pydisc - First Tutorial

March 2, 2020

1 pydisc Notebook - 1.0

1.1 1 - Introduction

pydisc is a python package meant to ease the use the manipulation of maps and profiles and the computation of basic quantities pertaining to galactic Discs. In this Notebook, I will show you how to use the main functionalities of *pydisc*.

1.2 2 - Structures in *pydisc*

1.2.1 DataMaps, DataProfiles, Maps and Profiles

In the language of *pydisc*: - **DataMaps** are data on a grid (hence 2D), e.g., values like velocities, flux, etc. The grid on which it is defined should be regular. - **DataProfiles** are data on a radial profile (hence 1D). - **Maps** are then a **set of** *DataMaps* associated with a set of coordinates X, Y. - **Profiles** are then a **set of** *DataProfiles* associated with a set of radial coordinates R.

DataMaps have orientations defined as 'NE_direct' indicating if the North-to-East axis is direct (counter-clockwise) or indirect (clockwise). It also has an 'alpha_North' angle which provides the angle between the North and the top (positive y-axis). DataMaps also have a pixelscale which provides the conversion between arcseconds and pixels in case the X,Y grids are not defined. If this is the case, X, and Y will be computed using just the indices from the grid.

DataMaps and DataProfiles have 'units' as defined by astropy units. These should be compatible with e.g., arcseconds, so these are observational.

DataMap arguments: - dunit: astropy unit - order: velocity moment order. Hence velocities are order=1, flux or mass is 0, dispersion is 2, anything else would be -1 and there is a category for "dummy" maps with order=-10. - dname: name of the datamap - flag: a flag which is meant to add info (string) - data and edata: numpy arrays. If edata is not provided, it will be defined as None.

DataProfiles have similar arguments, but with punit (profile unit) and pname.

Maps arguments: - name: name of the map - X and Y: the 2 main arrays. If not provided, indices will be used. - Xcen and Ycen: centre for the 0,0 - XYunit: unit (astropy) for the X and Y axis - NE_direct, alpha_North, etc.

Note that a Map can have many datamaps: hence a set of X,Y can have many data associated to it (sharing the same coordinates), each one having a different dname, order, flag etc.

1.2.2 Galaxy

A 'Galaxy' is an object which has some characteristics like: a distance, a Position Angle for the line of Nodes, an inclination (in degrees) and the Position Angle for a bar if present.

1.2.3 GalacticDisc

A 'GalacticDisc' is a structure associating a set of Maps and Profiles and a given Galaxy.

This is the main structure which we will be using for the calculation of various quantities.

There are a number of associated classes, namely: - 'DensityWave': associated with methods for density waves like the Tremaine Weinberg method - 'GalacticTorque': associated with methods for deriving torques

all inheriting from the GalacticDisc class, thus sharing a number of functionalities, but also have their own specific ones (which require a set of maps).

The 'grammar' for maps and datamaps is simple (a priori): - if you have an attribute like "data" you can input this in the argument list as: "data*". Hence if the map is name"MUSE" and the datamap named "vstar" you should have an argument for the data as "dataMUSE_vstar" and the associated "edataMUSE_vstar" if you have uncertainties for this map etc. Same applies for all argument of the maps and data, for example (using the same example): orderMUSE_vstar, XMUSE, YMUSE, XcenMUSE, YcenMUSE, flagMUSE_vstar… - In this way you can have several datamaps attached to a single map and have e.g.,: XMUSE, YMUSE, dataMUSE_vstar, dataMUSE_gas, dataMUSE…*

2 3- Examples

2.1 3.1 - Tremaine Weinberg

Here is an example of how to get a Tremaine-Weinberg calculation made on a set of maps using the *DensityWave* class.

[2]:
```
# Importing the package and the DensityWave class
import pydisc
from pydisc.density_wave import DensityWave

# Importing useful modules
from astropy.io import fits as pyfits
from os.path import join as joinpath
import numpy as np

# Getting the data
ddata = "/home/science/PHANGS/MUSE/MUSEDAP/"
n1512 = "NGC1512_MAPS.fits"
# Open the Maps files
maps = pyfits.open(joinpath(ddata, n1512))
```

```
# Extract the mass, flux, and velocity maps
mass = maps['STELLAR_MASS_DENSITY'].data
flux = maps['FLUX'].data
vel = maps['V_STARS'].data
```

[3]:
```
# mname is for mapname.
mydisc = DensityWave(data_flux=flux, edata_flux=np.zeros_like(flux),
                     data_mass=mass, data_vel=vel, edata_vel=np.zeros_like(vel),
                     mname="MUSE", Xcen=462.5, Ycen=464.4, PAnodes=90)
```

```
WARNING: X or Y not provided. Using Pixel XY grid.
INFO: attaching map MUSE
```

[4]:
```
# We can now look at the structure itself. 'mydisc' has a one map, which is␣
↪named 'MUSE'.
# This map is in a dictionary and is a Map class, as shown when printing it.
mydisc.maps
```

[4]:
```
{'MUSE': <pydisc.disc_data.Map at 0x7f53c02a54d0>}
```

[5]:
```
# We can also find out about the other variables:
mydisc.maps['MUSE'].X
```

[5]:
```
array([[-452.5, -451.5, -450.5, …,  450.5,  451.5,  452.5],
       [-452.5, -451.5, -450.5, …,  450.5,  451.5,  452.5],
       [-452.5, -451.5, -450.5, …,  450.5,  451.5,  452.5],
       …,
       [-452.5, -451.5, -450.5, …,  450.5,  451.5,  452.5],
       [-452.5, -451.5, -450.5, …,  450.5,  451.5,  452.5],
       [-452.5, -451.5, -450.5, …,  450.5,  451.5,  452.5]],
      dtype=float32)
```

[6]:
```
# This Map actually has datamaps as shown here, each one having data.
# You can see that this Map has actually three datamaps, one with flux, one␣
↪with mass, the last one with vel.
mydisc.maps['MUSE'].dmaps
```

[6]:
```
{'flux': <pydisc.disc_data.DataMap at 0x7f539747e950>,
 'mass': <pydisc.disc_data.DataMap at 0x7f539755b690>,
 'vel': <pydisc.disc_data.DataMap at 0x7f539755b990>}
```

[7]:
```
# We can call the data like this (note that the array shows the nan from the␣
↪outer part of the map)
mydisc.maps['MUSE'].dmaps['flux'].data
```

[7]:
```
array([[nan, nan, nan, …, nan, nan, nan],
       [nan, nan, nan, …, nan, nan, nan],
```

```
        [nan, nan, nan, …, nan, nan, nan],
        …,
        [nan, nan, nan, …, nan, nan, nan],
        [nan, nan, nan, …, nan, nan, nan],
        [nan, nan, nan, …, nan, nan, nan]])
```

[8]:
```
# or like this using the combined "data" with the name of the data map.
mydisc.maps['MUSE'].dmaps.flux.data
```

[8]:
```
array([[nan, nan, nan, …, nan, nan, nan],
       [nan, nan, nan, …, nan, nan, nan],
       [nan, nan, nan, …, nan, nan, nan],
       …,
       [nan, nan, nan, …, nan, nan, nan],
       [nan, nan, nan, …, nan, nan, nan],
       [nan, nan, nan, …, nan, nan, nan]])
```

[9]:
```
# to make it simpler, the maps and dmaps are merged into one attribute␣
 ↪automatically
mydisc.MUSE_flux
```

[9]:
```
<pydisc.disc_data.DataMap at 0x7f539747e950>
```

[10]:
```
mydisc.MUSE_flux.data
```

[10]:
```
array([[nan, nan, nan, …, nan, nan, nan],
       [nan, nan, nan, …, nan, nan, nan],
       [nan, nan, nan, …, nan, nan, nan],
       …,
       [nan, nan, nan, …, nan, nan, nan],
       [nan, nan, nan, …, nan, nan, nan],
       [nan, nan, nan, …, nan, nan, nan]])
```

[11]:
```
# Now let's do the Tremaine Weinberg step. Defining slits of 5 arcsec.
# The programme will align the axes using the PA of the line of nodes as␣
 ↪provided.
# The warning is just about nan and 0's being used in the division.
mydisc.tremaine_weinberg(slit_width=5.0, map_name="MUSE")
```

```
/home/soft/python/pydisc/src/pydisc/density_wave.py:114: RuntimeWarning: invalid
value encountered in true_divide
  fV_err = fV * np.sqrt((eFlux / Flux)**2 + (eVel / Vel)**2)
```

[12]:
```
# And you can now look at the result
print("Slicings: ", mydisc.slicings)
# Looking at the slicings
print("MUSE Slicing", mydisc.slicings['MUSE'])
```

```
# and its content
print("Yedges = ", mydisc.slicings['MUSE'].yedges)
print("Nslits?: ", mydisc.slicings['MUSE'].nslits)
print("Omega sinus(inclin) of TW method", mydisc.slicings['MUSE'].Omsini_tw)
```

```
Slicings:  {'MUSE': <pydisc.disc_data.Slicing object at 0x7f53631b4890>}
MUSE Slicing <pydisc.disc_data.Slicing object at 0x7f53631b4890>
Yedges =  [-453.5        -448.54371585 -443.58743169 -438.63114754 -433.67486339
 -428.71857923 -423.76229508 -418.80601093 -413.84972678 -408.89344262
 -403.93715847 -398.98087432 -394.02459016 -389.06830601 -384.11202186
 -379.1557377  -374.19945355 -369.2431694  -364.28688525 -359.33060109
 -354.37431694 -349.41803279 -344.46174863 -339.50546448 -334.54918033
 -329.59289617 -324.63661202 -319.68032787 -314.72404372 -309.76775956
 -304.81147541 -299.85519126 -294.8989071  -289.94262295 -284.9863388
 -280.03005464 -275.07377049 -270.11748634 -265.16120219 -260.20491803
 -255.24863388 -250.29234973 -245.33606557 -240.37978142 -235.42349727
 -230.46721311 -225.51092896 -220.55464481 -215.59836066 -210.6420765
 -205.68579235 -200.7295082  -195.77322404 -190.81693989 -185.86065574
 -180.90437158 -175.94808743 -170.99180328 -166.03551913 -161.07923497
 -156.12295082 -151.16666667 -146.21038251 -141.25409836 -136.29781421
 -131.34153005 -126.3852459  -121.42896175 -116.4726776  -111.51639344
 -106.56010929 -101.60382514  -96.64754098  -91.69125683  -86.73497268
  -81.77868852  -76.82240437  -71.86612022  -66.90983607  -61.95355191
  -56.99726776  -52.04098361  -47.08469945  -42.1284153   -37.17213115
  -32.21584699  -27.25956284  -22.30327869  -17.34699454  -12.39071038
   -7.43442623   -2.47814208    2.47814208    7.43442623   12.39071038
   17.34699454   22.30327869   27.25956284   32.21584699   37.17213115
   42.1284153    47.08469945   52.04098361   56.99726776   61.95355191
   66.90983607   71.86612022   76.82240437   81.77868852   86.73497268
   91.69125683   96.64754098  101.60382514  106.56010929  111.51639344
  116.4726776   121.42896175  126.3852459   131.34153005  136.29781421
  141.25409836  146.21038251  151.16666667  156.12295082  161.07923497
  166.03551913  170.99180328  175.94808743  180.90437158  185.86065574
  190.81693989  195.77322404  200.7295082   205.68579235  210.6420765
  215.59836066  220.55464481  225.51092896  230.46721311  235.42349727
  240.37978142  245.33606557  250.29234973  255.24863388  260.20491803
  265.16120219  270.11748634  275.07377049  280.03005464  284.9863388
  289.94262295  294.8989071   299.85519126  304.81147541  309.76775956
  314.72404372  319.68032787  324.63661202  329.59289617  334.54918033
  339.50546448  344.46174863  349.41803279  354.37431694  359.33060109
  364.28688525  369.2431694   374.19945355  379.1557377   384.11202186
  389.06830601  394.02459016  398.98087432  403.93715847  408.89344262
  413.84972678  418.80601093  423.76229508  428.71857923  433.67486339
  438.63114754  443.58743169  448.54371585  453.5       ]
Nslits?:  183
Omega sinus(inclin) of TW method [ 1.60138439e+00  5.81666013e-01
1.09223844e+00  1.13889107e+00
```

```
 1.28640187e+00  1.41789567e+00  1.61600995e+00  1.89170327e+00
 2.17694775e+00  1.84294469e+00  2.07268205e+00  2.83561306e+00
 3.07668532e+00  2.94893032e+00  3.15404790e+00  3.21187224e+00
 3.00960209e+00  2.95929843e+00  3.14109119e+00  2.96410390e+00
 3.02598723e+00  3.66776260e+00  4.12119283e+00  3.43925555e+00
 2.64493352e+00  2.86972970e+00  2.49831874e+00  4.31817640e+00
-1.31775782e+00 -5.00421127e-01 -1.23585138e+00  3.55382096e+00
 2.98415166e+00  3.27506900e+00  3.50245330e+00  3.39129348e+00
 3.78478216e+00  3.84819736e+00  3.31404858e+00  3.18747688e+00
 3.16425434e+00  3.05899521e+00  2.91543243e+00  2.82008955e+00
 2.94399716e+00  2.87392501e+00  2.75315199e+00  2.52853410e+00
 2.69539589e+00  2.70617544e+00  2.56676546e+00  2.62079978e+00
 2.29008481e+00  2.28424723e+00  2.18303260e+00  1.95991606e+00
 1.35639657e+00  1.22609251e+00  6.76509022e-01  5.97649978e-01
 8.57888678e-01  4.01108369e-01  2.78479263e-01 -4.75741331e-02
-4.83063222e-01 -4.23571705e-01 -4.63065155e-01 -8.44697625e-01
-1.73764065e+00 -1.37582878e+00 -1.64284603e+00 -2.65960492e+00
-3.63591860e+00 -4.67667802e+00 -5.41098768e+00 -5.84873282e+00
-7.11309863e+00 -8.32553319e+00 -9.11351495e+00 -1.21849365e+01
-1.44751347e+01 -1.68702091e+01 -2.21220417e+01 -3.16844033e+01
-6.70428358e+01  1.57748312e+02  4.13143584e+01  3.78159435e+01
 4.13610084e+01  4.59038404e+01  5.58368653e+01  5.61944260e+01
 5.53301001e+01  4.89515876e+01  4.48815950e+01  4.29960001e+01
 3.73727420e+01  3.00255100e+01  2.55518753e+01  2.32267146e+01
 2.16928955e+01  1.99472354e+01  1.89922501e+01  1.83575947e+01
 1.71275422e+01  1.66905867e+01  1.58977803e+01  1.52307989e+01
 1.52057422e+01  1.47450830e+01  1.43507833e+01  1.37593019e+01
 1.37097963e+01  1.35006706e+01  1.32799338e+01  1.31750212e+01
 1.30924089e+01  1.30051123e+01  1.28698013e+01  1.25829141e+01
 1.17600554e+01 -4.85470418e+00  9.98173429e+00  1.21591084e+01
 1.21127599e+01  9.94051643e-01 -7.14393257e+02 -4.12492598e+00
 1.17078168e+01  1.13966305e+01  1.15084927e+01  1.17444767e+01
 1.17346974e+01  1.18005511e+01  1.17402053e+01  1.18871822e+01
 1.19003689e+01  1.20521672e+01  1.18605579e+01  1.14458160e+01
 1.14067918e+01  1.13649085e+01  1.14346512e+01  1.07843007e+01
 1.03919124e+01  1.05298274e+01  1.05631445e+01  1.06338715e+01
 1.06502086e+01  1.05329283e+01  1.02780615e+01  1.02799227e+01
 1.03927025e+01  1.07656466e+01  1.08558474e+01  1.07704459e+01
 1.08610448e+01  1.11347068e+01  1.15201845e+01  1.21620432e+01
 1.23523263e+01  1.21587059e+01  1.19063346e+01  1.17873405e+01
 1.20004226e+01  1.20036116e+01  1.21302081e+01  1.22502449e+01
 1.24351038e+01  1.25105342e+01  1.26226332e+01  1.26092751e+01
 1.27971391e+01  1.26567256e+01  1.24406885e+01  1.28234662e+01
 1.34039075e+01  1.38850028e+01  1.45161748e+01  1.18368487e+01
 8.45413158e+00  7.54698290e+00  6.04143985e+00]
```

2.2 3.2 Torques

Now let's consider the other class inheriting from the GalacticDisc class, namely: GalacticTorque, which itself uses TorqueMap(s).

[13]:
```
from pydisc.torques import GalacticTorque

n6951folder = "/soft/python/pytorque/examples/data_6951/"
mass6951 = pyfits.getdata(n6951folder+"r6951nicmos_f160w.fits")
gas6951 = pyfits.getdata(n6951folder+"co21-un-2sigma-m0.fits")
gas6951 = gas6951.reshape(gas6951.shape[0]*gas6951.shape[1], gas6951.shape[2])
vc6951 = "rot-co21un-01.tex"
```

[14]:
```
t51 = GalacticTorque(vcfile_name=n6951folder+vc6951, vcfile_type="ROTCUR",
                     datamass=mass6951, datacomp=gas6951, Xcenmass=178.0,␣
 ↪Ycenmass=198.0,
                     Xcencomp=148.0, Ycencomp=123.0, inclination=41.5,␣
 ↪distance=35.0,
                     PA_nodes=138.7, pixel_scalecomp=0.1, pixel_scalemass=0.025)
```

```
WARNING: X or Y not provided. Using Pixel XY grid.
INFO: attaching map mass
WARNING: X or Y not provided. Using Pixel XY grid.
INFO: attaching map comp
INFO: Adding the provided Vc file
INFO[match_datamaps]: Creating the first map massgrid and attaching first
datamap dmass01
INFO[match_datamaps]: attaching the datamap dcomp01 to map massgrid
INFO: attaching map massgrid
INFO[match_comp_mass]: new map is massgrid
```

You can then see what is in t51 structure:

[15]:
```
t51.maps
```

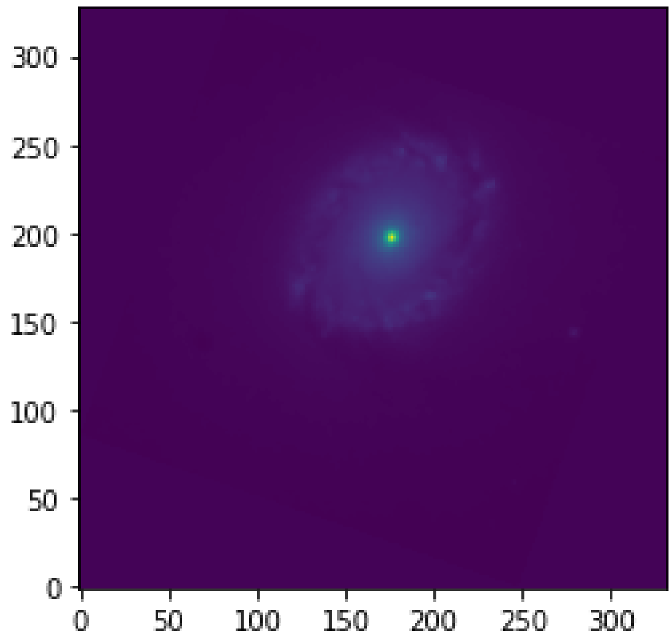
[15]:
```
{'mass': <pydisc.disc_data.Map at 0x7f53627f8690>,
 'comp': <pydisc.disc_data.Map at 0x7f5397749350>,
 'massgrid': <pydisc.disc_data.Map at 0x7f53627454d0>}
```

[16]:
```
t51.maps['mass'].dmaps
```

[16]:
```
{'mass01': <pydisc.disc_data.DataMap at 0x7f539636e310>,
 'dmass01': <pydisc.disc_data.DataMap at 0x7f53977497d0>}
```

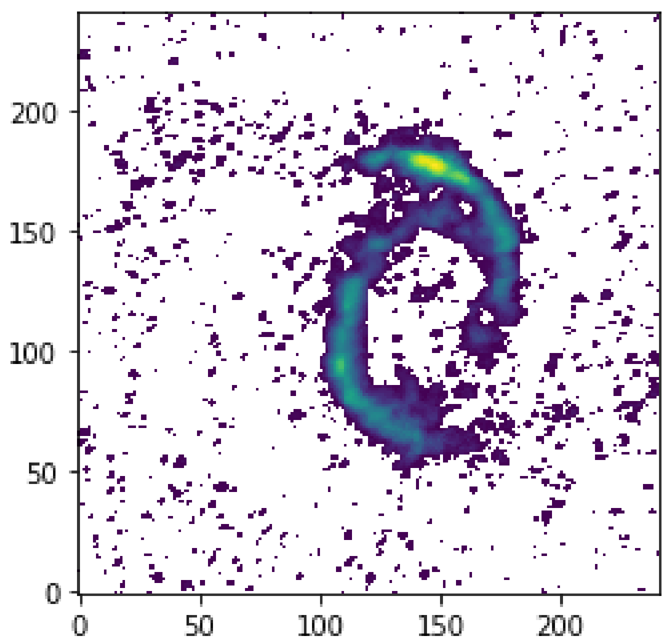
[17]:
```
from matplotlib import pyplot as plt
plt.imshow(t51.maps['mass'].dmaps['mass01'].data)
```

[17]:
```
<matplotlib.image.AxesImage at 0x7f5362693850>
```



[18]:
```
plt.imshow(t51.maps['comp'].dmaps['comp01'].data)
```

[18]: <matplotlib.image.AxesImage at 0x7f53625bc8d0>



[19]:
```
# Now running the torques
t51.run_torques()
```

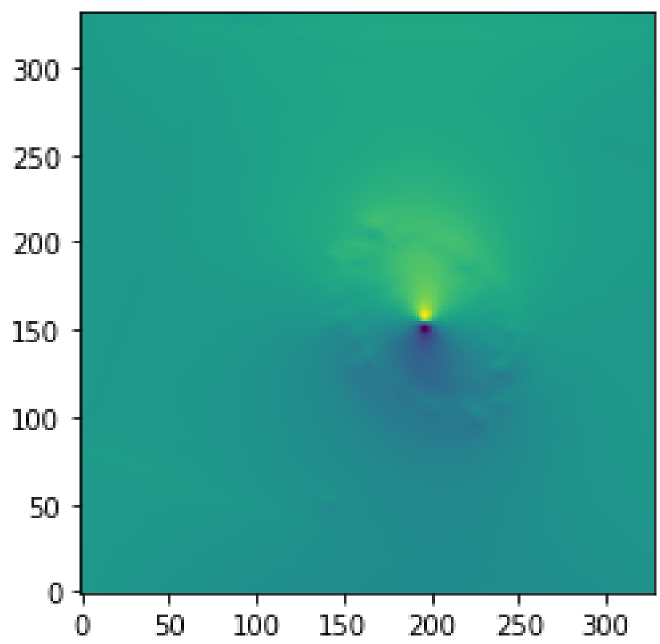
```
Deriving the radial profile …


/home/soft/python/pydisc/src/pydisc/fit_functions.py:45: RuntimeWarning: invalid
value encountered in greater
  sel_good = (flux > 0.)
/home/soft/python/pydisc/src/pydisc/gravpot_functions.py:116: RuntimeWarning:
invalid value encountered in greater
  goodw = (weights > 0.).ravel()
```

[20]:
```
t51.tmaps
```

[20]:
```
{'Torq01': <pydisc.torques.TorqueMap at 0x7f53625fa190>}
```

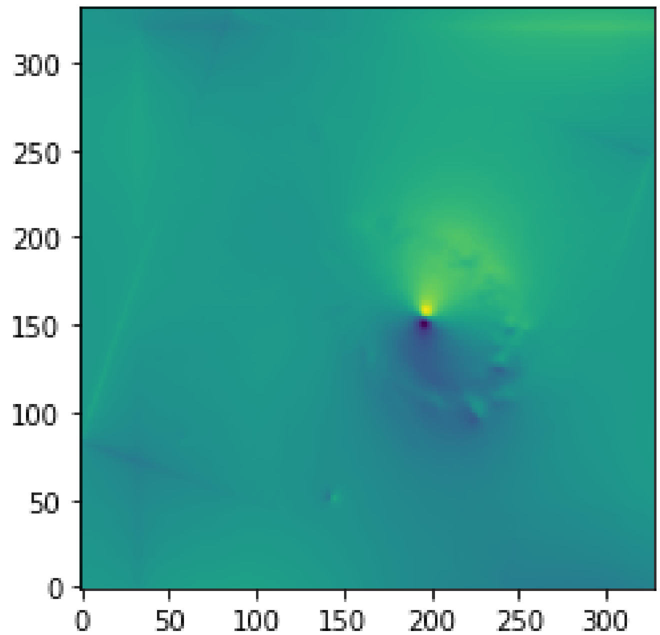
[21]:
```
plt.imshow(t51.tmaps['Torq01'].Fx)
```

[21]:
```
<matplotlib.image.AxesImage at 0x7f5362531f90>
```



[24]:
```
plt.imshow(t51.tmaps['Torq01'].torque_map)
```

[24]:
```
<matplotlib.image.AxesImage at 0x7f5360178550>
```


300
250
200
150
100
50
0
0
50
100
150
200
250
300