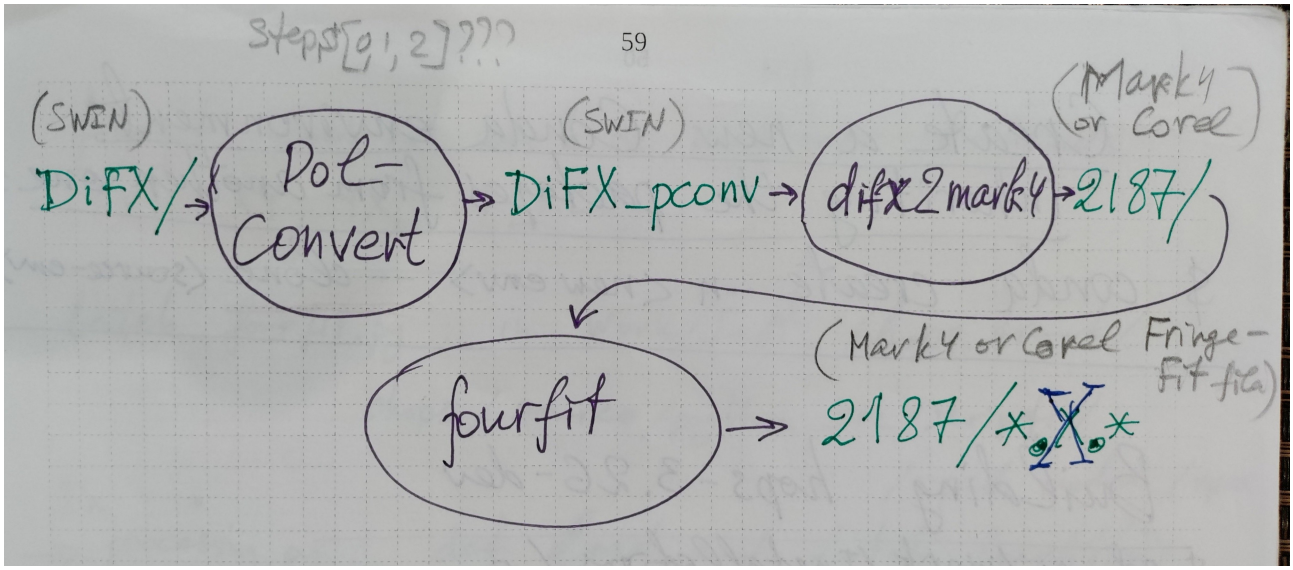
PolConvert Input and Output

Generally, PolConvert inputs the directory with SWIN (or DiFX) data in linear polarization, and saves the results of conversion into circular polarization in SWIN (or DiFX) format in another directory, say, DiFX_pconv/. The latter should be fed to difx2mark4 to convert it into the Mark4 (or Corel) format saved in a directory with a 4-digit name (in my case, I chose 2187/). Finally, the circularly-polarized Mark4 (or Corel) files should be given to fourfit for fringe-fitting.

Below is a photo of the workflow chart I once made for myself.



Obtaining and Installing PolConvert

It can be downloaded as a zip file from https://github.com/marti-vidal-i/PolConvert .
Unzipping creates the PolConvert-main/ directory. The file INSTALL provides the detailed instructions. At the very bottom it has
IMPORTANT NOTES:

1.- You want to use the **standalone** version for the EU-VGOS calibration.
2.- You will ALWAYS have to load the PCONV conda environment before you run PolConvert.

So, no CASA bindings needed.

I installed it on my workstation leonid2.haystack.mit.edu, at /home/benkev/Work/PolConvert/, but it should be installed on demi. Ask Tim Morin?

Running PolConvert

The information below may be classified. At least, I couldn't find it in any open source. But well, I'm sharing it (not disclosing the source, of course).

In the subdirectory **PolConvert-main/EU-VGOS/** find the file **master_script_python3_MP.py**. This is the script doing the PolConversion. It includes fairly good instructions. You need to set a few main parameters right in the text of script:
- experiment name;
- directory with the original DiFX (or SWIN) data with **linear** polarization;
- destionation directory of the PolConverted data (DiFX (or SWIN) + metadata) with **circular** polarization;
- the step number in the mysteps list.

The PolConversion is fulfilled in 7 steps, from 0 up to 6. Each step needs setting its number in mysteps and running **master_script_python3_MP.py** again.

There are several more parameters to set in the script. However, I was lucky to find in its original text **EXPNAME = 'vo2187'**. I assumed the parameters were already correctly set for this particular experiment, so I left them intact. Probably, this needs to be double-checked.

Further Conversions and Fringe-Fit

The circularly-polarized DiFX (or SWIN) data generated by PolConvert are converted into circularly-polarized Mark4 (or Corel) format with `difx2mark4.`

Instead of direct usage of fourfit on the whole set of Mark4 (or Corel) data, using the Python script `batch_fourfit.py` from the HOPS package can be recommended as more convenient. However, the script needs to be fixed in odder to “understand” the circular polarization notations. Specifically, in the code of batch_fourfit.py the piece

```
if pp not in ['XX', 'YY', 'XY', 'YX', 'I']:
    print("error: ", pp, "not understood, polarization products
           must be an element from: {XX, YY, XY, YX, I}")
    sys.exit(1)
```

should be replaced with

```
if pp not in ['XX', 'YY', 'XY', 'YX', 'I',
              'LL', 'LR', 'RL', 'RR', 'LL+RR']:
    print("error: ", pp, "not understood, polarization products
           must be an element from: {XX, YY, XY, YX, I, LL, LR,
           RL, RR, LL+RR}")
    sys.exit(1)
```

Again, it is known from trustworthy sources that “LL+RR” is an euphemism for pseudo-Stokes “I” for the circularly polarized data.

Note

After PolConversion of the VO2187 data, I noticed that the Yebes station (Y, Yj, RAEGYEB, 13m at Yebes, Spain) was completely omitted in the resultant data. The fringe-fit Mark4 data lack all the baselines with the Y station. I have not investigated the issue yet. It needs examining the PolConvert source codes.