pyXsurf: an open-source library for analysis of surface metrology data

Vincenzo Cotroneo
INAF – Osservatorio Astronomico di Brera

INTRODUCTION

The pyXurf library consists in a set of Python-powered routines and classes, operating on data with coordinates and enabling to perform complex actions on data in a simpler way.
This is useful for example to handle data with different sampling or a mismatch in x-y positions.

Overview

The main class `Data2D` represents 2D data linked to `x` and `y` coordinates.

A `Data2D` object can be initialized in the most general way by providing a matrix of 2-dimensional data and coordinates, and conversely be exported as `data, x, y`.

```
D = Data2D(data,x,y)      # load data in the class

dd, xx, yy = D()          # export data as np.array
```

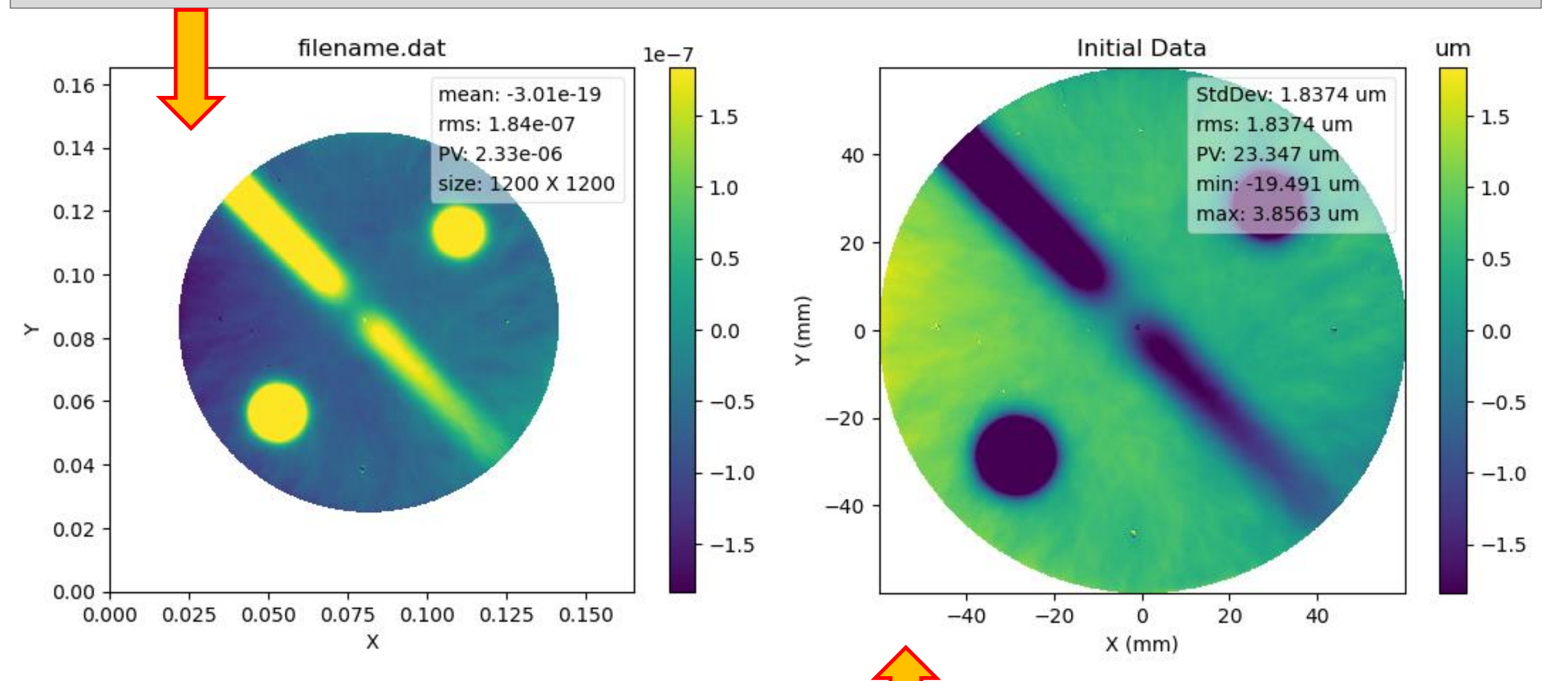
A number of methods can now be called on the data object to perform analysis and operations.

Operations

Additional options can passed at inizialization to speficy object properties.
A number of methods can be called on the object to perform analysis and operations.

```
D = Data2D('filename.dat')  #no options, the string input
is interpreted as filename, the format is guessed from
extension

D.plot() # default options
```



```
D = Data2D(file,
    reader = matrixdat_reader, #format of file, auto if
none
    units=['mm','mm','um'],            # gives units
    scale=[1000.,1000.,-10000000.],     # and scale
    center=[0,0],                      #set centering
    name='Initial Data',
    strip=True)          #remove external invalid data

D.plot(stats=[1,2,3,4,5]) # select stats to show
```

Comparison of default options (top) for loading and plotting data and customized options (bottom). If format is not specified the reader function try to guessed and import available metadata. Note how the custom options incorporate units, flip the z axis, and crop invalid data. Statistics can also be tuned in plot.

Python Environment

As usual in Python, objects can be inspected to consult documentation or inspect available methods.
The function interface was kept from common Python functions (e.g. `np.genfromtxt`, `plt.plot`, `savefig`, ..) and should be easy to learn for the user already familiar with the language.

Docstring

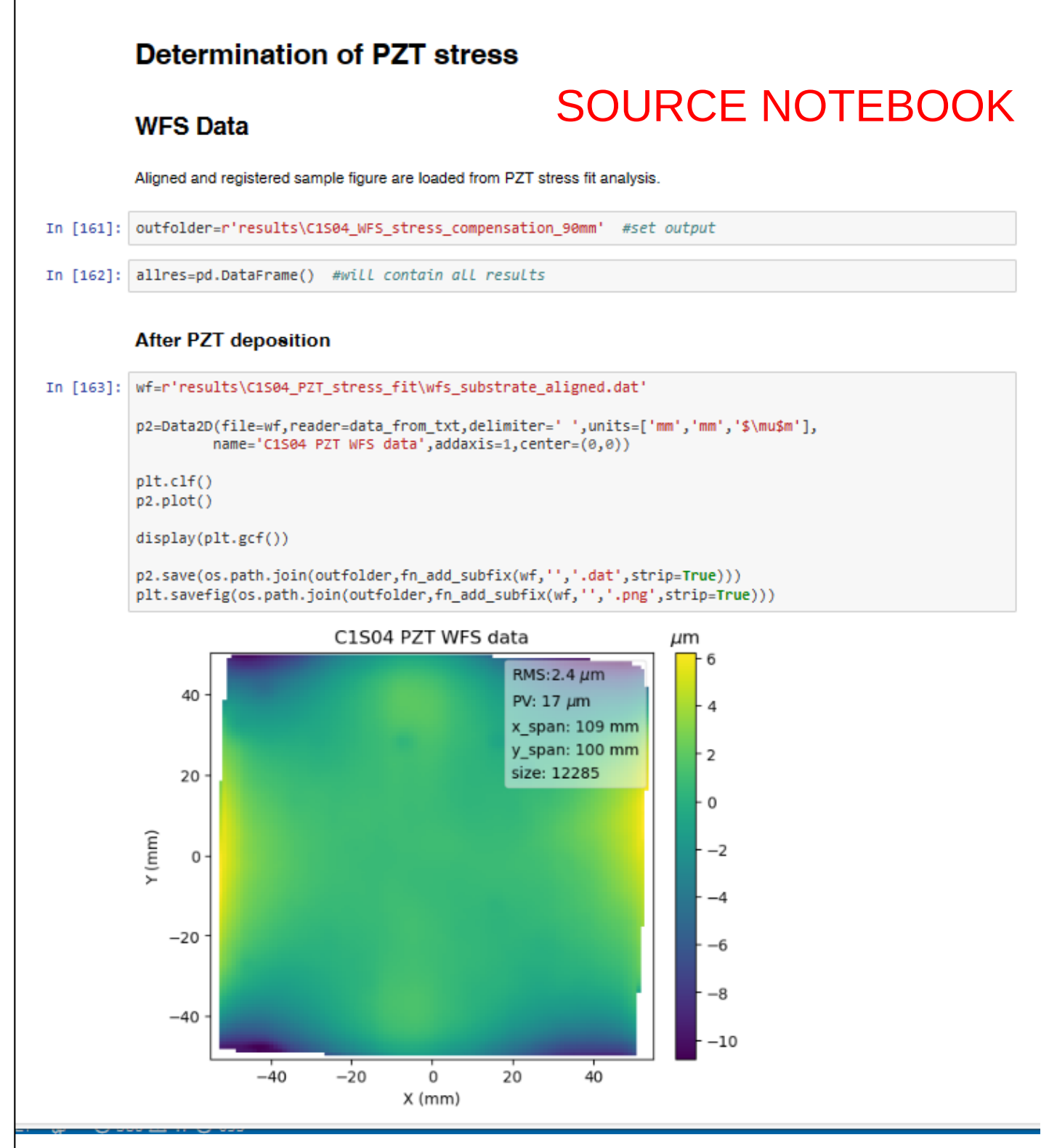
```
>> Data2D?


Init signature:
Data2D(
data=None,
x=None,
y=None,
file=None,
reader=None,
units=None,
name=None,
*args,
**kwargs,
)
Docstring:
A class containing 2d data 
Init docstring:
can be initialized with dat
File: c:\users\kovor\docume
Subclasses: PSD2D
```

Methods

```
add_markers        update_wrapper
align_interactive
apply_to_data
apply_transform
copy
crop
extract_profile
histostats
level
load
merge
plot
printstats
projection
psd
remove_nan_frame
remove_outliers
resample
rot90
rotate
save
shift
slope
stats
std
topoints
transpose
tv
```

Python functionalities make easy to implement the workflow on different frontends, from GUI to notebooks (can be exported to report), command line, script and config files, interactive documents or slides, and nearly any form of interface.


SOURCE NOTEBOOK

Determination of PZT stress

WFS Data

Aligned and registered sample figure are loaded from PZT stress fit analysis.

```
In [161]: outfolder=r'results\C1S04_WFS_stress_compensation_90mm'  #set output

In [162]: allres=pd.DataFrame()  #will contain all results
```

After PZT deposition

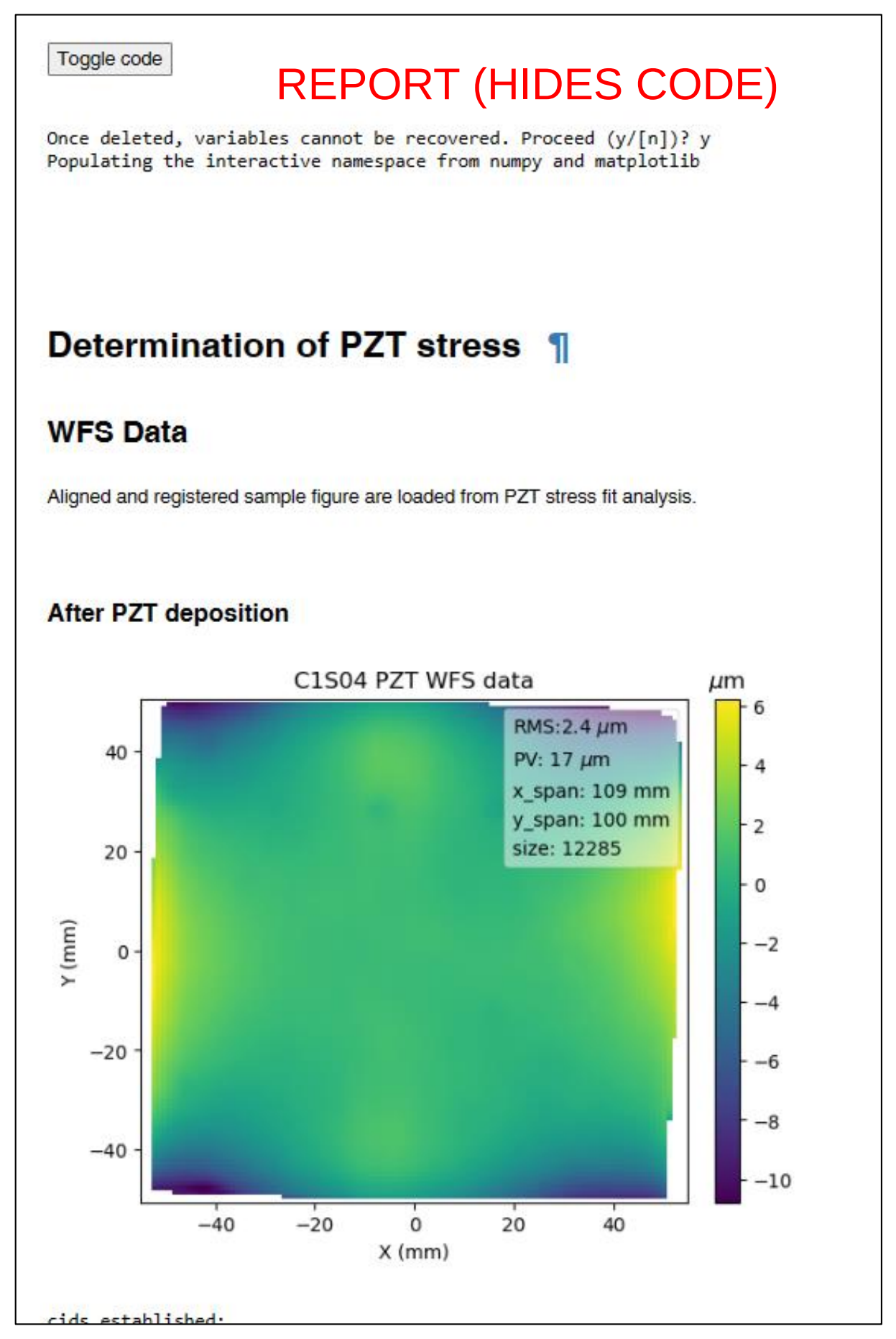
```
In [163]: wf=r'results\C1S04_PZT_stress_fit\wfs_substrate_aligned.dat'

          p2=Data2D(file=wf,reader=data_from_txt,delimiter=' ',units=['mm','mm','$\mu$m'],
                  name='C1S04 PZT WFS data',addaxis=1,center=(0,0))

          plt.clf()
          p2.plot()

          display(plt.gcf())

          p2.save(os.path.join(outfolder,fn_add_subfix(wf,'','.dat',strip=True)))
          plt.savefig(os.path.join(outfolder,fn_add_subfix(wf,'','.png',strip=True)))
```


Toggle code

REPORT (HIDES CODE)

```
Once deleted, variables cannot be recovered. Proceed (y/[n])? y
Populating the interactive namespace from numpy and matplotlib
```

Determination of PZT stress ¶

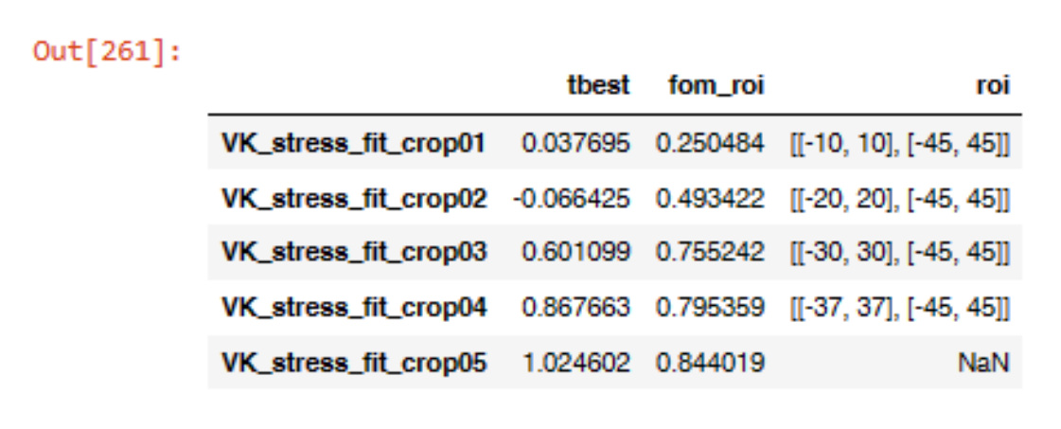
WFS Data

Aligned and registered sample figure are loaded from PZT stress fit analysis.

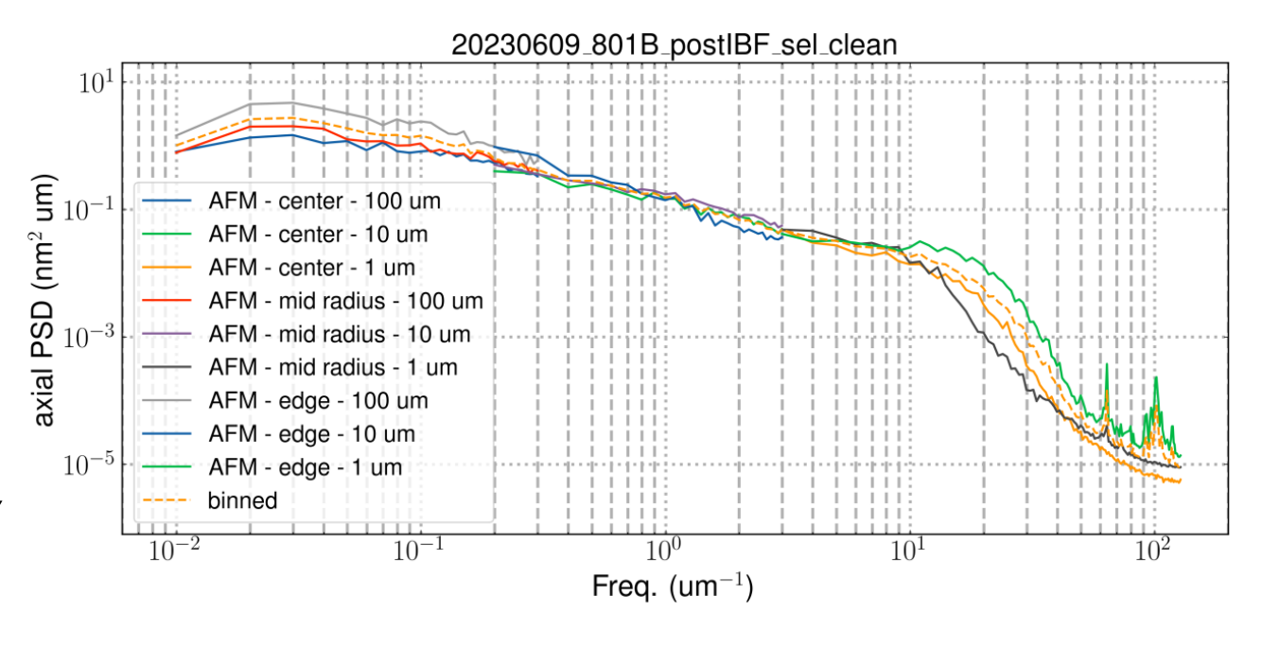
After PZT deposition

```
cids established:
```

Traceable analysis and automatic report generation (including pdf, slides, or interactive documents) with Jupyter notebook features


Out[261]:

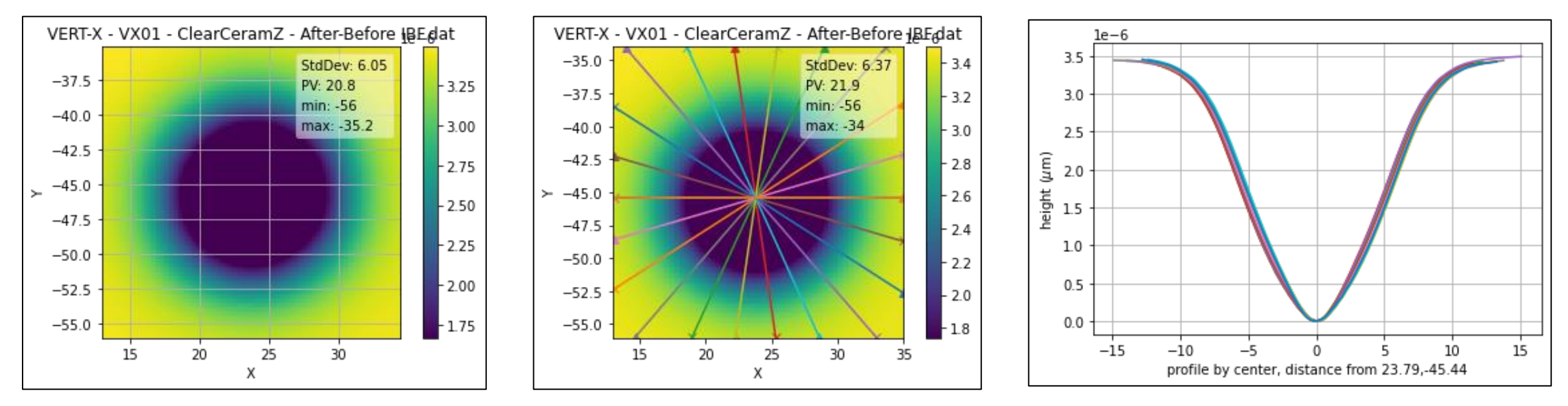
	tbest	fom_roi	roi
VK_stress_fit_crop01	0.037695	0.250484	[[-10, 10], [-45, 45]]
VK_stress_fit_crop02	-0.066425	0.493422	[[-20, 20], [-45, 45]]
VK_stress_fit_crop03	0.601099	0.755242	[[-30, 30], [-45, 45]]
VK_stress_fit_crop04	0.867663	0.795359	[[-37, 37], [-45, 45]]
VK_stress_fit_crop05	1.024602	0.844019	NaN



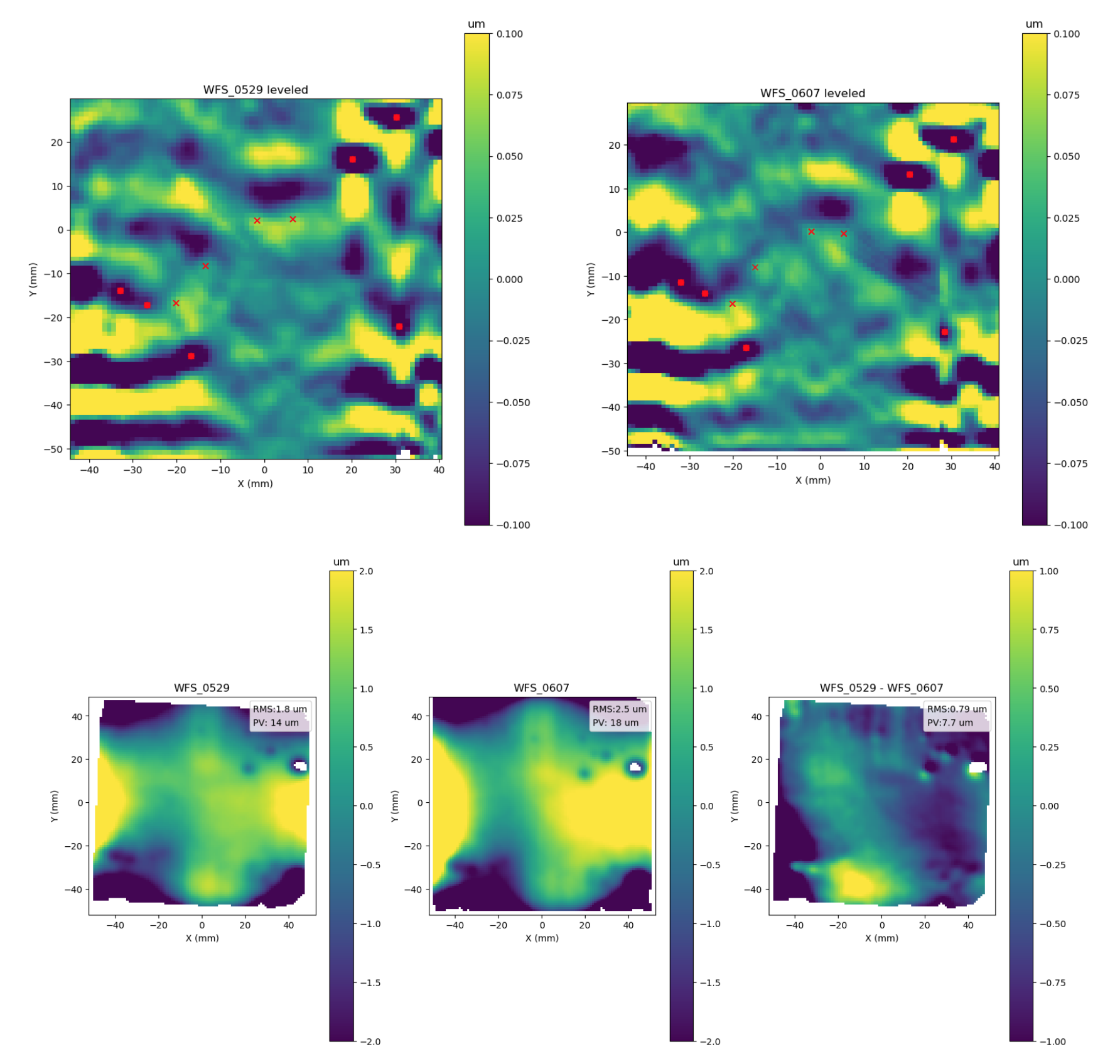
Pandas library for data analysis enables easy data processing, filtering, and visulalization.

Other functions

Crop and profile extraction



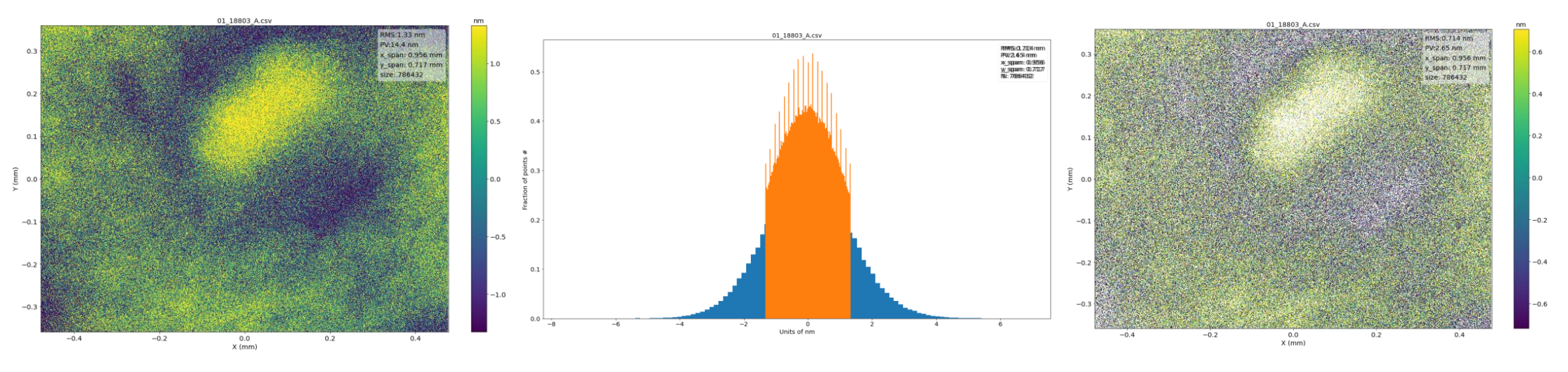
Advanced leveling, interactive (point and click) alignment on fiducials , and data difference (as algebraic operation).



```
d1l = data1.level((10,0)) # remove high order on y axis
d2l = data2.level((10,0)) # to highlight features
mref,mtrans = data1.align_interactive(data2)# return transform
data1_trans = data1.apply_transform(mtrans)  # apply transform
plt.figure()
diff = data1-data2_trans    # difference
```

Histogram analysis and outliers removal

```
h1 = d.histostats()
h2 = d.remove_outliers(nsigma=1).histostats()
d.histostats(bins=h2[1])
```



AND MORE… Point cloud analysis, form fit, PSD analysis, slope analysis, profile handling, profile analysis and stitching, etc..

Visit the links below to see more!!

CONCLUSIONS

This project comes from the work of a single person over several years, it now reached some maturity, but it is still chasing Python best-practices, and it is in a good state to be released to the community and seek for support.

The ongoing improvements to this project are funded by INAF "Bando per innovazione tecnologica", see links below to stay updated.

Any contribution is welcome!!

vincenzo.cotroneo@inaf.it

DOCS
https://pyxsurf.readthedocs.io

GitHub **https://github.com/`**
vincenzooo/pyXsurf