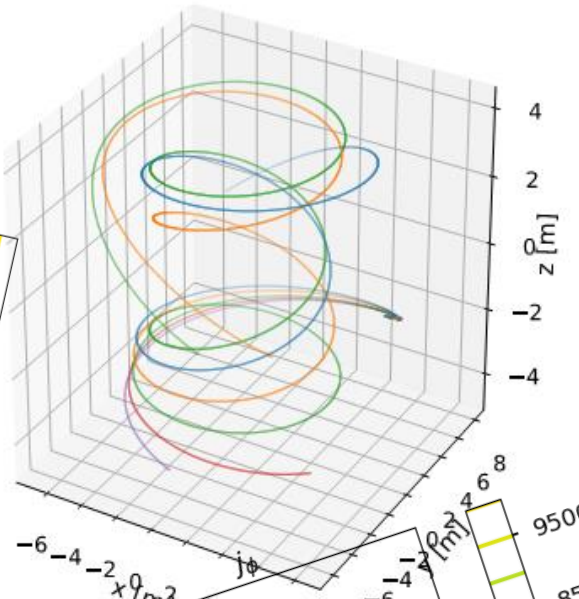


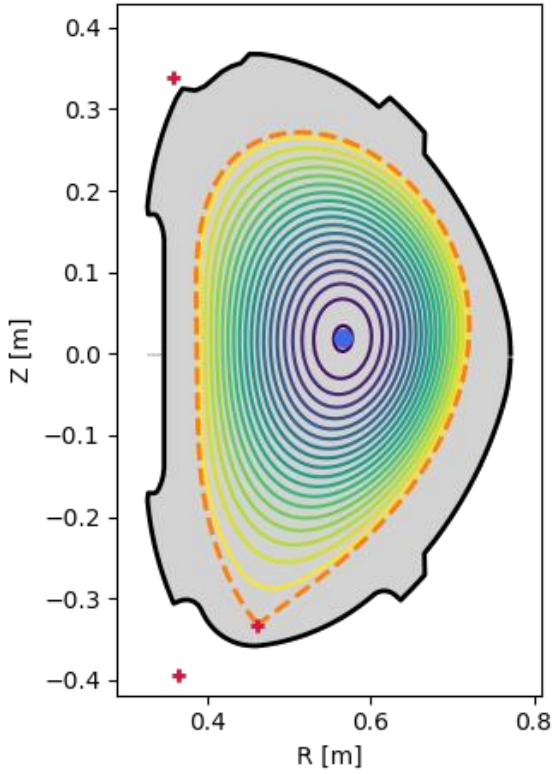

Chord of the view

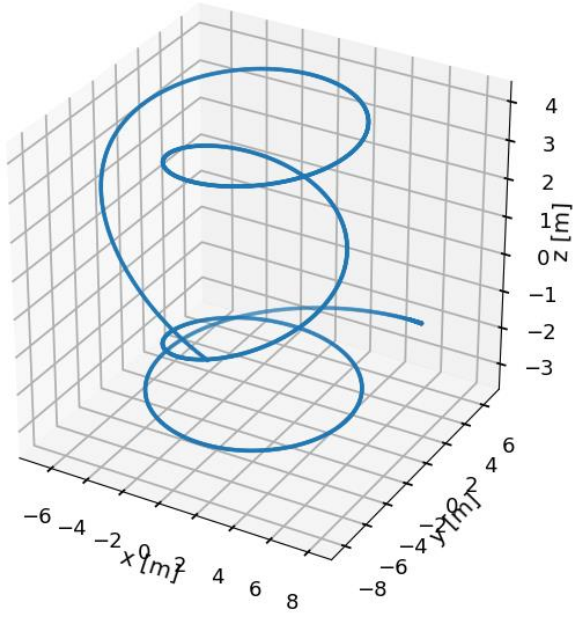


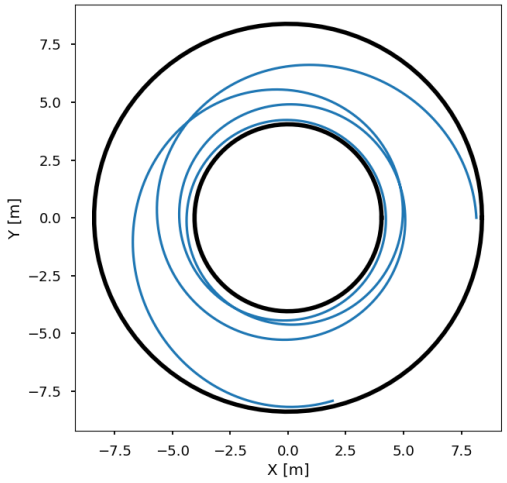


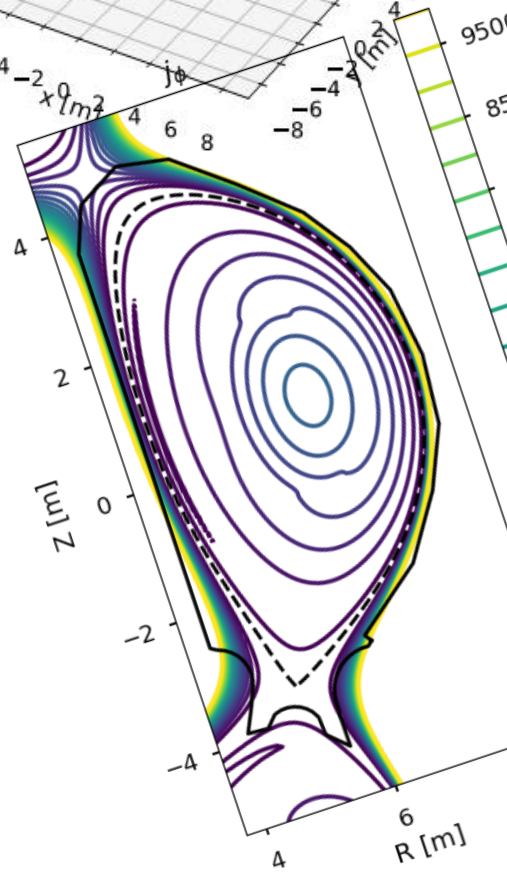

B_t
Z [m]
R [m]


z [m]
jϕ


Z [m]
R [m]


z [m]


Y [m]
X [m]


Z [m]
R [m]

Questions?

- Create an issue on gitlab/github.
- To shy for that? Write me email to kripner@ipp.cas.cz