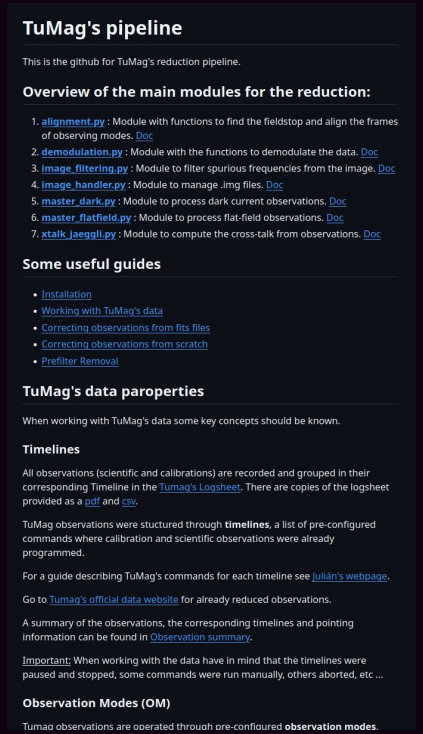
TuMag
Hands on reduction

IAA TEAM

TuMag's pipeline GitHub

All tools required for the reduction are included in TuMag's pipeline github page:

1. Scripts
2. Documentation.
3. Guides.
4. Links to relevant pages.



TuMag's pipeline

This is the github for TuMag's reduction pipeline.

Overview of the main modules for the reduction:

1. **alignment.py** : Module with functions to find the fieldstop and align the frames of observing modes. Doc
2. **demodulation.py** : Module with the functions to demodulate the data. Doc
3. **image_filtering.py** : Module to filter spurious frequencies from the image. Doc
4. **image_handler.py** : Module to manage .img files. Doc
5. **master_dark.py** : Module to process dark current observations. Doc
6. **master_flatfield.py** : Module to process flat-field observations. Doc
7. **xtalk_jaeggli.py** : Module to compute the cross-talk from observations. Doc

Some useful guides

- Installation
- Working with TuMag's data
- Correcting observations from fits files
- Correcting observations from scratch
- Prefilter Removal

TuMag's data paroperties

When working with TuMag's data some key concepts should be known.

Timelines

All observations (scientific and calibrations) are recorded and grouped in their corresponding Timeline in the Tumag's Logsheet. There are copies of the logsheet provided as a pdf and csv.

TuMag observations were stuctured through **timelines**, a list of pre-configured commands where calibration and scientific observations were already programmed.

For a guide describing TuMag's commands for each timeline see Julián's webpage.

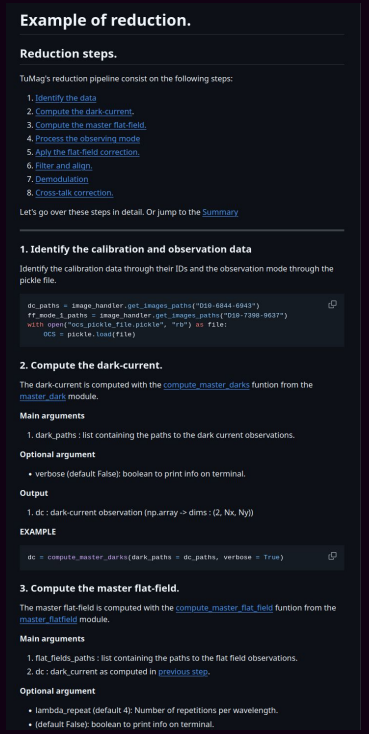
Go to Tumag's official data website for already reduced observations.

A summary of the observations, the corresponding timelines and pointing information can be found in Observation summary.

Important: When working with the data have in mind that the timelines were paused and stopped, some commands were run manually, others aborted, etc ...

Observation Modes (OM)

Tumag observations are operated through pre-configured **observation modes**.



Example of reduction.

Reduction steps.

TuMag's reduction pipeline consist on the following steps:

1. Identify the data
2. Compute the dark-current.
3. Compute the master flat-field.
4. Process the observing mode
5. Aply the flat-field correction.
6. Filter and align.
7. Demodulation
8. Cross-talk correction.

Let's go over these steps in detail. Or jump to the Summary

---

1. Identify the calibration and observation data

Identify the calibration data through their IDs and the observation mode through the pickle file.

```
dc_paths = image_handler.get_images_paths("DID-6844-6943")
ff_mode_1_paths = image_handler.get_images_paths("DID-7398-9637")
with open("ocs_pickle_file.pickle", "rb") as file:
    OCS = pickle.load(file)
```

2. Compute the dark-current.

The dark-current is computed with the compute_master_darks funtion from the master_dark module.

Main arguments

1. dark_paths : list containing the paths to the dark current observations.

Optional argument

- verbose (default False): boolean to print info on terminal.

Output

1. dc : dark-current observation (np.array -> dims : (2, Nx, Ny))

EXAMPLE

```
dc = compute_master_darks(dark_paths = dc_paths, verbose = True)
```

3. Compute the master flat-field.

The master flat-field is computed with the compute_master_flat_field funtion from the master_flatfield module.

Main arguments

1. flat_fields_paths : list containing the paths to the flat field observations.
2. dc : dark_current as computed in previous step.

Optional argument

- lambda_repeat (default 4): Number of repetitions per wavelength.
- (default False): boolean to print info on terminal.

https://github.com/PabloSGN/TuMags_Reduction_Pipeline

TuMag Observing Modes

(Info included in the Readme file from github)

TuMag Observations are organized through observing modes, wich determine:

1. The observed spectral line.
2. The modulation scheme:
   a. Vectorial (Full Stokes)
   b. Longitudinal (I and V)
   c. No modulation.
3. Number of wavelengths

TuMag observing modes summary.

Observing Mode	Filter	N° wavelengths	Modulation scheme
0p	517	12	Vectorial
0s	517	12	No modulation
1	517	10	Vectorial
2.02	525.02	8	Vectorial
2.06	525.06	8	Vectorial
3.02	525.02	5	Longitudinal
3.06	525.06	5	Longitudinal
4	517	3	Vectorial
5.02	525.02	3	Vectorial
5.06	525.06	3	Vectorial

https://github.com/PabloSGN/TuMags_Reduction_Pipeline

Fits files description.

(Info included in “working with fits files” guide from github)

TuMag fits files main properties:

1. One observing mode per fits file.
2. Some reduction steps missing (determined by the Reduction level of the file)

Main reduction steps:

1. Dark current + Flat-field correction (level 0.5)
2. Alignment of modulations and cameras (level 0.8)
3. Demodulation + Cross-talk correction. (level 1.0)

- If wavefront reconstruction (PD) applied → + 0.1:

4. Dark current + Flat-field correction + PD (level 0.6)
5. Alignment of modulations and cameras + PD (level 0.9)
6. Demodulation + Cross-talk correction. + PD (level 1.1)

https://github.com/PabloSGN/TuMags_Reduction_Pipeline

Fits files description.

https://github.com/PabloSGN/TuMags_Reduction_Pipeline

(Info included in “working with fits files” guide from github)

TuMag fits files main properties:

1. One observing mode per fits file.
2. Some reduction steps missing (determined by the Reduction level (RL) of the file)

Reduction level summary:

Reduction Level	Flats + Darks	Wavefront reconstruction	Alignment and filtering	Stokes Parameters
0.5	✅	❌	❌	❌
0.6	✅	✅	❌	❌
0.8	✅	❌	✅	❌
0.9	✅	✅	✅	❌
1.0	✅	❌	✅	✅
1.1	✅	✅	✅	✅

Fits files description.

https://github.com/PabloSGN/TuMags_Reduction_Pipeline

(Info included in “working with fits files” guide from github)

Fits dimensions:

1. Level < 1.0 → Dims: (Ncams, Nlambda, Nmods, Nx, Ny)
2. Level => 1.0 → Dims: (Nlambda, Nmods, Nx, Ny)

Filename convention:

SUNRISE_ID_lineOM_Nlambda_stratingtime_LV_ReductionLevel_pipeline.fits

1. Sunrise ID → As defined in Obs summary table.
2. line → Fe / Mg
3. OM → TuMag observing mode.
4. Nlambda → Number of wavelengths.
5. Nmods → Number of modulations.
6. startingtime → Time for the first image of the observing mode.
7. ReductionLevel → Level of reduction.
8. pipeline → Pipeline version

Hands on data reduction

Preparation

Pipeline download and installation

(Info included in “working with fits files” guide from github)

Before start working on the data, the pipeline should be downloaded and the required libraries installed.

Preparation

Pipeline download and installation

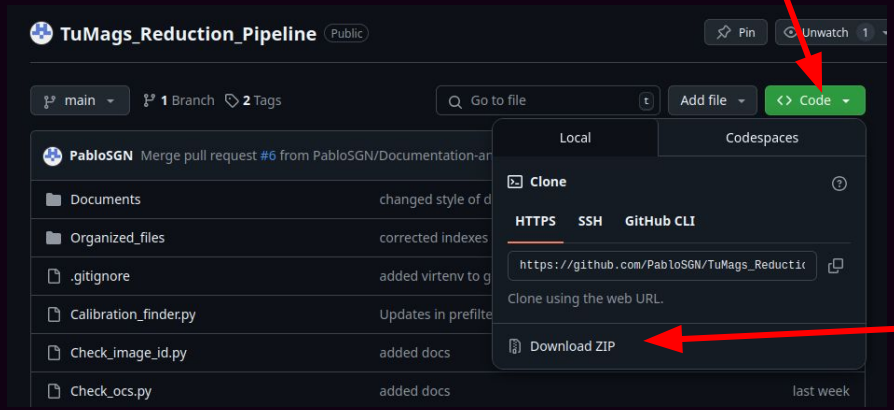
(Info included in “working with fits files” guide from github)

Before start working on the data, the pipeline should be downloaded and the required libraries installed.

2 approaches for the download:

1. Download directly from github.
2. Clone the repository.

Go to the main GitHub page and download a ZIP containing all files and extract.



https://github.com/PabloSGN/TuMags_Reduction_Pipeline

Preparation

Pipeline download and installation

(Info included in “working with fits files” guide from github)

Before start working on the data, the pipeline should be downloaded and the required libraries installed.

2 approaches for the download:

1. Download directly from github.
2. Clone the repository.

In any machine with git installed, open a terminal on the directory you want the pipeline to be stored and type:

```
git clone https://github.com/PabloSGN/TuMags_Reduction_Pipeline
```

https://github.com/PabloSGN/TuMags_Reduction_Pipeline

Preparation

Pipeline download and installation

(Info included in “working with fits files” guide from github)

Before start working on the data, the pipeline should be downloaded and the required libraries installed.

We recommend to create a virtual environment to not break your python installation!

To create and activate your python environment, type the following commands in the terminal:

```
python3 -m venv $virtenv$
source $myenv$/bin/activate
```

To install the libraries, simply run: `pip install -r requirements.txt`

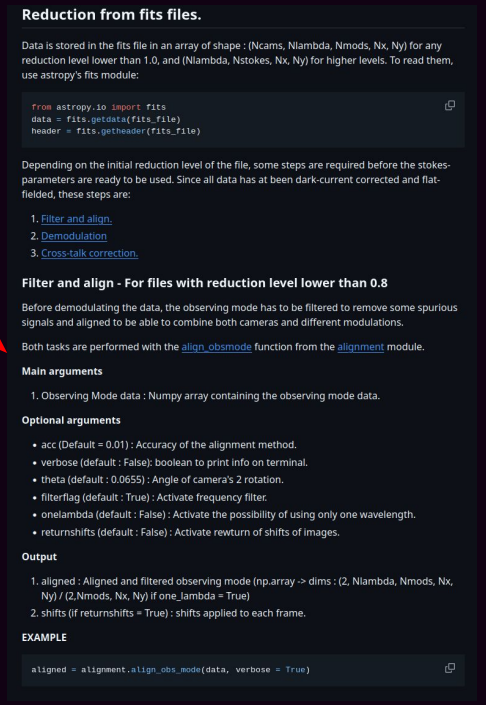
https://github.com/PabloSGN/TuMags_Reduction_Pipeline

TuMag's reduction process

(Info included in "working with fits files" guide from github)

We will provide an example of loading the data and applying the required steps for computing the corrected Stokes parameters.

This process can be followed with the "Working with fits files" guide in the GitHub page in the Documents/folder.



Reduction from fits files.

Data is stored in the fits file in an array of shape : (Ncams, Nlambda, Nmods, Nx, Ny) for any reduction level lower than 1.0, and (Nlambda, Nstokes, Nx, Ny) for higher levels. To read them, use astropy's fits module:

```
from astropy.io import fits
data = fits.getdata(fits_file)
header = fits.getheader(fits_file)
```

Depending on the initial reduction level of the file, some steps are required before the stokes-parameters are ready to be used. Since all data has at been dark-current corrected and flat-fielded, these steps are:

1. Filter and align.
2. Demodulation
3. Cross-talk correction.

Filter and align - For files with reduction level lower than 0.8

Before demodulating the data, the observing mode has to be filtered to remove some spurious signals and aligned to be able to combine both cameras and different modulations.

Both tasks are performed with the align_obsmode function from the alignment module.

Main arguments

1. Observing Mode data : Numpy array containing the observing mode data.

Optional arguments

- acc (Default = 0.01) : Accuracy of the alignment method.
- verbose (default : False): boolean to print info on terminal.
- theta (default : 0.0655) : Angle of camera's 2 rotation.
- filterflag (default : True) : Activate frequency filter.
- onelambda (default : False) : Activate the possibility of using only one wavelength.
- returnshifts (default : False) : Activate rewturn of shifts of images.

Output

1. aligned : Aligned and filtered observing mode (np.array -> dims : (2, Nlambda, Nmods, Nx, Ny) / (2,Nmods, Nx, Ny) if one_lambda = True)
2. shifts (if returnshifts = True) : shifts applied to each frame.

EXAMPLE

```
aligned = alignment.align_obs_mode(data, verbose = True)
```

The whole process involves:

1. Alignment (RL < 0.8)
2. Demodulation (RL < 1.0)
3. Stokes correction (RL < 1.0)

https://github.com/PabloSGN/TuMags_Reduction_Pipeline

Imports and loading the data

(Info included in “working with fits files” guide from github)

The first step consists on importing the required libraries and loading the data.

IMPORTS

We are going to need.

1. Astropy.io
2. sys
3. alignment.align_obsmode
4. demodulation.demodulate
5. xtalk_jaeggli.fit_muller_matrix

```
# IMPORTS

# External libraries
import sys
from astropy.io import fits

# TuMag pipeline functions
sys.path.append(pipeline_path)
from alignment import align_obsmode
from demodulation import demodulate
from xtalk_jaeggli import fit_mueller_matrix
```

TuMag’s Pipeline. External libraries

https://github.com/PabloSGN/TuMags_Reduction_Pipeline

Imports and loading the data

(Info included in “working with fits files” guide from github)

The first step consists on importing the required libraries and loading the data.

Loading data.

Read the fits file, get the data and headers.

```
# LOAD the data
obs_data = fits.getdata(fits_file)
obs_info = fits.getheader(fits_file)

nmods = obs_info["NMODS"]
nlambda = obs_info["NLAMBDA"]
line = obs_info["FW2"]
obsmode = obs_info["OBS_MODE"]
```

From the header we extract the relevant information:

a. Number of modulations
b. Number of wavelengths
c. Spectral line (used in demodulation step)
d. Observation Mode

The dimensions of the data are: (Ncams, Nmods, Nwavelengths, Nx, Ny)

Ncams : 2
Nx, Ny : 1644

Alignment and filtering

RL < 0.8

(Info included in “working with fits files” guide from github)

Data with levels lower than 0.7 must be aligned in first place.

For this purpose we use the “align_obsmode” function from the alignment.py module.

This function filters the data and aligns all modulations and both cameras.

Code:

```
aligned = align_obsmode(obs_data)
```

1. Alignment (RL < 0.8)
2. Demodulation (RL < 1.0)
3. Stokes correction (RL < 1.0)

Demodulation

0.7 < RL < 1.0

(Info included in “working with fits files” guide from github)

After alignment data can be demodulated to compute the Stokes parameters.

For this purpose we use the “demodulate” function from the demodulation.py module.

This function computes the Stokes parameters and combines both cameras (dual-beam).

Code:

```
stokes = demodulate(aligned, filt = line)
```

1. Alignment (RL < 0.7)
2. Demodulation (RL < 1.0)
3. Stokes correction (RL < 1.0)

Cross-Talk correction

0.7 < RL < 1.0

(Info included in “working with fits files” guide from github)

After demodulation data must undergo a cross-talk correction.

For this purpose we use the “fit_mueller_matrix” function from the xtalk_jaeggli.py module.

This function performs a cross-talk correction.

Code:

```
xtalk_corr, _ = fit_mueller_matrix(stokes)
```

1. Alignment (RL < 0.7)
2. Demodulation (RL < 1.0)
3. Stokes correction (RL < 1.0)

Summary

RL < 0.7

(Info included in “working with fits files” guide from github)

Whole code for RL < 0.7

```
obs_data = fits.getdata(fits_file)
obs_info = fits.getheader(fits_file)
line = obs_info["FW2"]
aligned = align_obsmode(obs_data)
stokes = demodulate(aligned, filt = line)
xtalk_corr, _ = fit_mueller_matrix(stokes)
```