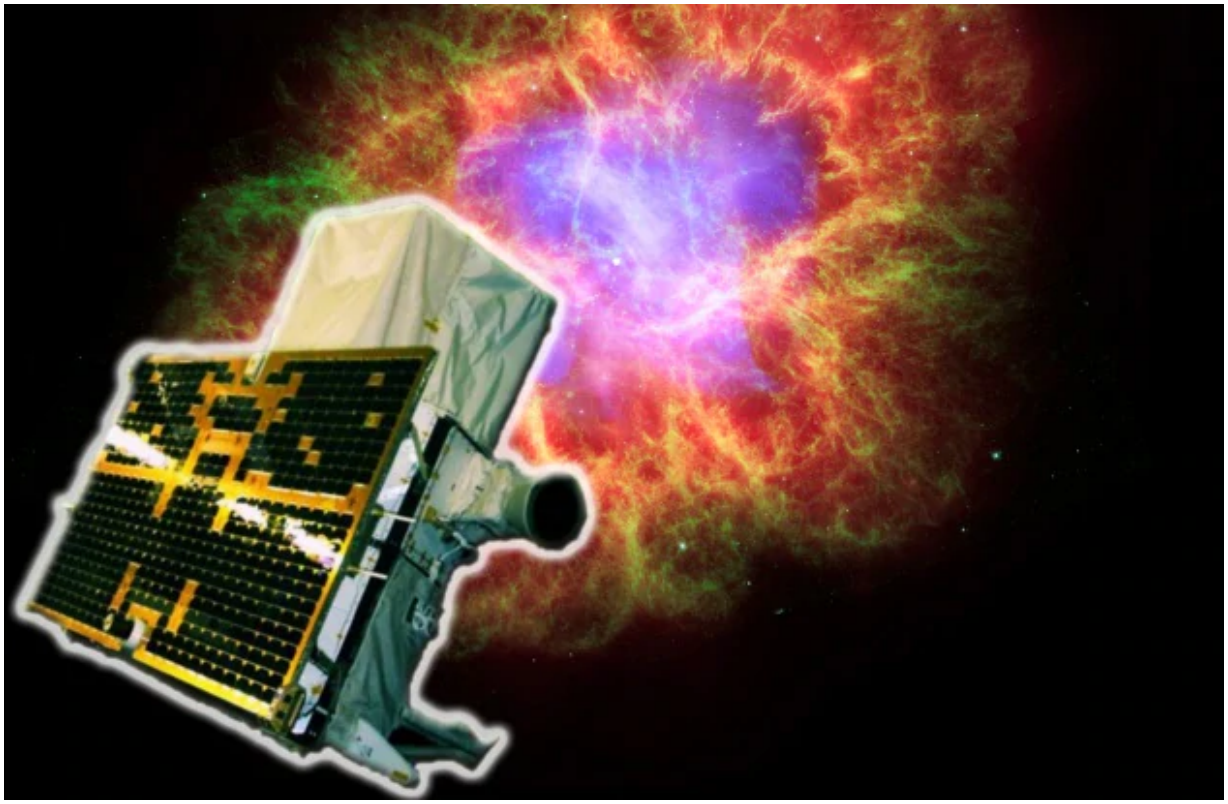
Agilepy’s documentation

Agilepy is an open-source Python package developed at INAF/OAS Bologna [https://www.oas.inaf.it] to analyse AGILE/GRID data built on top of the command-line version of the AGILE/GRID Science Tools.

The main purpose of the package is to provide an easy to use high-level Python interface to analyse AGILE/GRID data by simplifying the configuration of the tasks and ensuring straightforward access to the data. The current features are the generation and display of sky maps and light curves, the access to gamma-ray sources catalogues, the analysis to perform spectral model and position fitting including the background evaluation, the aperture photometry analysis, and the wavelet analysis. In addition, Agilepy provides an engineering interface to analyse the time evolution of the AGILE off-axis viewing angle for a chosen sky region, comparing them with Fermi/LAT off-axis evolution.



Quickstart

- Agilepy
  - AGILE
  - AGILE Science Tools
  - Agilepy analysis
  - Maximium Likelihood Estimator caveats
- Installation
  - Installation with Anaconda
  - Installation with Docker
  - Manual Installation
  - Uninstalling
  - Package distribution structure
- Quickstart guide
- Jupyter Notebooks
  - Tutorial notebooks
  - Analysis notebooks

Manual

- Download AGILE-GRID data
  - Automated download using SSDC REST Api
  - Manual download
  - Agilepy test data
- Configuration file
  - General
  - Updating the configuration options
  - Configuration options
- Sources file
  - Spectral models
  - Source library format (xml document)
- Working with sources
  - The Source abstraction
  - How to load or add new sources
  - How to select Source objects
  - How to let the source’s parameters to vary
  - How to check which source’s parameters are free to vary

    - The “multi” description of a Source object
    - How to manually inspect source’s attributes
    - How to manually change a source’s attributes
- Products
    - Sky maps
    - Result of the Maximum Likelihood Estimator
    - Light curves
- Advanced configuration
    - Selection
    - Maps
    - Model
    - mle

API

- Analysis API
    - AGBaseAnalysis
    - AGAnalysis
    - AGAnalysisWavelet
- Engineering API
    - AGEngAgileOffaxisVisibility
    - AGEngAgileFermiOffAxisVisibilityComparison
- AstroUtils API
- Source API
    - Source
    - Point Source

Science Tools

- Science Tools
- Input files of the MLE
    - Map list input files
        - *‘.maplist4’* file
    - AGILE format (text file)
- Output files of the MLE
    - *‘.source’* file
    - *‘.source’* Attributes
    - *‘.source.con’* file

    - ‘*.source.reg*’ file
    - ‘*.log*’ file
    - HTML output

Help

- Need Help
    - Known issues
- Development
    - Install the development environment
        - Anaconda
        - Docker
    - Git flow
        - Branches
        - Versioning
        - Branches names
        - Getting started
            - Development of a new feature
            - Add configuration parameters
            - Add a new science tool
            - Release of a new version
    - DevOps

Agilepy

Agilepy is an open-source Python package developed at INAF/OAS Bologna [https://www.oas.inaf.it] to analyse AGILE/GRID data built on top of the command-line version of the AGILE/GRID Science Tools.

The main purpose of the package is to provide an easy to use high-level Python interface to analyse AGILE/GRID data by simplifying the configuration of the tasks and ensuring straightforward access to the data. The current features are the generation and display of sky maps and light curves, the access to gamma-ray sources catalogues, the analysis to perform spectral model and position fitting including the background evaluation, the aperture photometry analysis, and the wavelet analysis. In addition, Agilepy provides an engineering interface to analyse the time evolution of the AGILE off-axis viewing angle for a chosen sky region, comparing them with Fermi/LAT off-axis evolution.

Agilepy is similar to Fermipy (https://fermipy.readthedocs.io/) and gammapy (https://docs.gammapy.org/) tools, providing a common way to analyse gamma-ray data.

Agilepy provides the last version of the available Science Tools (BUILD25), the H0025 instrument response functions (IRFs), and the latest version of the diffuse Galactic emission model.

Agilepy (and its dependencies) can be easily installed using Anaconda (https://www.anaconda.com/).

AGILE

AGILE (Astrorivelatore Gamma ad Immagini LEggero) is an astrophysics mission of the Italian Space Agency (ASI) operating since 2007 April devoted to gamma-ray and X-ray astrophysics. It carries two instruments observing at hard X-rays between 18 and 60 keV (Super-AGILE) and in the gamma-ray band between 30 MeV and 50 GeV (Silicon Tracker). The

payload is completed by a calorimeter (MCAL) sensitive in the 0.4–100 MeV range and an anticoincidence (AC) system. The combination of the Silicon Tracker, MCAL, and AC forms the Gamma-Ray Imaging Detector (GRID).

A set of different on-board triggers enables the discrimination of background events (mainly cosmic rays in the AGILE Low Earth Orbit) from gamma-ray events. AGILE raw data are down-linked every about 100 min to the ASI Malindi ground station in Kenya, and transmitted first to the Telespazio Mission Control Center at Fucino, and then to the AGILE Data Center (ADC), which is part of the ASI Space Science Data Center (SSDC, previously known as ASDC) and then to the INAF/OAS Bologna for the real-time analysis of data.

Main AGILE websites:

- SSDC: https://agile.ssdc.asi.it
- IAPS/Rome: http://agile.rm.iasf.cnr.it/

AGILE Apps:

- iOs iPhone: https://apps.apple.com/it/app/agilescience/id587328264
- iOs iPad: https://apps.apple.com/it/app/agilescience-for-ipad/id690462286
- Android: https://play.google.com/store/apps/details?id=com.agile.science&hl=en&gl=US
- Support wed site: http://www.agilescienceapp.it/wp/agilescienceen/

AGILE Science Tools

The AGILE/GRID Science Tools developed by the AGILE Team are used to analyse gamma-ray data starting from spacecraft files (called LOG), and the acquired events (EVT files or event list). They provide a way to generate gamma-ray counts, exposure and diffuse emission maps that are used as input for the binned maximum likelihood estimator (MLE). The analysis depends on the isotropic and Galactic diffuse emission, the gamma-ray photon statistics, and on the instrument response functions

(IRFs). IRFs are matrices that characterise the effective area (Aeff), the point spread function (PSF), and the energy dispersion probability (EDP), that depend on the direction of the incoming gamma-ray in instrument coordinates, its energy and on the on-ground event filter.

The result of the MLE is an evaluation of the presence of one or more point-like or extended sources in the sky maps: this is the essential step for the scientific results of AGILE.

A full description and characterisation of the last release of the Science Tools is available in https://arxiv.org/abs/1903.06957. Science Tools, IRFs and Galactic emission model are publicly available from the AGILE website at SSDC: https://agile.ssdc.asi.it.

Agilepy analysis

The AGILE-GRID data analysis can be performed with Agilepy with different techniques:

- using the maximum likelihood estimator analysis
- wavelet techniques
- Lomb-Scargle periodogram analysis (coming soon)

The likelihood analysis reach better sensitivity, more accurate flux measurement, better evaluation of the backgrounds and can work with a detailed source models where more sources can be considered at the same time.

AGILE-GRID light curves can be created in two different ways:

- using the maximum likelihood estimator analysis
- using aperture photometry.

Aperture photometry provides a raw measure of the flux of a sigle source and is less computing demanding.

Maximium Likelihood Estimator caveats

During the fitting process some values are fixed and others are variable, depending on the values of the flags. The execution time strongly depends on the number of the variable parameters. It is not possible to predict how long the fitting process will last or how it depends on the number of parameters, but the dependence is not linear. If all the diffuse coefficients are variable and all spectral parameters are free, for M maps and S sources the number of variable parameters will be 2M+4S. In the case of many maps and many sources, this may lead to a very long execution time.

The fitting process takes place in two steps, according to the method of Maximum Likelihood. During each step all the sources are considered one by one, and several fitting attempts are performed by invoking the function TH1D::Fit() provided by the ROOT library, developed by CERN and will find the related documentation on the CERN web site.

Installation

Agilepy is available as Anaconda package or into a ready-to-use Docker container (from 1.4.0)

Note

AGILE DATASET DOWNLOAD Now it possible to download all the public AGILE dataset stored on SSDC datacenter through a REST Api. Agilepy automatically handles the data and no actions are required from the user. For more information visit this page. This major release includes many new important features and a general refactoring.

Installation with Anaconda

Agilepy (and its dependencies) can be easily installed using Anaconda. You just need to decide the name of the virtual environment that will be created by anaconda.

```
conda config --add channels conda-forge
conda config --add channels plotly
conda create -n <virtualenv_name> -c agilescience agilepy
```

Note

If you want to try agilepy's new features that are not officially released yet, a develpoment environment called agilepy-environment is available into Anaconda cloud. It contains all the dependencies unless agilepy, which must be installed by hand cloning the repository. Check the installation instructions here

Supported platforms:

- linux-64
- osx-64

Note

An experimental package for IBM POWER architecture(ppc64le) is available on Anaconda cloud. Due to some incompability this package does not contain ROOT and AGILE science tools that need to be installed from source. Check the instructions to install AGILE science tools here [https://github.com/AGILESCIENCE/AGILE-GRID-ScienceTools-Setup]

Tested on:

- CentOs 7.6
- Ubuntu 18.04
- Ubuntu 19.10
- Ubuntu 20.04
- macOs 10.14
- macOs 10.15
- macOS 12.0.1

In order to use the software you need to activate the virtual environment first:

```
conda activate <virtualenv_name>
```

or

```
source activate <virtualenv_name>
```

Running jupyter server:

```
start_agilepy_notebooks.sh
```

Installation with Docker

You can pull the image directly from dockerhub using the following command:

```
docker pull agilescience/agilepy:release-<version>
```

Note

Check the installation instructions for Docker here [https://docs.docker.com/get-docker/]

Note

If you want to try agilepy's new features that are not officially released yet, you need to pull a develop image available using **agilepy:develop-latest** tag

Using this command you can launch the container and automatically start jupyter notebook.

```
docker run --rm -it -p 8888:8888 \
-e DISPLAY=$DISPLAY \
-v /tmp/.X11-unix:/tmp/.X11-unix:rw \
-v $PWD/shared_dir:/shared_dir \
agilescience/agilepy:release-<version> /bin/bash -c \
"source /opt/anaconda3/etc/profile.d/conda.sh && conda activate
jupyter notebook --ip='*' --port=8888 --no-browser --allow-root
```

shared_dir must be created before launching the command, it is not necessary, but useful for several cases (exporting analysis outside the container, link another dataset etc.)

Jupyter server is at localhost:8888

Agilepy's containers can be found at dockerhub page [https://hub.docker.com/repository/docker/agilescience/agilepy]

Supported platforms:

- linux-64
- osx-64
- win-64(see note)

Tested on:

- CentOs 7.6
- Ubuntu 18.04
- Ubuntu 19.10
- Ubuntu 20.04
- macOs 10.14
- macOs 10.15
- Windows 10 v2004 (May 2020 Update)

Note

It's possible to run Agilepy's container in Windows10(still not supported by Anaconda installation), in order to do that, you need to install WSL2 and docker first.

Check the installation instructions for WSL2 here [https://docs.microsoft.com/en-us/windows/wsl/install-win10] and docker here [https://docs.docker.com/docker-for-windows/wsl/]

Manual Installation

If the isntallation does not work with the instructions above, it is recommended to install agilepy and its dependencies from scratch. The dependencies required by Agilepy are:

Root 6.26 Cfitsio 4.1 Zlib

AGILE's Science Tools [https://github.com/AGILESCIENCE/AGILE-GRID-ScienceTools-Setup/tree/master] (the correct tag to install is on sciencetools_version.txt in the repository main directory)

Agilepy python dependencies [https://github.com/AGILESCIENCE/Agilepy-recipe/blob/master/recipes/docker/base/requirements.txt]

Uninstalling

Anaconda

```
conda env remove --name <virtualenv_name>
```

Docker

```
docker rmi agilescience/agilepy:release-<version>
```

Package distribution structure

The virtual environment <virtualenv_name> folder is under the “envs” folder within the root folder of your anaconda installation.

It contains all the dependencies Agilepy requires. Here, there is the “agiletools” directory, containing AGILE’s scientific software.

Quickstart guide

To import the library:

```
from agilepy.api import AGAnalysis
```

You can create the (required) yaml configuration file, calling the following static method:

```
AGAnalysis.getConfiguration(
      "./agconfig.yaml", # the destination path of the configura
      "username", # the name of the flare advocate
      "OJ287", # the name of the source
      58930, # tmin
      58936, # tmax
      "MJD", # time type
      206.8121188769472, # glon
      35.8208923457401, # glat
      "$HOME/agilepy_analysis", # the destination path of the ou
      1, # the verbosity level
      evtfile="/AGILE_PROC3/FM3.119_ASDC2/INDEX/EVT.index", # op
      logfile="/AGILE_PROC3/DATA_ASDC2/INDEX/LOG.log.index" # op
)
```

In order to interact with the library you need to obtain an instance of the AGAnalysis class:

```
ag = AGAnalysis('agconfig.yaml')
```

Then you have to load the models of the sources (you can filter them by their distance (degree) from l,b provided within the configuration file):

```
sources = ag.loadSourcesFromCatalog('2AGL', rangeDist=(0, 10))
```

Keyword arguments can be passed via setOptions() to override configuration parameters:

```
ag.setOptions(binsize=0.50, outdir="./output")
```

To generate sky maps:

```
maplistfile = ag.generateMaps()
```

To display and interact with the sky maps:

```
ag.displayCtsSkyMaps(smooth=True, sigma=3)
ag.displayExpSkyMaps()
ag.displayGasSkyMaps()
```

To perform an maximum likelyhood estimation analysis:

```
sourcefiles = ag.mle()
```

You can query the sources with an arbitrary boolean expression string..

```
selectedSources = ag.selectSources("flux > 0 AND dist <= 1 OR sc
```

..and fix or free a source's parameter:

```
sourcefiles = ag.freeSources('name == "CYGX3"', "flux", True)
```

You can generate a light curve data file with…

```
lightCurveData = ag.lightCurveMLE("CYGX3", tmin=58930 , tmax=589
```

…and display the interactive light curve plot with:

```
ag.displayLightCurve("mle")
```

If you want to manually update the value of a source's spectrum parameter, you can do it with:

```
sources = ag.selectSources('name == "2AGLJ2021+4029"')
source = sources.pop()
source.spectrum.set("index", 1.8)
```

Hint

Check out the API documentation and the Jupiter notebooks section!

Jupyter Notebooks

Several Jupyter notebooks are available. You can start a Jupyter server calling:

```
start_agilepy_notebooks.sh
```

Tutorial notebooks

There're several categories of tutorial notebooks:

- science_api_tutorial: the most important ones. They show the basic usage of Agilepy to perform a scientific analysis for sources detection. The following notebooks are useful example of the use of Agilepy and they can be runned using provided sample data:
  - VELA: analysis of Vela region
  - 3C454.3: analysis of November's 2010 gamma-ray flare of AGN 3C454.3.
  - AITOFF: how to produce a full sky AITOFF projection image.
- Wavelet analysis: it shows how to use the Agilepy's wavelet analysis API.
- engineering_api_tutorial: they show how to use the Agilepy's engineering analysis API.

The following notebook is another useful example of the use of Agilepy that is runned downloading AGILE data from SSDC website:

- PKS1510-089: analysis of 2009 gamma-ray flare

Analysis notebooks

These notebooks have been developed for internal purposes of the AGILE Team. A template notebook is also provided to speed up the development of a new analysis notebook for AGILE Flare Advocate team.

Download AGILE-GRID data

The AGILE-GRID data can be downloaded in two ways: automated by Agilepy or manually.

Automated download using SSDC REST Api

The AGILE-GRID data is download automatically by agilepy.

Version 1.5.0 has implemented the SSDC REST Api in order to get the AGILE dataset from SSDC datacenter. All the required data is downloaded according to tmin and tmax values selected in configuration file. This feature works when generatesmaps method is called.

Pre-requisites:

- Internet connection (> 200 Mb/s)

SSDC Data policy:

- EVT files contain 15 days of data (2 files per month)
- LOG files contain 1 day of data

Eg for getting data from 10/10/2018 to 05/11/2018 it returns:

- 3 EVT files (30/09-15/10, 15/10-31/10, 31/10-15/11)
- 26 LOG files, one file for each day

Two query files are created to keep track of the query history and to implement the policy above. Before calling Rest api, Agilepy checks if the dates selected are in query files, if True download is not performed. If False Agilepy downloads the data in /tmp/ folder, it unpacks them into the selected datapath and it automatically calls indexgen tool for generating index files. Finally, it updates query files.

Example

```
AGAnalysis.getConfiguration(
        confFilePath = confFilePath,
        evtfile=None,
```

```
        logfile=None,
        userName = "username",
        sourceName = "PKS1510-089",
        tmin = 54891,
        tmax = 54921,
        timetype = "MJD",
        glon = 351.29,
        glat = 40.13,
        outputDir = "$HOME/agilepy_analysis",
        verboselvl = 0,
        userestapi=True,
        datapath="$HOME/agile_dataset"
)
```

Advanced Information

Query files

Agilepy uses text files called “qfiles”. These files contain the slots requested by the user, according to SSDC policies. With query file it is possible to avoid multiple downloads for the same dates (useful for slow connections and if there are no data in the selected range days).

Index files

Index file is created by Indexgen tool immediately after the download. No action is required from the user.

Plotting index files vs query files

A function to plot the dates from index and query file can be useful to check differences between asked data and real data. Sometimes could happen that data are not available for several reasons (instruments off etc), data will not be downloaded but agilepy writes in query files in order to not perform a second request.

AGILE Data Coverage

SSDC uploads AGILE dataset once per month, this means that it could not be possible to select a date close to the present day. In this particular case, query files must not be uploaded, because in the future data will be available. Agilepy gets AGILE data coverage from SSDC and writes it into a file called Agilepy_coverage, when AGDataset starts it checks if last coverage is more than 60 days old from the present date and it updates it if positive.

Manual download

The AGILE-GRID data obtained both in pointing and in spinning mode are publicly available and can be download manually from the ASI/SSDC https://www.asdc.asi.it/mmia/index.php?mission=agilemmia

Prepare index files

There're two types of data files: events list (EVT) and log data (LOG). They both are compressed fits files. Each file refers to a specific time interval.

Example:

```
agql1511240600_1511240730.LOG.gz
agql1511240730_1511240900.EVT__FM.gz
```

In order to use Agilepy (or the Agile science tools) a special file, called "index", is needed. This file is used by Agilepy to know the position of the data files and which file refers to which interval. Two index files are needed: one for the event data and one for the log data.

Those index files have four column:

- file name
- time start of the file in Terrestrial Time (TT)
- time end of the file in Terrestial time (TT)
- LOG or EVT marker to identify the fole types

Here some examples of LOG and EVT indexes:

```
head -n 3 /ASDC_PROC3/DATA_ASDCe/INDEX/LOG.log.index
/AGILE_PROC3/DATA_ASDCe/LOG/ag-107092735_STD0P_GO.LOG.gz 1070927
/AGILE_PROC3/DATA_ASDCe/LOG/ag-107179135_STD0P_GO.LOG.gz 1071791
/AGILE_PROC3/DATA_ASDCe/LOG/ag-107265535_STD0P_GO.LOG.gz 1072655


head -n 3 /ASDC_PROC3/FM3.119_ASDCSTDk/INDEX/EVT.index
/ASDC_PROC3/FM3.119_ASDCSTDk/EVT/ag0910311200_0911301200_STD1Kal
/ASDC_PROC3/FM3.119_ASDCSTDk/EVT/ag0911301200_0912201200_STD1Kal
/ASDC_PROC3/FM3.119_ASDCSTDk/EVT/ag0912201200_1001151200_STD1Kal
```

You can use the AG_indexgen tool to generate the .index file:

```
AG_indexgen <path to data> <type> <output file>
```

Where <type> can be EVT or LOG.

Example:

```
AG_indexgen /AGILE_PROC3/FM3.119_ASDC2/EVT EVT /home/user/data.i
```

Agilepy test data

The Agilepy conda package gets shipped with two subsets of the AGILE data archive for the purpose of unit testing and to show how to run scientific analysis with the tutorial notebooks.

test_dataset_6.0

A test data to analyse Vela region. The provided period is MJD 58026.50-58031.50.

The index files are the following:

```
evtfile="$AGILE/agilepy-test-data/test_dataset_6.0/EVT/EVT.index
logfile="$AGILE/agilepy-test-data/test_dataset_6.0/LOG/LOG.index
```

test_dataset_agn

A test data to analyse the November's 2010 flare of 3C454.3 source. The provided period is MJD 55513.00-55520.00.

The index files are the following:

```
evtfile="$AGILE/agilepy-test-data/test_dataset_agn/EVT/EVT.index
logfile="$AGILE/agilepy-test-data/test_dataset_agn/LOG/LOG.index
```

Configuration file

General

A yaml [https://docs.ansible.com/ansible/latest/reference_appendices/YAMLSyntax.html] configuration file is required in order to run Agilepy.

It is composed by several sections and each section holds several configuration options, some of which are optional (having a default value), some others are required.

It supports environment variables (they can be used to define file system paths).

It can be created easily, calling the following static method and passing the minimal set of (required) configuration parameters.

```
AGAnalysis.getConfiguration(
      confilepath="./agconfig.yaml", # the destination path of the configura
      userName="username", # the name of the flare advocate
      sourceName="OJ287", # the name of the source
      tmin=58930, # tmin
      tmax=58936, # tmax
      timetype="MJD", # time type
      glon=206.8121188769472, # glon
      glat=35.8208923457401, # glat
      outputDir="$HOME/agilepy_analysis", # the destination path of the outp
      verboselvl=1, # the verbosity level
      evtfile="evt indexfile", # optional parameter
      logfile="log indexfile", # optional parameter
      datapath="datapath",
      userestapi=True,

)
```

The method above will create the following configuration file:

```
input:
  evtfile: None
  logfile: None
  userestapi: True
  datapath: datapath
output:
  outdir: $HOME/agilepy_analysis
  filenameprefix: analysis_product
  logfilenameprefix: analysis_log
  sourcename: sourcename
```

```
  username: username
  verboselvl: 2

selection:
  emin: 100
  emax: 10000
  tmin: 54935.0
  tmax: 54936.0
  timetype: MJD
  glon: 355.447
  glat: -0.2689
  proj: ARC
  timelist: None
  filtercode: 5
  fovradmin: 0
  fovradmax: 60
  albedorad: 80
  dq: 0
  phasecode: null
  lonpole: 180
  lpointing: null
  bpointing: null
  maplistgen: "None"

maps:
  mapsize: 40
  useEDPmatrixforEXP: false
  expstep: null
  spectralindex: 2.1
  timestep: 160
  projtype: WCS
  proj: ARC
  binsize: 0.25
  energybins:
    - 100, 10000
  fovbinnumber: 1
  offaxisangle: 30

model:
  modelfile: null
  galmode: 1
  isomode: 1
  galcoeff: null
  isocoeff: null
  emin_sources: 100
  emax_sources: 10000
  galmode2: 0
  galmode2fit: 0
  isomode2: 0
  isomode2fit: 0
```

```
mle:
  ranal: 10
  ulcl: 2
  loccl: 95
  expratioevaluation: true
  expratio_minthr: 0
  expratio_maxthr: 15
  expratio_size: 10
  minimizertype: Minuit
  minimizeralg: Migrad
  minimizerdefstrategy: 2
  mindefaulttolerance: 0.01
  integratortype: 1
  contourpoints: 40
  edpcorrection: 0.75
  fluxcorrection: 0

ap:
  radius: 3
  timeslot: 3600

plotting:
  twocolumns: False
```

Updating the configuration options

The user should not directly manipulate the configuration file, because the configuration file is read only once, when the AGBaseAnalysis constructor is called. Hence, the configuration file modification will not affect the internal configuration object. Also, updating the values held by this object will not affect the original values written on disk.

In order to update the internal configuration object, the user can rely on the following methods:

- getOption(optionName)
- setOption(**kwargs)

For example:

```
ag.setOptions(binsize=0.50, energybins=[[100, 300], [500, 1000]])
print(ag.getOption("energybins"))
```

Configuration options

This section describes the configuration options.

Section: *'input'*

This section defines the input data files. The input data files are indexes: each row holds the file system position of an actual event data/log file, together with the time interval it refers to. If userestapi if True the selection of evtfile and logfile is not required, Agilepy creates its own index files automatically. See more details in this link.

Option	Description	Type	Required	Default
evtfile	Path to index evt file name	str	no	None
logfile	Path to index log file name	str	no	None
userestapi	If true downloads date into datapath	bool	no	True
datapath	the position of AGILE data	str	no	None

Section: *'output'*

The output section collects options related to the output files generation and logging.

The *'outdir'* option sets the root directory of the analysis results where all output files are written.

Agilepy use two loggers, one logs messages on the console, the other writes messages on disk. The *'verboselvl'* option sets the verbosity of the Agilepy console logger. The Agilepy file logger verbosity is set to 2 by default. There are 4 kind of messages based on their importance factor:

- CRITICAL: a message describing a critical problem, something unexpected, preceding a program crash or an Exception raise.
- WARNING: an indication that something unexpected happened, or indicative of some problem in the near future (e.g. 'disk space low'). The software is still working as expected.
- INFO: confirmation that things are working as expected.
- DEBUG: detailed information, typically of interest only when diagnosing problems.

Option	Description	Type	Required	Default

Option	Description	Type	Required	Default
outdir	Path of the output directory	str	yes	null
filenameprefix	The filename prefix of each output file	str	yes	null
logfilenameprefix	The filename prefix of the log file	str	yes	null
sourcename	The name of the source under analysis	str	yes	null
userName	The name of the user performing the analysis	str	yes	null
verboselvl	0 ⇒ *CRITICAL* and *WARNING* messages are logged on the console. 1 ⇒ *CRITICAL*, *WARNING* and *INFO* messages are logged on the console. 2 ⇒ *CRITICAL*, *WARNING*, *INFO* and *DEBUG* messages are logged on the console	int	no	1

Section: *'selection'*

The temporal, spatial and spectral binning of the data can be customized using the configuration options of this section.

The center of the *ROI* (region of interest) is defined by explicit Galactic sky coordinates (glon and glat).

Option	Description	Type	Default	Required

Option	Description	Type	Default	Required
emin	Energy min in MeV	int	100	no
emax	Energy max in MeV	int	10000	no
glat	Center of the ROI (’*latitude*’ or *‘b’*)	float	null	yes
glon	Center of the ROI (’*longitude*’ or *‘l’*)	float	null	yes
tmin	Minimum time (in MJD or TT)	float	null	yes
tmax	Maximum time (in MJD or TT)	float	null	yes
timetype	The date format of tmin and tmax. Possibile values: [*‘MJD’, ‘TT’*]	str	null	yes
timelist	it’s a path to a file containing a list of time intervals in TT format to generate maps integrated within a time window. If specified, *‘tmin’* and *‘tmax’* are ignored.	str	null	no
filtercode	filtercode = 5 select G filtercode = 0 select G+L+S	int	5	no
fovradmin	fovradmin < fovradmax	int	0	no
fovradmax	fovradmax > fovradmin (dq = 0 is necessary for setting)	int	60	no

Option	Description	Type	Default	Required
albedorad	albedo selection cut (dq = 0 is necessary for setting)	int	80	no
dq	Data quality selection filter. A combination of fovradmax and albedorad. Possible values are [0,1,2,3,4,5,6,7,8,9] dq = 0 -> albedorad and fovradmax are free and they must always be specified in setOption dq = 1 -> albedorad=80, fovradmax=60 dq = 2 -> albedorad=80, fovradmax=50 dq = 3 -> albedorad=90, fovradmax=60 dq = 4 -> albedorad=90, fovradmax=50 dq = 5 -> albedorad=100, fovradmax=50 dq = 6 -> albedorad=90, fovradmax=40 dq = 7 -> albedorad=100, fovradmax=40 dq = 8 -> albedorad=90, fovradmax=30 dq = 9 -> albedorad=100, fovradmax=30	int	0	no
phasecode	Photon list selection parameter based on the orbital phase. If ‘None’, the automated selection is done following the *‘phasecode’* rule	int	null	no

Phasecode rule

- phasecode = 2 -> spinning mode, SAA excluded with AC counts method.
- phasecode = 6 -> spinning mode, SAA excluded according to the magnetic field intensity (old definition of SAA, defined by TPZ)
- phasecode = 18 -> pointing mode, SAA and recovery exluded.

It is suggested to use phasecode = 2 for data taken in spinning mode.

```
def setPhaseCode(tmax)
   if @phasecode == -1
      if tmax.to_f >= 182692800.0
         @phasecode = 6 #SPIN
      else
```

```
          @phasecode = 18 #POINTING
       end
    end
end
```

filtercode rule

A set of different on-board triggers enables the discrimination of background events (mainly cosmic rays in the AGILE Low Earth Orbit) from gamma-ray events. The data processing of the GRID events use an additional on-ground filters and provides a classification of each event:

- P : events classified as a charged particle and rejected
- G : events classified as gamma-ray photons. This is the most useful class for the analysis
- S : events classified as single-track: this is a special class of events with no separation between the electron and positron tracks
- L : limbo events, not clearly classified.

The events provided in the EVT files are of type G, S, and L. The AGILE team recommends to use the G class for scientific analysis. Only for gamma-ray bursts or other short transient events, and for pulsar timing analysis the G, S and L classes should be used together.

Section: *'maps'*

These options control the behaviour of the sky maps generation tools. The *'energybin'* and *'fovbinnumber'* options set the number of maps that are generated:

```
number of maps = number of energy bins * fovbinnumber
```

The *'energybin'* option is a list of strings with the following format:

```
energybins:
  - 100, 1000
  - 1000, 3000
```

The *'fovbinnumber'* option sets the number of bins between *'fovradmin'* and *'fovradmax'* as:

```
number of fov bins = (fovradmax-fovradmin)/fovbinnumber
```

Note

One map is generated for each possible combination between the *'energybin'* (emin, emax) and the *'fovbinnumber'* (fovmin, fovmax). The order of map generation is

described by the following pseudocode:

```
For each fovmin..fovmax:
    For each emin..emax:
        generateMap(fovmin, fovmax, emin, emax)
```

Option	Description	Type	Default	Required
mapsize	Width of the ROI in degrees	float	40	no
useEDPmatrixforEXP	Use the EDP matrix to generate the exposure map.	boolean	False	no
expstep	Step size of the exposure map, if ‘None’ it depends by round(1 / binsize, 2) (e.g. 0.3->3, 0.25->4, 0.1->10)	int	None	no
spectralindex	Spectral index of the exposure map	float	2.1	no
timestep	LOG file step size of exposure map (LOG file are at 0.1s)	float	160	no
projtype	Projection mode. Possible values: [’*WCS*’]	str	WCS	no
proj	Spatial projection for WCS mode. Possible values: [’*ARC*’, ‘*AIT*’]	str	ARC	no

Option	Description	Type	Default	Required
skytype	gasmap: 0) SKY000-1 + SKY000-5, 1) gc_allsky maps + SKY000-5, 2) SKY000-5 3) SKY001 (old galcenter, binsize 0.1, full sky), 4) SKY002 (new galcenter, binsize 0.1, full sky)	int	4	no
binsize	Spatial bin size in degrees	float	0.25	no
energybin	The enegy bins of analysis. A list of value. | To configure: | 1) directly in the yaml configuration file; | 2) Use the method e.g. ag.setOptions(energybins= [[100, 300], [500, 1000]]) | 3) Use the method ag.setOptionEnergybin(value)	List<String>	[100, 10000]	no
fovbinnumber	Number of bins between fovradmin and fovradmax. Dim = (fovradmax-fovradmin)/fovbinnumber	int	1	no

Section: *'model'*

The '*galcoeff*' and '*isocoeff*' options values can take the default value of null or they can be a a list of values separated by a comma. If they are set to null it means they are free to change.

```
model:
  galcoeff: 0.8, 0.6, 0.5, 0.4
  isocoeff: 8, 10, 12, 14
```

In this case, you should pay attention on how the sky maps are generated: the following example show which iso/gal coefficients are assigned to which map.

```
selection:
  fovradmin: 0
  fovradmax: 60

maps:
  energybins:
    - 100, 300
    - 300, 1000
  fovbinnumber: 2

model:
  galcoeff: 0.8, 0.6, 0.5, 0.4
  isocoeff: 8, 10, 12, 14
```

FOV bins:
(0, 30), (30, 60)

Map #1 has: fovmax:0 fovmax:30 emin:100 emax:300 galcoeff:0.8 isocoeff:8
Map #2 has: fovmax:0 fovmax:30 emin:300 emax:1000 galcoeff:0.6 isocoeff:10
Map #3 has: fovmax:30 fovmax:60 emin:100 emax:300 galcoeff:0.5 isocoeff:12
Map #4 has: fovmax:30 fovmax:60 emin:300 emax:1000 galcoeff:0.4 isocoeff:14

Option	Description	Type	Default	Required
modelfile	A file name that contains point sources, diffuse and isotropic components	string	null	yes
galmode	int	1	no	
isomode	int	1	no	
galcoeff	set into .maplist if >= 0	null, float or str	null	no

Option	Description	Type	Default	Required
isocoeff	set into .maplist if >= 0	null, float or str	null	no
emin_sources	energy min of the modelfile	int	100	no
emax_sources	energy max of the modelfile	int	10000	no

galcoeff and isocoeff

galcoeff and isocoeff are the coefficients for the Galactic and isotropic diffuse emission components respectively. The values may be fixed during the fitting process or some or all of them may be optimized by allowing them to vary. Agilepy allows to evaluate these coefficient and fix them or to keep these coefficient free.

Positive values are considered fixed, while negative values are free to vary starting from their absolute values. These coefficients are affected by the galmode and isomode coefficients described in the following section.

galmode and isomode

'galmode' and *'isomode'* are integer values describing how the corresponding coefficients *'galcoeff'* or *'isocoeff'* found in all the lines of the maplist will be used:

0: all the coefficients are fixed.
1: all the coefficients are fixed if positive, variable if negative (the absolute value is the initial value). This is the default behaviour.
2: all the coefficients are variable, regardless of their sign.
3: all the coefficients are proportionally variable, that is the relative weight of their absolute value is kept.

Section: *'mle'*

The maximum likelihood estimation analysis is configured by the following options:

Option	Description	Type	Default	Required
ranal	Radius of analysis	float	10	No
ulcl	Upper limit confidence level, expressed as sqrt(TS)	float	2	No
loccl	Source location contour confidence level (default 95 (%)confidence level). Possible values: [*99*, *95*, *98*, *50*]	int	95	No
fluxcorrection	Correction of the flux taking into account the spectral model. Possible values: [0 (no correction), 1 (enable correction)].	int	0	No

Exp-ratio evaluation options

See details in this link.

Option	Type	Default	Required	Description
expratioevaluation	bool	yes	none	
expratio_minthr	float	0	none	
expratio_maxthr	float	15	none	
expratio_size	float	10	none	

Section: *'ap'*

This section describes the configuration parameters for the Aperture Photometry analysis.

Option	Description	Type	Required	Default
radius	The radius of analysis	float	no	3
timeslot	The size of the temporal bin	int	no	3600

Section: *'plot'*

This section defines the plotting configuration.

Option	Description	Type	Required	Default
twocolumns	The plot is adjusted to the size of a two column journal publication	boolean	False	no

Sources file

The sources can be defined using one of two different formats: xml document and text file.

The flux parameter estimates are relevant in the fitting process, as the sources are considered one by one, starting with the one with the brightest initial flux value, regardless of the order they are given in the source file.

Spectral models

A full energy band spectral fit of the data is performed with different spectral model. The spectral representations used in the BUILD25 are PL, exponential cut-off PL, super-exponential cut-off PL, and log parabola (LP). More details are reported in https://arxiv.org/abs/1903.06957

The PL spectral model is used for all sources that are not significantly curved and have low exposure,

$$\frac{dN}{dE} = N_0 E^{-\alpha}$$

where N0 is the prefactor and alpha is the index explicitly evaluated by the MLE method. Our MLE spectral fitting does not explicitly output the prefactor value, which is internally calculated by the numerical procedure. The majority of the AGILE sources are described by a PL.

The exponential cut-off PL spectral model (PC) is

$$\frac{dN}{dE} = N_0 E^{-\alpha} \exp\left(-\frac{E}{E_c}\right)$$

where N0 is the prefactor, α is the index, and Ec is the cut-off energy. The values Ec and α are explicitly provided by the MLE method.

The super exponential cut-off PL spectral model (PS) is

$$\frac{dN}{dE} = N_0 E^{-\alpha} \exp\left(-\left(\frac{E}{E_c}\right)^{\beta}\right)$$

where N0 is the prefactor, α is the first index, β the second index, and Ec is the cut-off energy. The parameters α, Ec, and β are explicitly provided by the MLE method.

The LP spectral model is

$$\frac{dN}{dE} = N_0 E^{-\alpha-\beta\ \ln(E/E_c)}$$

where N0 is the prefactor, Ec is the pivot energy, α is the first index, β the curvature. The parameters α, Ec, and β are explicitly provided by the MLE method.

The selection of curved spectra followed the acceptance criteria described in bulgarelli19. Briefly, a source is considered significantly curved if T Scurved > 16, where T Scurved = 2 × (log L(curved spectrum)−log L(power law), where L is the likelihood function obtained changing only the spectral representation of that source and refitting all free parameters.

Source library format (xml document)

```
<?xml version="1.0" ?>
<source_library title="source library">

  <!-- Point Sources -->
  <source name="2AGLJ2202+4214" type="PointSource">
    <spectrum type="PowerLaw">
      <parameter name="flux" free="1"  value="7.45398e-08"/>
      <parameter name="index" free="1" scale="-1.0" value="1.969
    </spectrum>
    <spatialModel type="PointSource" location_limit="0">
      <parameter name="pos" value="(92.4102, -10.3946)" free="0"
    </spatialModel>
  </source>

  <source name="2AGLJ0007+7308" type="PointSource">
    <spectrum type="PLExpCutoff">
       <parameter name="flux" free="1"  value="41.6072e-08"/>
       <parameter name="index" free="1" scale="-1.0" value="1.29
       <parameter name="cutoffEnergy" free="1" scale="-1.0" valu
    </spectrum>
    <spatialModel type="PointSource" location_limit="0">
       <parameter name="pos" value="(119.677, 10.544)" free="0"
    </spatialModel>
  </source>

  <source name="2AGLJ0835-4514" type="PointSource">
    <spectrum type="PLSuperExpCutoff">
      <parameter name="flux" free="1"  value="969.539e-08"/>
      <parameter name="index1" free="1" scale="-1.0" value="1.71
      <parameter name="cutoffEnergy" free="1" value="3913.06" mi
      <parameter name="index2" free="1" value="1.3477" min="0"
    </spectrum>
    <spatialModel type="PointSource" location_limit="0">
      <parameter name="pos" value="(263.585, -2.84083)" free="0"
    </spatialModel>
  </source>

  <source name="2AGLJ1801-2334" type="PointSource">
    <spectrum type="LogParabola">
      <parameter name="flux" free="1"  value="35.79e-08"/>
```

```
      <parameter name="index" free="1" scale="-1.0" value="3.379
      <parameter name="pivotEnergy" free="1" scale="-1.0" value=
      <parameter name="curvature" free="1" scale="-1.0" value="0
    </spectrum>
    <spatialModel type="PointSource" location_limit="0">
      <parameter name="pos" value="(6.16978, -0.0676943)" free="
    </spatialModel>
  </source>

</source_library>
```

Working with sources

The Source abstraction

The main abstraction of Agilepy is the Source class. It is described by several parameters, some of which can be free to vary, and they are changed by the mle() analysis.

The set of the parameters describing the source can vary, depending on the spectrum and spatial model types of the source.

The different types of sources are described here.

How to load or add new sources

In order to perform a scientific analysis with Agilpy, at least one Source model must be loaded. There are several ways to do that.

The loadSourcesFromCatalog(catalogName, rangeDist=0, inf, show=False) allows to load a source catalog, while filtering the sources by their distance (degree) from the l,b position provided within the configuration file.

```
sources = ag.loadSourcesFromCatalog('2AGL', rangeDist=(0, 10))
```

The loadSourcesFromFile(sourcesFilepath, rangeDist=0, inf, show=False) loads the sources, reading their model from a file.

The addSource(sourceName, sourceDict) method allows the user to define on the fly a source model with a python dictionary. Check the tutorial notebooks for an example.

How to select Source objects

The sources can be selected via the the selectSources(selection, show=False) method. The “selection” argument supports either lambda functions and boolean expression strings. The user can call selectSources (with show=True) to show the source description

```
source = ag.selectSources('name == "2AGLJ2254+1609"', show=Fals
print(source)
```

```
------------------------------------------------------------
Source name: 2AGLJ2254+1609 (PointSource)
 * Free parameters: flux
 * Initial source parameters: (PowerLaw)
    - flux(ph/cm2s): 7.50937e-07
    - index: 2.20942
    - Source position: (86.1236, -38.1824) (l,b)
    - Distance from map center: 0.011 deg
------------------------------------------------------------
```

Other examples:

```
sources = ag.selectSources('name == "PKS1510-089"', show=False)
```

```
sources = ag.selectSources('flux > 0', show=False)
```

```
sources = ag.selectSources(lambda name, sqrtTS: name == "2AGLJ20
```

How to let the source’s parameters to vary

In order to free or fix a sources’ parameter, the user can rely on the freeSources(selection, parameterName, free, show=False) method. The “selection” argument is used like in *selectSources*, so you can free a parameter of multiple sources at once.

```
aganalysis.freeSources(lambda name, dist, flux : Name == "2AGLJ2
```

```
ag.freeSources('name == "2AGLJ1513-0905"', "index", True, show=T
```

Check the api documentation or the tutorial notebooks for additional examples.

How to check which source's parameters are free to vary

The user can obtain this information by printing the Source object or calling the getFreeParams() method of the Source object.

```
print(source.getFreeParams())
['flux']
```

The "multi" description of a Source object

If the user performs an mle analysis, the Source object will contain also the analysis results.

```
print(source)
-------------------------------------------------------------
Source name: 2AGLJ2254+1609 (PointSource) => sqrt(ts): 10.2226
 * Free parameters: flux index
 * Initial source parameters: (PowerLaw)
 - flux(ph/cm2s): 7.50937e-07
 - index: 2.20942
 - Source position: (86.1236, -38.1824) (l,b)
 - Distance from map center: 0.011 deg
 * Last MLE analysis:
 - flux(ph/cm2s): 8.68363e-06 +/- 1.62474e-06
 - index: 2.51001 +/- 0.173795
 - upper limit(ph/cm2s): 1.13967e-05
 - ergLog(erg/cm2s): 1.39217e-09 +/- 2.6048e-10
 - galCoeff: [0.7, 0.7, 0.7, 0.7, 0.7]
 - isoCoeff: [5.24757, 3.14662, 0.953512, 1.59944e-10, 0.557554
 - exposure(cm2s): 13672200.0
 - exp-ratio: 0.0
 - L_peak: 86.1236
 - B_peak: -38.1824
 - Distance from start pos: 0.0
 - position:
     - L: -1.0
     - B: -1.0
     - Distance from start pos: -1.0
     - radius of circle: -1.0
```

```
      - ellipse:
      - a: -1.0
      - b: -1.0
      - phi: -1.0
-------------------------------------------------------------
```

The values L_peak and B_peak set to the initial values in the source location is fixed. If it is allowed to vary then they are set to the position for which the TS is maximized. If a confidence contour was found, the parameters of the “ellipse” section describe the best-fit ellipse of the contour, described in detail below. The counts and fluxes are provided, as well as their symmetric, positive, and negative errors if the flux is allowed to vary. For convenience, the exposure of the source, used to calculate the source counts from the flux, is also provided. Finally, the spectral index and its error, or the other spectral parameters, if applicable, are provided.

How to manually inspect source’s attributes

The user can rely on a getter method get(sourceAttribute) method.

```
print(source.get("cutoffEnergy"))
print(source.get("index"))
print(source.get("pos"))
print(source.get("dist"))
print(source.get("locationLimit"))
print(source.get("multiFlux"))
```

How to manually change a source’s attributes

The user can rely on a setter method set(sourceAttribute) method.

```
source.set("index2", 1.34774)
```

The setAttributes() method allows to change the following attributes: value, free, scale, min, max, locationLimit. Example:

```
source.spectrum.cutoffEnergy.setAttributes(min=3000, max=5000)
```

In order to change the position of a source, the user can rely on the updateSourcePosition(sourceName, glon, glat) method.

Products

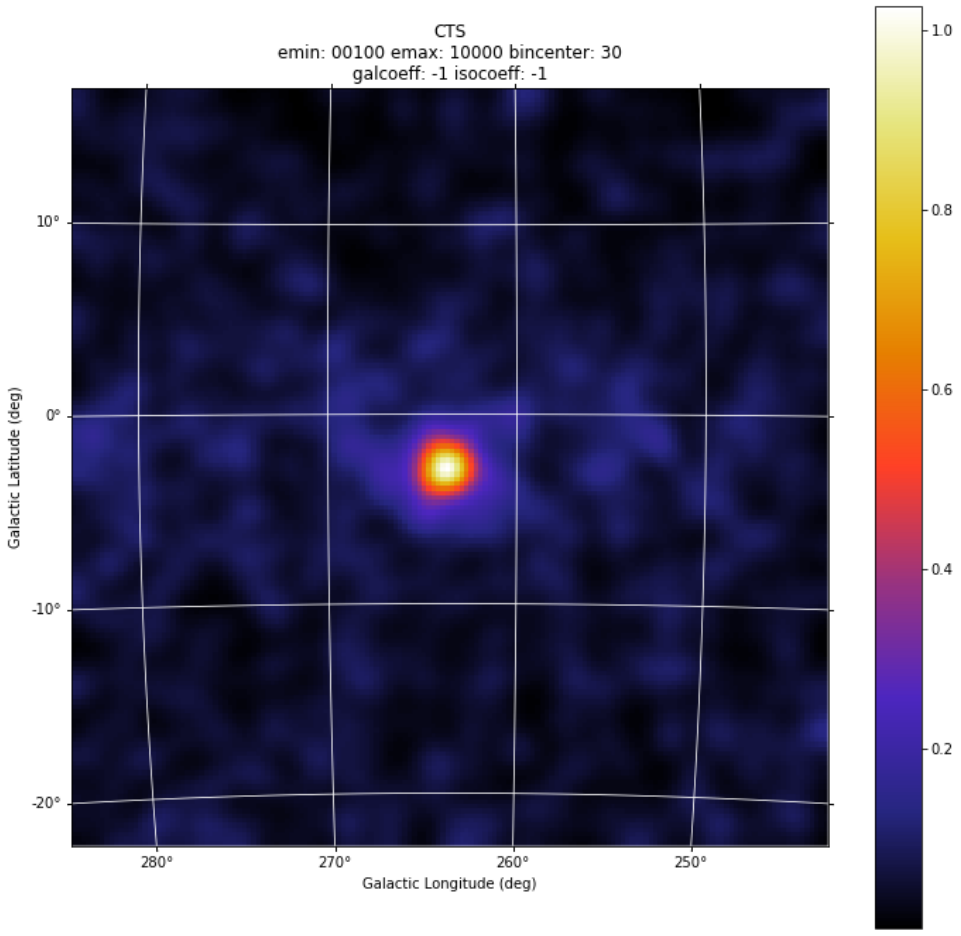
Sky maps

‘*.cts.gz*’ file

Counts maps are generated by the procedure AG_ctsmapgen embedded into Agilpy.

AG_ctsmapgen reads the event files listed in the event file index (see “AGILE data” section), bins the counts between tmin and tmax, and outputs a FITS image file. The image is a two-dimensional array in the ARC or AIT projection. The projection, size and resolution, the center and rotation of the map in Galactic coordinates, tmin, tmax, emin, and emax, along with various integration parameters (fovrad, fovradmin, albrad, phasecode) are managed by Agilepy.

The parameters are described in the “Configuration file” section.



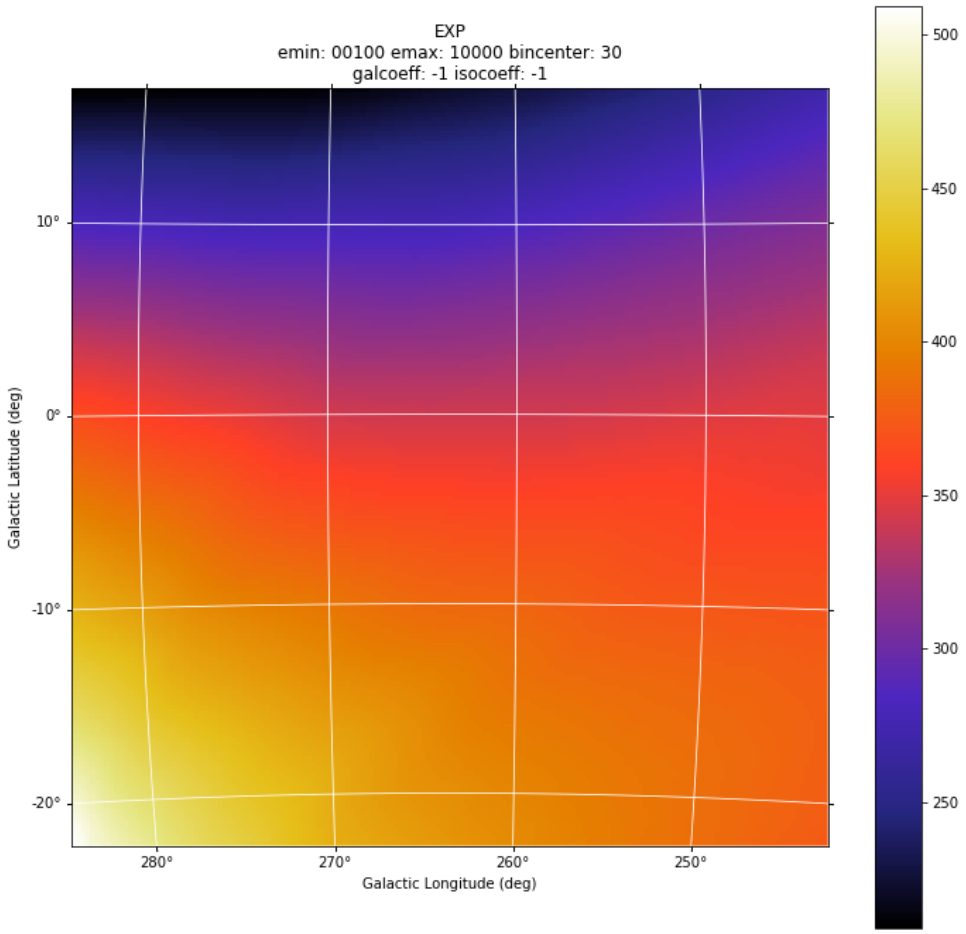
‘*.exp.gz*’ file

Exposure maps are generated by the procedure AG_expmapgen embedded into Agilpy.

The task AG_expmapgen reads the log files listed in the LOG index (see “AGILE data” section), integrates the exposure between tmin and tmax, and outputs a FITS exposure image file. The image is a two-dimensional array in either the ARC or AIT projection. The projection, size and resolution,

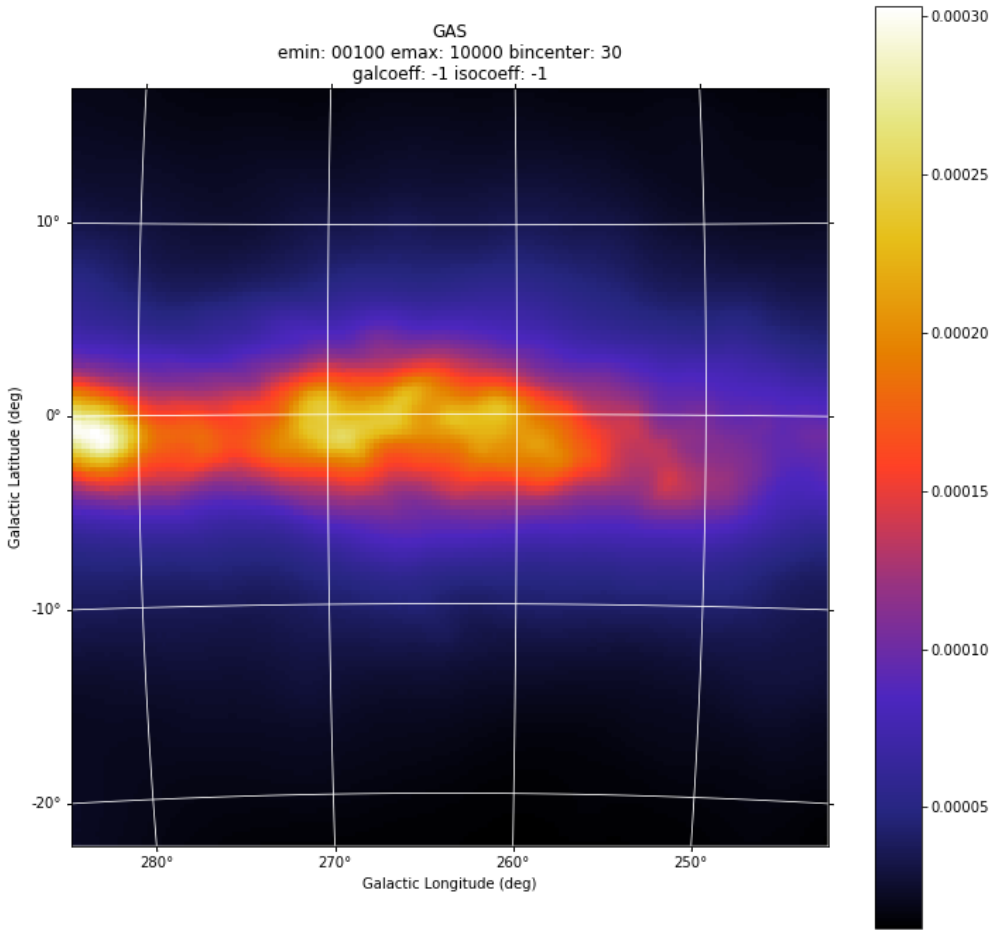
and center and rotation of the map in Galactic coordinates, tmin, tmax, emin, emax, and index file are managed by Agilepy, along with various integration parameters (fovrad, albrad, y tol, roll tol, earth tol, phasecode), and an interpolation step size (binstep). The interpolation procedure is a linear interpolation method in which only one bin each N is calculated (where N is the step size parameter). For a bin size of 0.5 deg or 0.25 deg with a step size of N = 4 it is possible to get a good approximation of the exposure map.

The parameters are described in the “Configuration file” section.



‘*.gas.gz*’ file

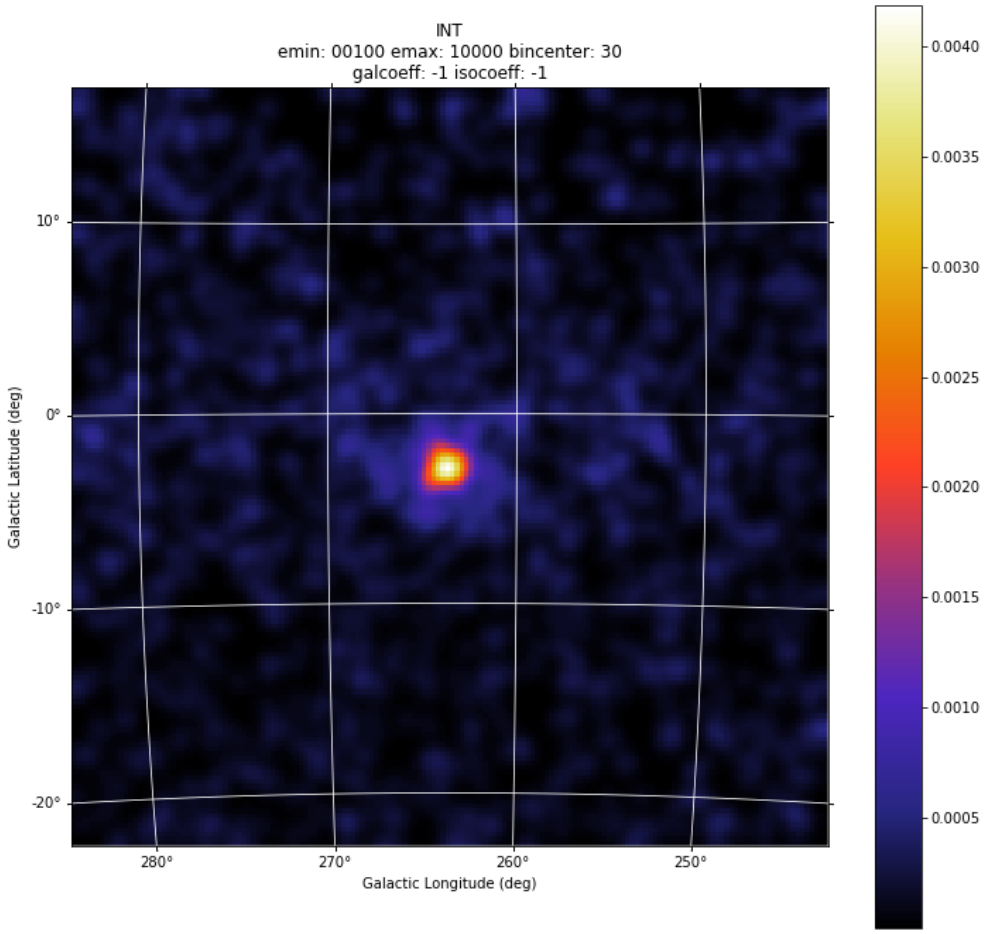
Diffuse emission maps are generated by the procedure AG_gasmapgen embedded into Agilpy. AG_gasmapgen reads an exposure map produced by AG_expmapgen and the master diffuse emission map and outputs a FITS image file, in the same format as the exposure map, in which each pixel contains the diffuse emission in that pixel. The image is a square array in the ARC projection. The diffuse emission map contain models of the diffuse emission convolved with the energy-dependent point spread function and combined into predefined observed energy ranges according to the appropriate energy dispersion function for G events using the FM3.119 background filter. The diffuse emission map automatically selected by Agilepy based on the energy range of the analysis; e.g. if the analysis is performed between 100 MeV and 50 GeV, Agilepy select the file 100_50000.0.1.SFMG_H0025.conv.sky.gz. The first number in the file name is the minimum energy and the second number is the maximum energy, followed by the resolution of the maps (0.1), the background event rejection filter (FM3.119) and the instrument response functions (IRFs).



‘*.int.gz*’ file

Intensity maps are generated by the procedure AG_intmapgen embedded into Agilepy. AG_intmapgen reads an exposure map produced by AG_expmapgen and a counts map produced by AG_ctsmapgen and outputs a FITS image file, in the same format as the counts map, in which each pixel contains the intensity in that pixel. The image is a square array in the ARC projection. The two input maps should have been produced using the

same set of parameters. The intensity map is not used in scientific analysis; it is useful solely as a visualization tool.



Result of the Maximum Likelihood Estimator

Agilepy shows a high-level view of the results of the maximum likelihood estimator. See this link for more details.

The details of the output of the science tool AG_multi that performs the likelihood procedure is still accessible. This section describe the output of the AG_multi science tool, that performs a Maximum Likelyhood Estimator analysis to find the best position, flux and spectral parameters of a list of sources given set of count maps.

Confidence Contour files

If a confidence contour was found, the parameters on the following line describe the best-fit ellipse of the contour, described in detail below.

If source location was requested for a given source and a source location contour was found, then three additional files are generated for that source. These files are written using galactic coordinates in degrees and can be loaded by applications such as ds9 and overlaid on the maps provided as input to AG_multi to visualize the source location contours. One of the three files, with extension .con, contains the source contour as found by the ROOT functions, expressed as a list of galactic coordinates, one point per line, where the last line is a repetition of the first. It may depict any shape. The other two files describe the ellipse that best fits the contour. One has extension .ellipse.con and represents the ellipse as a contour in a format analogous to that of the .con file. The other has extension .reg and describes same ellipse by its axes and orientation.

Determination of the ellipse. If AG_multi was able to find a source contour, an ellipse is fit to the contour. The source contour is a list of points which defines a polygon by connecting each point sequentially. The value of Radius found in the HTML output is the radius in degrees of a circle with the same area as the polygon. AG_multi determines the ellipse which best fits the contour. This ellipse will have the same area as the polygon, and the distance between each contour point and the intersection between the ellipse and the line connecting that point to the centre will be minimized. The ellipse is completely described by three parameters: the two axes and the rotation (in degrees) of the first axis around the centre, as expected by the ds9 application. If the ellipse is a circle, its axes will both be equal to the Radius found in the HTML output. The ellipse is described by two files that are readable by ds9: one is a .reg file which contains the centre, the

axes and the rotation of the ellipse, while the other describes the same ellipse as a list of points in galactic coordinates, thus using the same syntax of a contour file, and has extension .ellipse.con. This is an example of ellipse .reg file:

ExpRatio

Owing to the non-homogeneous sky coverage of the AGILE observations, it is possible that sources lie near the borders of certain pointings. In order to have an unbiased estimate of the coeffcients of the Galactic diffuse emission and isotropic background that could lead to to an incorrect evaluation of the flux and position of the source, exposure uniformity within the region of the analysis is required. We applied a specific check to verify the uniformity of the exposure within the 10-degree radius of the AGILE MLE analysis centred at each source candidate position, over the considered timescale. The fraction of pixels of the exposure map within the region of analysis having a value below a pre-defined threshold was calculated, and if it was more than 10% the region was considered unreliable and the candidate was discarded. The exposure threshold value is evaluated by calculating the mean exposure of the observation over the full FoV area and comparing this exposure with the values of some reference good exposures.

The parameters expratioevaluation, expratio_minthr, expratio_maxthr, expratio_size described here.

Light curves

AGILE-GRID light curves can be created in two different ways:

- using a maximum likelihood estimator analysis
- using aperture photometry.

The likelihood analysis reach better sensitivity, more accurate flux measurement, better evaluation of the backgrounds and can work with a detailed source models where more sources can be considered at the same time.

Aperture photometry provides a raw measure of the flux of a sigle source and is less computing demanding.

The likelihood light curve file contains the results of the generation of a light curve. The columns described are the following sections.

Time of the analysis in MJD:

- time_start_mjd: time start (MJD)
- time_end_mjd: time end (MJD)

Result of the analysis of the single source:

- sqrt(ts): the square root of the Test Statistic value of the results of the maximum likelihood estimator (mle)
- flux (ph/cm2/s/sr)
- flux_err (ph/cm2/s/sr)
- flux_ul (ph/cm2/s/sr)
- gal: the value of the galactic diffuse emission (gal) parameter
- gal_error: the error of the galactic diffuse emission (gal) parameter
- iso: the value of the isotropic emission (iso) parameter
- iso_error: the error of the isotropic emission (iso) parameter
- (l_peak, b_peak): position in Galactic coordinate (l_peak, b_peak): peak coordinates. If it is allowed to vary then they are set to the position for which the TS is maximized.
- dist_peak: distance between current l_peak, b_peak and previous position
- (l, b): position in Galactic coordinate evaluated by mle with the determination of the 95% confidence level elliptical confidence region
- r: radius of 95% c.l. circular confidence region, deg. Statistical error only
- ell_dist: the distance between (l,b) and the initial position
- a: the semimajer axis of the elliptical confidence region
- b: the semiminor axis of the elliptical confidence region
- phi: rotation of the elliptical confidence region
- exposure
- ExpRatio: see above section
- counts

- counts_err
- Index
- Index_Err
- Par2
- Par2_Err
- Par3
- Par3_Err
- Erglog
- Erglog_Err
- Erglog_UL

Time of the analysis in UTC and TT:

- time_start_utc
- time_end_utc
- time_start_tt
- time_end_tt

The following are the initial parameters of the analysis of the source:

- Fix: initial fixflag
- index: initial spectral index
- ULConfidenceLevel
- SrcLocConfLevel
- start_l: initial Galactic coordinate l
- start_b: initial Galactic coordinate b
- start_flux: inital flux for the MLKE
- typefun: type of spectral model
- par2: initial value of par2
- par3: initial value of par3

The following are the parameters of the MLE analysis:

- galmode2
- galmode2fit
- isomode2
- isomode2fit
- edpcor

- fluxcor
- integratortype
- expratioEval
- expratio_minthr
- expratio_maxthr
- expratio_size

Parameters of the maps:

- Emin
- emax
- fovmin
- fovmax
- albedo
- binsize
- expstep
- phasecode

Technical results of the fitting:

- fit_cts
- fit_fitstatus0
- fit_fcn0
- fit_edm0
- fit_nvpar0
- fit_nparx0
- fit_iter0
- fit_fitstatus1
- fit_fcn1
- fit_edm1
- fit_nvpar1
- fit_nparx1
- fit_iter1
- fit_Likelihood1

```
time_start_mjd time_end_mjd sqrt(ts) flux flux_err flux_ul gal g
58026.49921296296 58027.49921296296 7.3538 944.812e-08 213.193e-
58027.49921296296 58028.49921296296 8.87831 1055.24e-08 211.709e
```

```
58028.49921296296 58029.49921296296 7.31495 820.826e-08 198.261e
58029.49921296296 58030.49921296296 6.78978 840.67e-08 208.19e-0
58030.49921296296 58031.49921296296 7.63221 820.4e-08 190.928e-0
```

Advanced configuration

Selection

Option	Type	Default	Required	Description
lonpole	int	180	no	
lpointing	float	TBD	no	l in Galactic coordinates of the center of the pointing during the ‘pointing period’
bpointing	float	TBD	no	b in Galactic coordinates of the center of the pointing during the ‘pointing period’
maplistgen	string	TBD	no	filename of a file for expmapgen with mapspec.fovradmin >> mapspec.fovradmax >> mapspec.emin >> mapspec.emax >> mapspec.index

Maps

Option	Type	Default	Required	Description

Option	Type	Default	Required	Description
offaxisangle	float	30	No	Off axis pointing for mle analysis. Values are between 0 and 60 deg. Agilepy into .maplist

Model

Option	Type	Default	Required	Description
galmode2	int	0	No	Fix the gal parameters using the internal analysis of the MLE. 0) gal0 and gal1 are kept free 1) set gal0 for L0 and gal1 for L1 2) set gal0 for L0 and L1 3) set gal1 for L0 and L1 4) set gal1 - gal1err for L0 and L1 5) set gal1 + gal1err for L0 and L1

Option	Type	Default	Required	Description
galmode2fit	int	0	No	Fix the gal parameters for different energy bins performing a linear fit of gal values evaluated using the internal analysis of the MLE. 0) do not fit 1) pol0 fit 2) powerlaw fit
isomode2	int	0	No	Fix the iso parameters using the internal analysis of the MLE. 0) none 1) set iso0 for L0 and gal1 for L1 2) set iso0 for L0 and L1 3) set iso1 for L0 and L1 4) set iso1 - iso1err for L0 and L1 5) set iso1 + iso1err for L0 and L1
isomode2fit	int	0	No	Fix the iso parameters for different energy bins performing a linear fit of iso values evaluated using the internal analysis of the MLE. 0) do not fit 1) pol0 fit 2) powerlaw fit

mle

Advanced options for optimizer

Option	Type	Default	Required	Description
minimizertype	string	Minuit	No	Use Minuit if position is free. For other values see below.
minimizeralg	string	Migrad	No	For other values see below.
minimizerdefstrategy	int	2	No	Default 2 for Minuit. For other values see below.
mindefaulttolerance	float	0.01	No	See below.
integratortype	int	1	No	1 gauss 2 gaussht 3 gausslefevre 4 gausslefevreht
contourpoints	int	40	No	Number of points to determine the contour (0-400).

minimizertype = Minuit (library libMinuit). Old version of Minuit, based on the TMinuit class. The list of possible algorithms (**minimizeralg**) are:

1) Migrad (default one)
2) Simplex
3) Minimize (it is a combination of Migrad and Simplex)
4) MigradImproved
5) Scan
6) Seek

minimizertype = Minuit2 (library libMinuit2). New C++ version of Minuit. The list of the possible algorithms (**minimizeralg**) :

1) Migrad (default)
2) Simplex
3) Minimize
4) Scan

minimizertype = Fumili . This is the same algorithm of TFumili, but implemented in the Minuit2 library.

minimizertype = GSLMultiMin (library libMathMore). Minimizer based on the Multidimensional Minimization routines of the Gnu Scientific Library (GSL). The list of available algorithms (minimizeralg) is | 1) BFGS2 (default) : second version of the vector Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm; | 2) BFGS : old version of the vector Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm; | 3) ConjugateFR : Fletcher-Reeves conjugate gradient algorithm; | 4) ConjugatePR : Polak-Ribiere conjugate gradient algorithm; | 5) SteepestDescent: steepest descent algorithm;

#*** * GSLMultiFit (library libMathMore). Minimizer based on the Non-Linear Least-Square routines of GSL. This minimizer can be used only for least-square fits.
#*** * GSLSimAn (library libMathMore). Minimizer based on simulated annealing.
#*** * Genetic (library libGenetic). Genetic minimizer based on an algorithm implemented in the TMVA package.

Each minimizer can be configured using the ROOT::Math::MinimizerOptions class. The list of possible option that can be set are:

minimizertype:
Minimizer type (MinimizerOptions::SetMinimizerType(const char *))
.

* Print Level (MinimizerOptions::SetPrintLevel(int)) to set the verbose printing level (default is 0).

mindefaulttolerance:
* Tolerance (MinimizerOptions::SetTolerance(double)) tolerance used to control the iterations.
* Precision (MinimizerOptions::SetTolerance(double)). Precision value in the evaluation of the minimization function. Default is numerical double precision.

- Maximum number of function calls (MinimizerOptions::SetMaxFunctionCalls(int)).
- Maximum number of iterations (MinimizerOptions::SetMaxIterations(int)). Note that this is not used by Minuit. FCN Upper value for Error Definition (MinimizerOptions::SetMaxIterations(int)). Value in the minimization function used to compute the parameter errors. The default is to get the uncertainties at the 68% CL is a value of 1 for a chi-squared function minimization and 0.5 for a log-likelihood function.

minimizerdefstrategy:
* Strategy (MinimizerOptions::SetStrategy(int)), minimization strategy used. For each minimization strategy Minuit uses different configuration parameters (e.g. different requirements in computing derivatives, computing full Hessian (strategy = 2) or an approximate version. The default is a value of 1. In this case the full Hessian matrix is computed only after the minimization.

Advanced options for internal corrections

Option	Type	Default	Required	Description
edpcorrection	float	0.75	No	Perform a flux correction based on EDP evaluation for highest energy channels. Default 0.75, otherwise any value between 0 and 1. EDP correction is enabled only for E>1000 MeV and if fluxcorrection=1, and only for point sources. flux = flux * edpcorrection
fluxcorrection	int	0	No	Perform a flux correction of the flux using the source spectral model and considering that the exposure is calculated with a Power Law with spectral index of 2.1. 0) no correction 1) Flux calculation correction for spectral shape in output 2) correction in input and output

Analysis API

AGBaseAnalysis

AGAnalysis

AGAnalysisWavelet

Engineering API

AGEngAgileOffaxisVisibility

AGEngAgileFermiOffAxisVisibilityComparison

AstroUtils API

class utils.AstroUtils.AstroUtils

static distance(*l1*, *b1*, *l2*, *b2*)

Computes the angular distance between two galatic coordinates.

Parameters:
- **l1** (*float*) – longitude of first coordinate
- **b1** (*float*) – latitude of first coordinate
- **l2** (*float*) – longitude of second coordinate
- **b2** (*float*) – latitude of second coordinate

Returns: the angular distance between (l1, b1) and (l2, b2)

static AP_filter(*filename*, *threshold*, *tstart*, *tstop*, *outpath*)

This function filters an aperture photometry file using a threshold value for exposure, it discards the rows lower than threshold and returns a new file merging the continous rows.

Parameters:
- **filename** (*str*) – path of the aperture photometry file
- **threshold** (*float*) – exposure threshold
- **tstart** (*float*) – time start in UTC
- **tstop** (*float*) – time stop in UTC

Returns: A filtered file result.txt

Source API

Source

Point Source

Science Tools

Name	Short description	Documents
AG_add_diff		
AG_checkMapValue		
AG_ctsmapgen	Generates a counts map from the Agile satellite events list, given a list of time intervals, an energy range and a field of view interval.	
AG_ctsmapgenT	Performs the same selection as AG_ctsmapgen, writing its output to text files for futher automatic usage.	
AG_diff_conv		

Name	Short description	Documents
AG_expmapgen	Generates an exposure map from the Agile satellite log files related to a given a list of time intervals.	ExpmapgenDoc.pdf
AG_expmapgenT	Performs the same selection as AG_expmapgen, writing its output to text files for futher automatic usage.	
AG_gasmapgen	Generates the Galactic diffuse emission maps	
AG_intmapgen	Generates the intensity maps	
AG_iterativeGenSrcList		

Name	Short description	Documents
AG_multi	Performs a Maximum Likelyhood Analysis to find the best position, flux and spectral index of a list of sources to explain the given set of count maps.	AG_multiUserManual.html (outdated) CalcoloTS.html PSF_generation.pdf
AG_multiext	It does the same job as AG_multi also considering a set of extended sources.	AG_multiExtUM.html

Name	Short description	Documents
AG_multisim	Given a list of sources and a set of exposure maps it generates the corresponding counts map by means of a Monte Carlo simulation. It optionally performs on these maps the same analysis of AG_multi. The entire process can be iterated. The generated maps can be saved to disk or just analyzed on the fly.	AG_multisimUM.html

Name	Short description	Documents
AG_multiterative	This application iteratively performs the same analysis of AG_multi, tentatively adding more sources, one at a time, from a second sources list. It produces the best sources list explaining the data.	
AG_pasteMap	This utility program joins two sky maps together adapting their resolution.	
AG_spotfinder		
AG_thetamapgen		

Input files of the MLE

This page provides detailed description of the input files of the science tool AG_multi.

Map list input files

'.maplist4' file

The map list is a text file listing containing at least one line of text. Each line of text describes one set of maps and it is possible to include empty lines or comment lines. The comment lines begin with an exclamation mark.

Each line contains a set of maps:

```
<countsMap> <exposureMap> <gasMap> <offaxisangle> <galcoeff> <isocoeff>
```

where:

- countsMap, exposureMap and gasMap are file system paths pointing to the corresponding sky maps (see `Sky Maps section<../manual/products.html>`_)
- offaxisangle is in degrees;
- galcoeff and isocoeff are the coefficients for the galactic and isotropic diffuse components. If positive they will be considered fixed (but see `galmode and isomode<manual/configuration_file.html#section-model>`_ section).

AGILE format (text file)

The source list is a text file listing at least one source. Each line of text describes one source, and it is possible to include empty lines or comment lines. The comment lines begin with an exclamation mark.

Each source is described by a line containing space-separated values, in the following order:

```
'flux' 'l' 'b' 'index' 'fixflag' 'minSqrt(TS)' 'name' 'locationlimit' 'funtype' 'par2' 'par3' 'i
```

The '*flux*' parameter is expressed in cm^-2 s^-1, and galactic longitude '*l* 'and latitude '*b*' are expressed in degrees.

minSqrt(TS) is the minimum acceptable value for the square root of TS: if the optimized significance of a source lies below this value, the source is considered undetected and will be ignored (set to flux = 0) when considering the other sources.

After the source's name (which should not contain a space), an optional value for the location limitation ('*locationlimit*') in degrees may be provided. If this value is present and not zero, the longitude and latitude of the source will not be allowed to vary by more than this value from its initial position.

According to the *fixflag*, some or all values will be optimized by being allowed to vary.

The '*funtype*' specify the spectral model. PL indicates power-law fit to the energy spectrum; PC indicates power-law with exponential cut-off fit to the energy spectrum; PS indicates power-law with super-exponential cut-off fit to the energy spectrum; LP indicates log-parabola fit to the energy spectrum.

The '*index*' of each source represents the initial estimates of the values for that source (a positive number) and could represent the spectral index of the source (see the following table). The other spectral parameters depend on

the spectral shape of the source. '*index limit min*' and '*index limit max*' specifies the minimum and maximum range where the '*index*' is searched.

The '*par2*' and '*par3*' parameters represent additional spectral parameters in the following table. '*par2 limit min*', '*par2 limit max*', '*par3 limit min*', and '*par3 limit max*' specify the minimum and maximum range of the '*par2*' and '*par3*' respectively.

funtype	**spectral model**		**index**	**par2**	**par3**	
0	PL	PowerLaw	x^(-[index])	α		
1	PC	PLExpCutoff	x^(-[index]) * e^(- x / [par2])	α	Ec	
2	PS	PLSuperExpCutoff	x^(-[index]) * e^(- pow(x / [par2], [par3]))	α	Ec	β
3	LP	LogParabola	(x / [par2]) ^ (-([index] + [par3] * log (x / [par2])))	α	Ec	β

The match of the parameteres is:

- *index* = α: Spectral index for PL, PC, and PS spectral models, first index for LP spectral model; Could be Index or Index1 in the XML format
- *par2* = Ec (MeV): cut-off energy for PC and PS spectral models, pivot energy for LP spectral model;
- *par3* = β: Second index for PS spectral models, curvature for LP spectral model;

The usual energy range used to calculate these parameters is 100 MeV – 10 GeV. The MLE procedure calculates also the 1σ uncertainty of the spectral parameters:

- Δα: Statistical 1σ uncertainty of α;
- ΔEc (MeV): Statistical 1σ uncertainty of Ec;
- Δβ: Statistical 1σ uncertainty of β.

According to the '*fixflag*' some or all of those values will be optimized by being allowed to vary. The fixflag is a bit mask, each bit indicating whether the corresponding value is to be allowed to vary:

fixflag = 0 everything is fixed (free="0")
fixflag = 1 indicates the flux (free="1" in <parameter name="Flux">)
fixflag = 2 the position is free (free="1" in <spatialModel type="PointSource">)
fixflag = 4 the Index or Index1 is free (free="1" in <parameter name="index"> or <parameter name="index1">)
fixflag = 8 the par2 is free (free="1" in <parameter name="cutoffEnergy"> or <parameter name="pivotEnergy">)
fixflag = 16 the par3 is free (free="1" in <parameter name="index2"> or <parameter name="curvature">)
fixflag = 32 force position to be variable only in Loop2 (free="2" in <spatialModel type="PointSource">)

The user may combine these values, but the flux will always be allowed to vary if at least one of the other values are.

	flux	pos(free=1)	index/index1	par2=cutoffEnergy/pivotEnergy	par3=index2/curvature	pos(free=
fixflag	1	2	4	8	16	32

Examples:
fixflag = 0: everything is fixed. This is for known sources which must be included in order to search for other nearby sources.
fixflag = 1: flux variable, position fixed
fixflag = 2: only the position is variable, but MLE will let the flux vary too, so this is equivalent to 3.
fixflag = 3: flux and position variable, *index* fixed
fixflag = 4: *index* variable (and flux variable)
fixflag = 5: flux and *index* variable, position fixed
fixflag = 7: flux, position and *index* variable and also
fixflag = 28: *index*, *par2* and *par3* variable (and flux variable)
fixflag = 30: position, *index*, *par2* and *par3* variable (and flux variable)
fixflag = 32: position=2, the rest is fixed

Output files of the MLE

This page provides detailed description of the output files of the science tool AG_multi.

The details of the output of the science tool AG_multi that performs the likelihood procedure is still accessible from agilepy. This section describes the “low level” results of the AG_multi procedure. The results are available in the $HOME/agilepy_analysis/<sourcename>_<username>_<date>-<time>/mle directory, where <sourcename> and <username> are defined in the yaml configuration file, <date> and <time> are defined by the system when the analysis starts.

At the end of the fitting process AG_multi generates two main files, describing the most relevant results for all the sources, and a set of source-specific files containing more detailed data about that source.

One of the two main files is in HTML format, and it includes both the input and output data grouped in tables. Having a look at this file the user should quickly understand the outcome of the fitting process and its main results. The next section describes the HTML output in more detail.

The second of the two main files contains the same data printed in text format. This file is divided in two sections. The first contains one line for each diffuse component and the second one line for each source. The first line of each section begins with an exclamation mark (a comment line for many applications) labeling the values printed beneath. In each line the values are separated by a space. This is an example of the text output of the analysis of the 2AGLJ2254+1609 (3C454.3) with the test dataset provided. For this analysis, only one set of maps and one source is used. The iotropic emission components coefficients are kep free and symmetric errors are provided. The flux and position of the source are allowed to vary, while the spectral index is fixed. The name, significance of the source detection, position, source counts with error, source flux with error, and spectral index with error are provided.

```
! DiffName, Flux, Err, +Err, -Err
Galactic 0.7 0 0 0
Isotropic 8.79898 0.969867 0.984804 -0.955381
! SrcName, sqrt(TS), L_peak, B_peak, Counts, Err, Flux, Err, Inc
2AGLJ2254+1609 35.5482 86.0638 -38.1753 719.369 35.2059 2.63371e
```

index, par2, par3 and related errors depend by the spectral mode used.

The counts and fluxes are provided, as well as their errors if the flux is allowed to vary. Finally, the spectral index and its error, if applicable, are provided.

Note

If a source is outside the Galactic plane, fix the diffuse emission coefficient parameter (gal) to 0.7 with ag.setOptions(galcoeff=[0.7])

‘*.source*’ file

The .source file is an internal technical file produced by the maximum likelihood estimator mle() procedure for each source. It contains all the analysis results for each source that is part of the ensemble of models. Agilepy extract from this .source file the most important parameters useful for the final user.

When possible, two additional files describing the source contour (possibile only if position is kept free).

The text file contains some comment-like lines (first character is an exclamation mark) labeling the values printed beneath. This is an example of text output, consistent with the example given above:

```
! Label Fix index ULConfidenceLevel SrcLocConfLevel start_l star
! sqrt(TS)
! L_peak B_peak Dist_from_start_position
! L B Dist_from_start_position r a b phi
! Counts Err +Err -Err UL
```

```
! Flux(ph/cm2s) [0 , 1e+07] Err +Err -Err UL(ph/cm2s) ULbayes(ph
! Index [0.5 , 5] Index_Err Par2 [20 , 10000] Par2_Err Par3 [0 ,
! cts fitstatus0 fcn0 edm0 nvpar0 nparx0 iter0 fitstatus1 fcn1 e
! Gal coeffs [0 , 100] and errs
! Gal zero coeffs and errs
! Iso coeffs [0 , 100] and errs
! Iso zero coeffs and errs
! Start_date(UTC) End_date(UTC) Start_date(TT) End_date(TT) Star
! Emin..emax(MeV) fovmin..fovmax(deg) albedo(deg) binsize(deg) e
! Fit status of steps ext1, step1, ext2, step2, contour, index,
! Number of counts for each step (to evaluate hypothesis)
! skytypeL.filter_irf skytypeH.filter_irf
2AGLJ2254+1609 1 2.20942 2 5.99147 86.1236 -38.1824 2.64387e-05
47.8468
86.1236 -38.1824 0
-1 -1 -1 -1 -1 -1 -1
718.633 31.0247 31.4119 -30.6392 782.234
2.64387e-05 1.14141e-06 1.15565e-06 -1.12722e-06 2.87787e-05 2.0
2.20942 0 0 0 0 0
909 -1 2456.44 0.5 0 8 3 0 1311.78 7.28513e-16 1 8 3 1828.16
0.7 0
0.7 0
8.83231 0
8.83231 0
2010-11-13T00:01:06 2010-11-21T00:01:06 216691200.0000000 217382
100..10000 0..60 80 0.25 0 6 0
-1 -1 -1 0 -1 -1 0
-1 2124 -1 2124 -1 -1 2124
SKY002.SFMG_H0025 SKY002.SFMG_H0025
```

The counts and fluxes are provided, as well as their symmetric, positive, and negative errors if the flux is allowed to vary. For convenience, the exposure of the source, used to calculate the source counts from the flux, is also provided. Finally, the spectral index and its error, if applicable, are provided.

‘*.source*’ Attributes

Parameter name	UM	Description	default	range
Label				
Fix		Value of the *fixflag*		
index		Initial value of the *index* parameter		
ULConfidenceLevel		Upper limit confidence level espressed as sqrt(TS)	2	
SrcLocConfLevel		Source location contour confidence level %	95 99 95 68 50	95
start_l				
start_b				
start_flux	(ph/cm2s)			

Parameter name	UM	Description	default	range
[lmin lmax]				
[bmin bmax]				
typefun				
par2				
par3				
galmode2				
galmode2fit				
isomode2				
isomode2fit				
edpcor				
fluxcor				
integratortype				

Parameter name	UM	Description	default	range
expratioEval				
expratio_minthr				
expratio_maxthr				
expratio_size				
[index_min index_max]				
[par2_min par2_max]				
[par3_min par3_max]				
contourpoints				
minimizertype				
minimizeralg				
minimizerdefstrategy				

Parameter name	UM	Description	default	range
minimizerdeftol				
sqrt(TS)				
L_peak				
B_peak				
Dist_from_start_position				
L				
B				
Dist_from_start_position				
r				
a				
b				
phi				

Parameter name	UM	Description	default	range
Counts				
Err				
+Err				
-Err				
UL				
Flux	(ph/cm2s)			
Err				
+Err				
-Err				
UL	(ph/cm2s)			
ULbayes	(ph/cm2s)			
Exp	(cm2s)			

Parameter name	UM	Description	default	range
ExpSpectraCorFactor				
Erglog	(erg/cm2s)			
Erglog_Err				
Erglog_UL	(erg/cm2s)			
Sensitivity				
FluxPerChannel	(ph/cm2s)			
Index				
Index_Err				
Par2				
Par2_Err				
Par3				
Par3_Err				

Parameter name	UM	Description	default	range
cts				
fitstatus0				
fcn0				
edm0				
nvpar0				
nparx0				
iter0				
fitstatus1				
fcn1				
edm1				
nvpar1				
nparx1				

Parameter name	UM	Description	default	range
iter1				
Likelihood1				
Gal coeffs				
errs				
Gal zero coeffs				
errs				
Iso coeffs				
errs				
Iso zero coeffs				
errs				
Start_date(UTC)				
End_date(UTC)				

Parameter name	UM	Description	default	range
Start_date(TT)				
End_date(TT)				
Start_date(MJD)				
End_date(MJD)				
Emin..emax	MeV			
fovmin..fovmax	deg			
albedo	deg			
binsize	deg			
expstep				
phasecode				
ExpRatio				
Fit status of steps ext1				

Parameter name	UM	Description	default	range
Fit status of steps step1				
Fit status of steps ext2				
Fit status of steps step2				
Fit status of steps contour				
Fit status of steps index				
Fit status of steps ul				
Number of counts for ext1				
Number of counts for step1				
Number of counts for ext2				
Number of counts for step2				

Parameter name	UM	Description	default	range
Number of counts for contour				
Number of counts for index				
Number of counts for ul				
skytypeL.filter_irf				
skytypeH.filter_irf				

‘*.source.con*’ file

outfile.source.con: source contour (if found).

‘*.source.reg*’ file

outfile.source.reg: ellipse best fitting the source contour (if found).

‘*.log*’ file

Log file with a line for each step of the fitting process.

HTML output

AG_multi provides an HTML output of the results. The HTML output file is divided into two sections, input and output. The input section contains three subsections: the command line options, the map list and the source list contents. The command line options are listed in two tables, one with the names of the IRFs (PSD, SAR and EDP) files, the other with the rest of the command line. The maplist subsection also contains two tables. The first lists the mapfile contents and the second contains the data from the map files themselves. This table contains one map per row, and each column contains one value only if it is the same for all the maps. The last table of the input section contains the source list contents. The output section is also divided into three subsections. The first is a table showing the Galactic and isotropic coefficients and their errors. Also in this table some cells may be grouped together when the values are all the same. The second is a table showing the fit results for the sources and their errors. One of the listed values is the contour equivalent radius, explained in the next section. The last table shows the source flux per energy channel, and it is present only when different energy channels are considered. This table has one row for each source and one column for each energy channel.

Need Help

If you have troubles please email to:

- leonardo.baroncelli[at]inaf.it
- antonio.addis[at]inaf.it
- andrea.bulgarelli[at]inaf.it

or open an issue of GitHub:
https://github.com/AGILESCIENCE/Agilepy/issues

Known issues

- The unit test “test_aperture_photometry” fails on macos, therefore the aperturePhotometry() method is not available on this OS.
- Each notebook should instantiate only one AGAnalysis object, otherwise the logger will be duplicated.
- In certain situations %matplotlib widget has a weird behaviour. If you have problems with map sizes or interactions, comment the line %matplotlib widget

Development

Install the development environment

If you want to try agilepy's new features that are not officially released yet, a develpoment environment called agilepy-environment is available into Anaconda cloud. It contains all the dependencies unless agilepy, which must be installed by hand cloning the repository.

Anaconda

```
conda config --add channels conda-forge
conda config --add channels plotly
conda create -n agilepydev -c agilescience agiletools agilepy-da
conda activate agilepydev
git clone https://github.com/AGILESCIENCE/Agilepy.git
cd Agilepy && git checkout develop
conda env update -f environment.yml
python setup.py develop
```

Docker

```
docker pull agilescience/agilepy-recipe:latest
mkdir shared_dir && cd shared_dir && git clone https://github.co
&& cd Agilepy && git checkout develop

docker run --rm -it -p 8888:8888 \
-e DISPLAY=$DISPLAY \
-v /tmp/.X11-unix:/tmp/.X11-unix:rw \
-v $SHARED_DIR_PATH:/shared_dir \
agilescience/agilepy-recipe:latest

## -- Inside the container --
conda activate agilepydev
cd /shared_dir/Agilepy python setup.py develop
```

```
jupyter notebook --port 8889 --ip 0.0.0.0  --allow-root
```

Now you have the agilepy's latest version installed in your environment.

Git flow

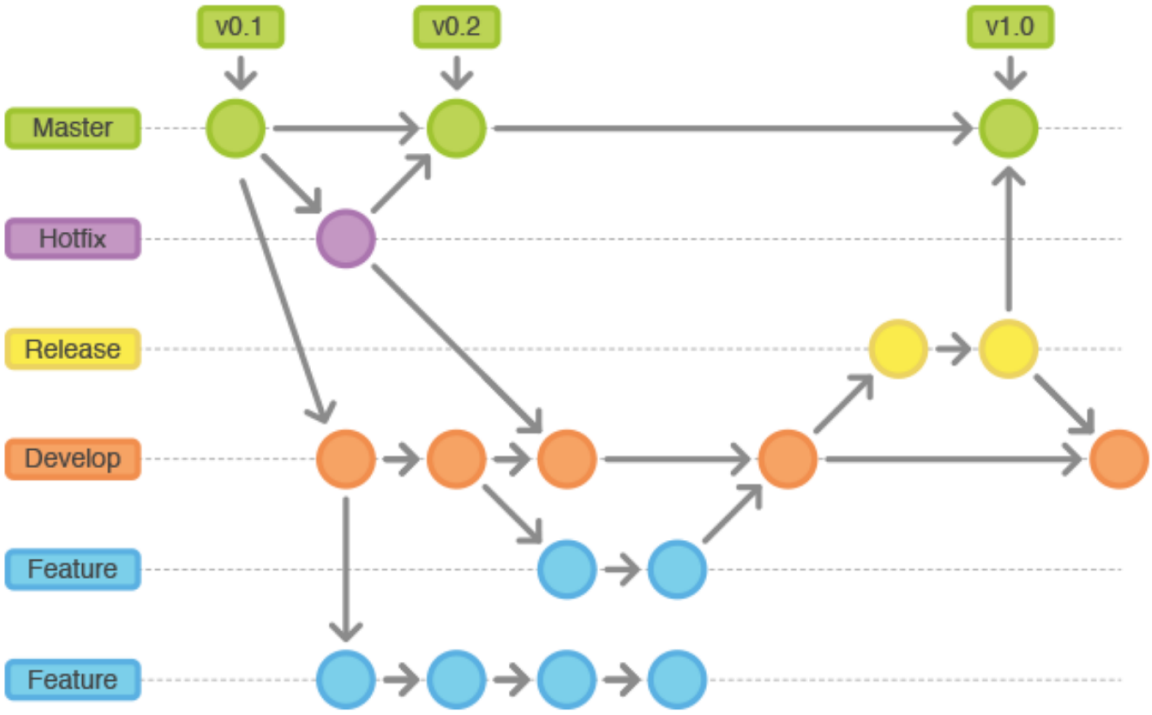
Branches

Two main branches:

- **master**: contains only production releases.
- **develop**: contains commits that will be included in the next production release.

Two support branches:

- **feature** branch: each new feature (Trello's card) should be developed in its own feature branch, branching from **develop** and merged back into it. The **feature** branch are not pushed into the remote.
- **hotfix** branch: if an hotfix is needed it should be develop in its own branch, branching from **master** and merged back to it.



Versioning

The **master** branch contains only production releases: when the **develop** branch (or **hotfix** branch) is merged to **master** a new release tag must be created. Its name follows the [semantic versioning](https://semver.org/) [https://semver.org/].

x.y.z

Incrementing:

- x version when you make incompatible API changes,
- y version when you add functionality in a backwards compatible manner, and
- z version when you make backwards compatible bug fixes.

Branches names

The **master** and the **develop** branch have an infinite lifetime, hence their name is fixed.

The **feature** branch takes the following format:

feature-#<card-number>-<short-description>

e.g. feature-#61-new-cool-feature

The **hotfix** branch name takes the following format:

hotfix-#<card-number>-<release-number>

e.g. hotfix-#57-1.0.0

The release number is the one of the production release from which it originates from.

Getting started

Development of a new feature

Create a new **feature** branch:

```
git checkout develop
git pull origin develop
git checkout -b feature-#61-new-cool-feature develop
```

Development and testing of the new feature.

When you have finished, update the CHANGELOG.md and commit your changes.

```
vim CHANGELOG
git commit -m "feature-#61-new-cool-feature done"
```

In the meantime it is possible that someone else have pushed his work into the develop branch. In this case you have to merge the changes in your feature branch.

```
git pull **origin** develop
```

Finally you can merge your feature branch back to **develop** branch.

```
git merge --no-ff feature-#61-new-cool-feature
git branch -d feature-#61-new-cool-feature
git push origin develop
```

Add configuration parameters

Let's say we want to add the following configuration section to the AGAnalysis' configuration file.

```
ap:
    radius: 0.25
    timeslot: 3600
```

- Add the new section to the AGAnalysis.getConfiguration() method.
- Add the type of the configuration parameters within the AGAnalysisConfig.checkOptionsType() method (in the corresponding lists).
- If the parameters need some kind of validation (this is not the case), add a new method in ValidationStrategies and call it within the AGAnalysisConfig.validateConfiguration() (check examples).
- If the parameters need some kind of transformation (this is not the case), add a new method in CompletionStrategies and call it within the AGAnalysisConfig.completeConfiguration() (check examples).
- Add the new configuration section to all the unit test configuration files.
- Document the new configuration parameters within the manual/configuration_file.rst file.

Add a new science tool

Let's say we want to add a new (c++) science tool: AG_ap.

- Add a new class within the api/ScienceTools.py script. You need to implement some abstract methods.
- You can use the new class as follows:

```
apTool = AP("AG_ap", self.logger)
apTool.configureTool(self.config)
if not apTool.allRequiredOptionsSet(self.config):
    raise ScienceToolInputArgMissing("Some options have not beer
products = apTool.call()
```

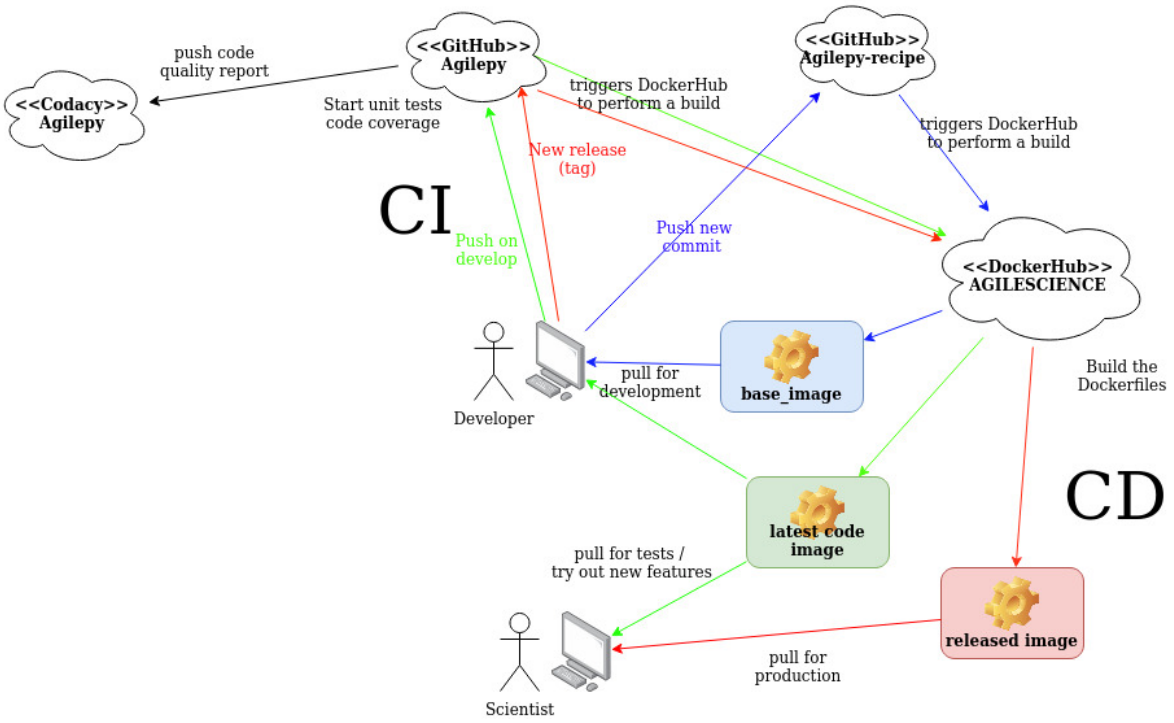
Release of a new version

Change the version of the software in setup.py. The version increment must be take in account all the commits of the **develop** branch. You can check the CHANGELOG.md to facilitate this process. Please, add the new tag within the CHANGELOG.md file.

```
git checkout master
git merge --no-ff develop
git tag -a <new-tag>
git push origin <new-tag>
```

DevOps

A high level description of agilepy's devops is in the image below:



This scheme workflow produces three images:

- **base_image**: It's an image with all the dependencies except Agilepy python library, it's used for developing purposes only by developers. Base image is built after a new commit in agilepy-recipe repository.
- **latest code image**: It's the base_image with Agilepy's develop branch at latest commit, useful for using or testing agilepy's updates not officially released. This image is not supported nor stable and is built by dockerhub after github's testing pipelines are successful.
- **released image**: The base_image with Agilepy's release tag. By default the community shall be download this image. It's built when a new tag is created.

Index

A | **D**

A

AP_filter() (utils.AstroUtils.AstroUtils static method)

AstroUtils (class in utils.AstroUtils)

D

distance() (utils.AstroUtils.AstroUtils static method)

AG_add_diff

AG_checkMapValue

AG_multi

AG_multi is a command line application running under Linux 64 bits, or Mac OS X. The aim of this application is to find the best values for the flux, the position and/or the spectral index of a list of gamma ray sources to explain a set of experimental data. The user will provide a list of maps containing maps of photons detected by the AGILE satellite during one or a series of observations, together with maps of the instrument exposure during those observations and the corresponding Galactic diffuse emission models. The user will also provide a list of sources that may explain the photons detected, giving a guess for the flux, position and spectral index of those sources. AG_multi will find the best values for the sources to fit the data using the method of maximum likelihood, estimating the improved likelihood due to the presence of each source in the list. The user has a variety of options to influence the process as explained in the following.

The Command Line

The command line is internally managed by the parameter interface library (PIL) developed by the INTEGRAL Science Data Centre (ISDC). Each command line option is described by a .par file, AG_multi.par in this case, a sample of which comes with the distribution. The environment variable PFILES should be defined in your account, pointing to the directory where the file AG_multi.par resides. The user has two ways to specify the option values in the command line. One is to specify the option name and its value (in any order), the other is to give just the option values in the order they appear in the .par file. For example

```
AG_multi option1=value1 option2=value2 option3=value3...
```

or

```
AG_multi4 value1 value2 value3...
```

If any of the command line options are missing, AG_multi will prompt the user to either confirm the previously used value or to provide a new one. The values used in the current session will be stored and used in the next session. This behaviour depends on the .par file that comes with the distribution, which the user may change. Refer to the PIL library online documentation for all the details.

List of parameters:

Name	**Type**	**Description**
maplist	String	Name of a text file containing the list of the maps
sarfile	String	SAR file name
edpfile	String	EDP file name
psdfile	String	PSD file name
ranal	Real	Radius of analysis (degrees)
galmode	Integer	Galactic parameter mode
isomode	Integer	Isotropic parameter mode

Name	Type	Description
srclist	String	The name of a text file containing the the list of the gamma ray sources to fit to the list of maps above.
outfile	String	The name of the main text output file - the title of the HTML output - the prefix of the other output file names.
ulcl	Real	Upper limit confidence level
loccl	Real	Location contour confidence level

Input files

A detailed description of the input files of the MLE is provided here .

Output files

A detailed description of the output files of the MLE is provided here .

Technical Documents

PSF_generation.pdf

AG_multiext

AG_multisim

AG_multiterative