softWAre for Dark matter ExpeRiments with Skippers

WADERS Framework

pronounced [v AI d e r s]

Exercises

gitlab repo: https://gitlab.in2p3.fr/damicm/pysimdamicm

official web for documentation: https://ncastell.web.cern.ch/ncastell/

DQM Compton web: (documentation **for analysis**) https://gev.uchicago.edu/compton/

Núria Castelló-Mor

IFCA Instituto de Física de Cantabria
CSIC Consejo Superior de Investigaciones Científicas
UC Universidad de Cantabria

1st DAMIC-M software school

11-13 January 2021
Online

https://indico.in2p3.fr/event/22866/

List of exercises

- **Ex1.** Download & Install
  At Lyon, in gev (not zev!), in your laptop, ... but it must have ROOT v6.XX against python3.X

Related to simulations:

- **Ex2.** Use the debug mode to see the geant4 energy depositions, the e-h pair created after diffusion is applied, and cluster found

- **Ex3.** Mimic only the detector response (no noise added) to get the reconstructed spectral energy of the clusters, and inspect all output trees

Related to data ... using skipper images:

Ex4. Launch panaSKImg

- **Ex4.1** to get the average of a single skipper image, of a set of skipper images
- **Ex4.2** to compress an skipper image by applying the STD
- **Ex4.3** to get the pedestal subtracted image: pedestal per row, in the overscan without masking
- **Ex4.4** to fit the dark current, the gain, and the single e- resolution
- **Ex4.5** to plot the single-to-noise ration vs numer of skips (in blocks of 15 skips)
- **Ex4.6** To check if there are charge losses between skip measurements
- **Ex4.7** as a DQM
- ... look into the notebooks
- ... try to add a new process

Recap: How to use WADERS

Use on of the two executables: psimulCCDimg or panaSKImg

```
psimulCCDimg --help

psimulCCDimg --json
```

```
psimulCCDimg psimulCCDimg_LBC_config.json --g4file ../DAMICM/G4Run/LablCeiling_241Am.root --g4out ../outputs
                  --event 5 --cls-pix-min 5
```

executable | **JSON cofig file** | **input file[s]** | **output directory**

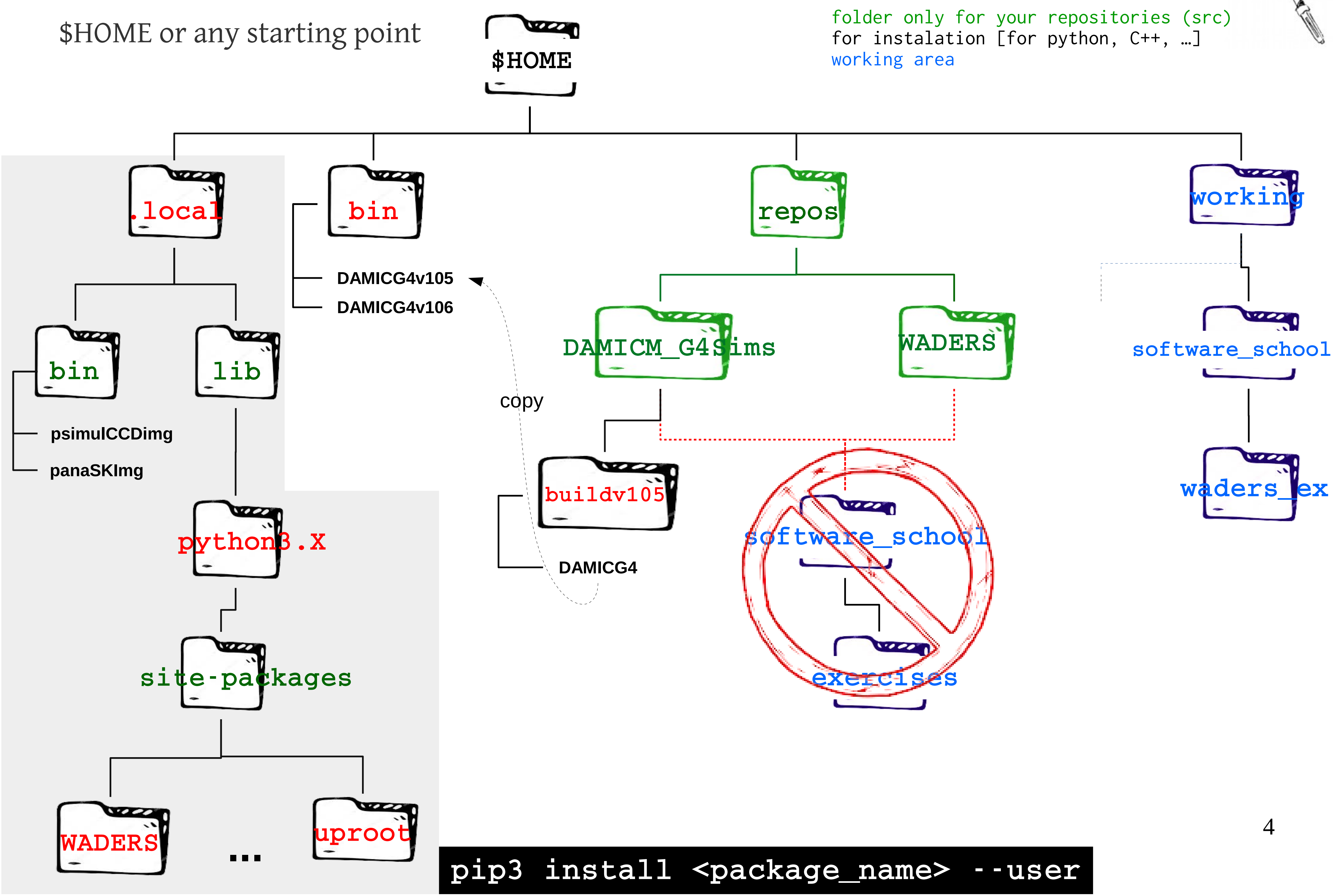
```
panaSKImg --json panaSKImg_compton_config.json "/data/compton/data/*Image*Source*fits" -o ../Run145/avg
          --save-plots --save-img --verbose --display --verbose
          -s CompressedSkipperProcess,PedestalSubtractionProcess,SignalPatterRecognition,ClusterFinder
          ...
```

```
panaSKImg --help

panaSKImg --list-processes help
```

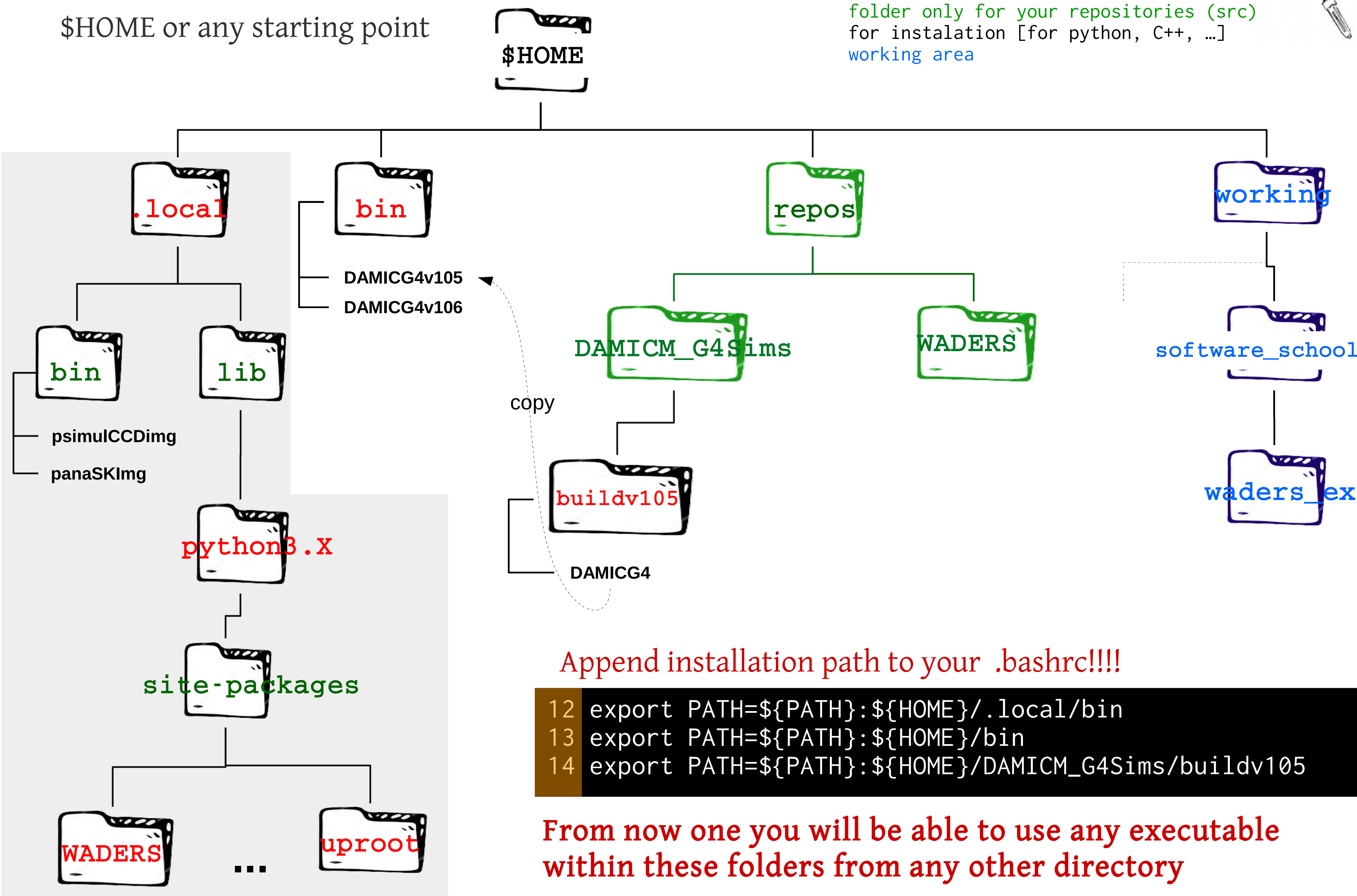

other useful command lines

Best practice in organizing when working with codes



`pip3 install <package_name> --user`

Best practice in organizing when working with codes



Append installation path to your .bashrc!!!!

```
12 export PATH=${PATH}:${HOME}/.local/bin
13 export PATH=${PATH}:${HOME}/bin
14 export PATH=${PATH}:${HOME}/DAMICM_G4Sims/buildv105
```

From now one you will be able to use any executable within these folders from any other directory

Let's use WADERS!

```
user@ darkB612$ ssh -X username@cca.in2p3.fr
```

```
user@ darkB612$ ssh -Y username@cca.in2p3.fr
```

Find both presentations and data for exercesis at
Lyon: /sps/hep/damic/school/waders_exercises/

Ex 1. Download & Install (I)

The first we need is to download and install **WADERS**

1) **Create** a **folder** to **clone** the WADER **repository** [if you do not have one yet]
2) **Clone repo** from gitlab.in2p3
3) **Select** the last **stable version**: **v3.0.1**
4) Look into file **`requirements.txt`**

   Do you have already intalled some of the packages listed in requirements.txt file?
   Conflicts with version?
   Do you have a **higher version** of **uproot**? **If yes, go to next slide**

5) **Install** ALL NEEDED **REQUIREMENTS** using **`pip3`** (make sure is python3)
6) **Install WADERS** using **pip3**
7) **Check** that **WADERS** has been **installed** in your $HOME/.local
   - Executable at $HOME/.local/bin (do you have panaSKImg and psimulCCDimg?
   - Modules at $HOME/.local/lib/python3.6/site-packages/waders

```
  user@ repos$ ccenv root
1 user@ repos$ cd $HOME
  user@ repos$ mkdir repos
  user@ repos$ cd repos
2 user@ repos$ git clone https://gitlab.in2p3.fr/damicm/pysimdamicm.git
  user@ repos$ cd pysimdamicm
3 user@ repos$ git checkout v3.0.1
4 user@ repos$ vim requirements.txt
5 user@ repos$ pip install -r requirements.txt —user
6 user@ repos$ pip install . --user
7 user@ repos$ ls $HOME/.local/bin
```

Ex 1. Download & Install (II): ONLY if you have uproot already installed

The first we need is to download and install the python package: **WADERS**

1) **Create** a **folder** to **clone** the WADER **repository** [if you do not have one yet]
2) **Clone repo** from gitlab.in2p3
3) **Select** the last **stable version**: **v3.0.1**
4) **Create** a **virtual environment** (for instance $**HOME/repos/WADERS/venv_waders**)
5) **Activate** the **virtual environment**
6) **Install** ALL NEEDED **REQUIREMENTS** using `pip3` (make sure is python3)
7) **Install WADERS** using **pip3**
8) **Check** that **WADERS** has been **installed**. **NOTE** that with the **virtual environment**, WADERS has been installed at
   `$HOME/repos/WADERS/venv_waders/bin`

9) Append this path to $PATH in your .bashrc (see slide 5)

```
  user@ repos$ ccenv root
1 user@ repos$ cd $HOME
  user@ repos$ mkdir repos
  user@ repos$ cd repos
2 user@ repos$ git clone https://gitlab.in2p3.fr/damicm/pysimdamicm.git
  user@ repos$ cd pysimdamicm
3 user@ repos$ git checkout v3.0.1
4 user@ repos$ python -m venv venv_waders
5 (venv_waders) user@ repos$ source v3.0.1
6 (venv_waders) user@ repos$ pip install -r requirements.txt
7 (venv_waders) user@ repos$ pip install .
8 (venv_waders) user@ repos$ ls venv_waders/bin
```

Ex 2. **psimulCCDimg** using **debug mode**

Several Geant4 simulations can be found at

`/sps/hep/damic/school/waders_exercises/g4sims`

- 241Am: ^{241}Am source in the Compton setup
- alpha: alpha particles in the CCDSensor volume (LBC setup)
- 60Co: ^{60}Co in the kapton cable volume (LBC setup)

CCD size: for the LBC 4000x6000, for the Compton setup 1024x6176 assuming a pixel size in both of 0.015 mm

Goal: *Use the **debug mode** (--debug). This will allow you to see the steps followed between G4 simulations (energy depositions) and the final produc (the clusters).*

Remember: use and independent folder to work (see slide 4)

1. Go to your working directory and copy the default configuration file from your repository into your working area (for instance **/sps/hep/damic/working/software_school/waders_ex/sims**)

```
user@ cca.in2p3$ cd /sps/hep/damic/working/software_school/waders_ex/sims
user@ cca.in2p3$ cp $HOME/repos/pysimdamicm/pysimdamicm/json/psimulCCDimg_config_file.json .
```

Configure the JSON file

2. Set processes to mimic the detector response without including any type of noise
3. Include ClusterFinder but not PatterRecognition
4. Check the configuration parameters of the processes (remember you have the command line --json to display a short description for each configurable parameter)
5. Set the properties of your CCD depending on the ROOT file you choose (241Am, alpha,60Co):
    1. *size of the CCD, pixel size*

psimulCCDimg (see slide 3)

6. *Launch psimulCCDimg with the debug mode [zoon in out, see output messages,...]*

Ex 3. **psimulCCDimg** using **production mode**

Several Geant4 simulations can be found at

`/sps/hep/damic/school/waders_exercises/g4sims`

- 241Am: ^{241}Am source in the Compton setup
- alpha: alpha particles in the CCDSensor volume (LBC setup)
- 60Co: ^{60}Co in the kapton cable volume (LBC setup)

CCD size: for the LBC 4000x6000, for the Compton setup 1024x6176 assuming a pixel size in both of 0.015 mm

Goal: *Mimic only the detector response (no dark current neither electronic noise) to get the reconstructed spectral energy of the clusters. Inspect all outputs trees. Sorry, the used root files has really low statistics!*

Remember: use and independent folder to work (see slide 4)

1. Go to your working directory and copy the default configuration file from your repository into your working area (for instance **/sps/hep/damic/working/software_school/waders_ex/sims**)

```
user@ cca.in2p3$ cd /sps/hep/damic/working/software_school/waders_ex/sims
user@ cca.in2p3$ cp $HOME/repos/pysimdamicm/pysimdamicm/json/psimulCCDimg_config_file.json .
```

Configure the JSON file

2. Set processes to mimic the detector response excluding the dark current and electronic noise
3. Include ClusterFinder but not PatterRecognition
4. Check the configuration parameters of the processes (remember you have the command line --json to display a short description for each configurable parameter)
5. Set the properties of your CCD depending on the ROOT file you choose (241Am, alpha,60Co):
    1. *size of the CCD, pixel size*

psimulCCDimg (see slide 3)

6. *Launch psimulCCDimg and by reading the output messages check that your simulations has been successfully completed*
7. *Inspect your output root file*

Ex 4.1 **panaSKImg** for skippers

Several skipper images from the Compton setup can be found at

`/sps/hep/damic/school/waders_exercises/compton_run107`

Where you can find

- 209 images with 64 skips: `Image_Am241_Source_107_<id>.fits`
- 1 image with 2000 skips: `Image_Am241_Source_107_skip_1.fits`

Goal: *Get the **averaged image** of an skipper, i.e. compress all single skip measurements per pixel (64 or 2000 depending on the image used) to its average.*
Do it for one image only and using the option --display, and then for a set of images without the option –display

https://gev.uchicago.edu/compton/howto_notebooks/analysis/II_howto_CompressSkipperProcess.html

Remember: use and independent folder to work (see slide 4)

1. Go to your working directory and copy the default configuration file from your repository into your working area (for instance **/sps/hep/damic/working/software_school/waders_ex/skippers**)

```
user@ cca.in2p3$ cd /sps/hep/damic/working/software_school/waders_ex/sims
user@ cca.in2p3$ cp $HOME/repos/pysimdamicm/pysimdamicm/json/panaSKImg_configuration.json .
```

Configure the JSON file

Note that the input section has been set for this skipper images. If you want to use your own images, take care and set it properly (keywords for the header, axis to compress, overcan regions, ...)

2. Set only the process to get the averaged image (remember you have the command --help to see a short description for all configuration parameters)
3. Set the config values for each process to get what you want

panaSKImg (see slide 3)

4. Launch panaSKImg using the command line -s (instead of using the parameter sequence in the json file). If your internation connection is goot display all plots (--display) and verbose (--verbose), if not just record them (--save-plots). You can also store the compressed image using --save-img
5. Takle a look into the output messages and plots ...

→ Re-do it and now ignore skip measurements from 0 up to 10

Ex 4.2 **panaSKImg** for skippers

Several skipper images from the Compton setup can be found at

`/sps/hep/damic/school/waders_exercises/compton_run107`

Where you can find

- 209 images with 64 skips: `Image_Am241_Source_107_<id>.fits`
- 1 image with 2000 skips: `Image_Am241_Source_107_skip_1.fits`

Goal: *Now get the **compressed image** by using the standard deviation function (std) instead of the mean.*
Do it for one image using the option –display
Can you do ex 4.1 and 4.2 in one step?
For does who know numpy, This process to compress skippers accepts any statistical function that has as an argument axis (rows:0, columns:1).

https://gev.uchicago.edu/compton/howto_notebooks/analysis/II_howto_CompressSkipperProcess.html

Remember: use and independent folder to work (see slide 4)

1. You can re-use the same configuration file than in the previous ex

Configure the JSON file

Note that the input section has been set for this skipper images. If you want to use your own images, take care and set it properly (keywords for the header, axis to compress, overcan regions, ...)

2. Set only the process to get the averaged image (remember you have the command --help to see a short description for all configuration parameters)
3. Set the config values for each process to get what you want

panaSKImg (see slide 3)

4. Launch panaSKImg using the command line -s (instead of using the parameter sequence in the json file). If your internation connection is goot display all plots (--display) and verbose (--verbose), if not just record them (--save-plots). You can also store the compressed image using --save-img
5. Takle a look into the output messages and plots ...

Ex 4.3 **panaSKImg** for skippers

Several skipper images from the Compton setup can be found at

`/sps/hep/damic/school/waders_exercises/compton_run107`

Where you can find

- 209 images with 64 skips: `Image_Am241_Source_107_<id>.fits`
- 1 image with 2000 skips: `Image_Am241_Source_107_skip_1.fits`

Goal: Estimate and subtract *the pedestal from the* ***averaged image:*** *using only the* ***overscan*** *and fitting a gaussian ('****gauss****_fit' method) for each* ***row****.*
Can you estimate the pedestal using the ***full*** *image? Masking pixels above 1*sigma?*
Using the method ***median****?*
For does who know numpy, This process to compress skippers accepts any statistical function that has as an argument axis (rows:0, columns:1).

https://gev.uchicago.edu/compton/howto_notebooks/analysis/III_howto_PedestalSubtractedProcess.html

Remember: use and independent folder to work (see slide 4)

1. You can re-use the same configuration file than in the previous ex

Configure the JSON file

Note that the input section has been set for this skipper images. If you want to use your own images, take care and set it properly (keywords for the header, axis to compress, overcan regions, ...)

2. Set the process to get the pedestal subtracted image. *Remember to set also the process to compress the image. Be careful with the order of the process in the sequence parameter or in the command line -s!*
3. Set the config values for each process to get what you want

panaSKImg (see slide 3)

4. Launch panaSKImg using the command line -s (instead of using the parameter sequence in the json file). If your internation connection is goot display all plots (--display) and verbose (--verbose), if not just record them (--save-plots). You can also store the compressed image using --save-img
5. Takle a look into the output messages and plots ...

Ex 4.4 **panaSKImg** for skippers

Several skipper images from the Compton setup can be found at

`/sps/hep/damic/school/waders_exercises/compton_run107`

Where you can find

- 209 images with 64 skips: `Image_Am241_Source_107_<id>.fits`
- 1 image with 2000 skips: `Image_Am241_Source_107_skip_1.fits`

Goal: *Use the image with the* ***highest resolution*** *(i.e. the one with 2000 skips, known as* ***image-HR*** *within WADERS)****, fit*** *the* ***dark current*** *using only* ***two single electron peaks****.*

Note: *This method fits the DC by fitting the pixel charge distribution to a set of exponentials convolved with a poisson. It can be done in units of* ***ADU*** *or* ***electrons*** *(change input image to be the one in this units). If you want to set the calibration constant at the same time you must set* ***do_calibration****. If you use this option and your input image is in units of ADU you should give an starting point for the* ***calibration*** *(for instance 5).*

https://gev.uchicago.edu/compton/howto_notebooks/analysis/IV_howto_FitDarkCurrentProcess.html

Remember: use and independent folder to work (see slide 4)

1. You can re-use the same configuration file than in the previous ex

Configure the JSON file

Note that the input section has been set for this skipper images. If you want to use your own images, take care and set it properly (keywords for the header, axis to compress, overcan regions, ...)

2. Set the process to do the fit of the DC. *Remember to include all previous process (compress image, pedestal subtracted, ...). I would suggest you to use the command line -s to set the sequence in case you haven't consideritit! Remember you have –list-processes to get the names of all process*
3. Set the config values for each process to get what you want

panaSKImg (see slide 3)

4. Launch panaSKImg using the command line -s (instead of using the parameter sequence in the json file). If your internation connection is goot display all plots (--display) and verbose (--verbose), if not just record them (--save-plots). You can also store the compressed image using --save-img
5. Takle a look into the output messages and plots ...
6. Re-do it after changing the binning
7. Re-do it after including CalibrationProcess to the sequence, and modifying the config param image to be the image after calibration (i.e. usin now the image in units of e-/pix).

Ex 4.5 **panaSKImg** for skippers

Several skipper images from the Compton setup can be found at

/sps/hep/damic/school/waders_exercises/compton_run107

Where you can find

- 209 images with 64 skips: Image_Am241_Source_107_<id>.fits
- 1 image with 2000 skips: Image_Am241_Source_107_skip_1.fits

Goal: *Use the image with the **highest resolution** (i.e. the one with 2000 skips, known as **image-HR** within WADERS)**,** plot the **signal-to-noise ratio** as a function of the **number** of **skips.***

https://ncastell.web.cern.ch/ncastell/pysimdamicm/howtouse_analysis.html#rnvsnskipsplot

Remember: use and independent folder to work (see slide 4)

1. You can re-use the same configuration file than in the previous ex

Configure the JSON file

Note that the input section has been set for this skipper images. If you want to use your own images, take care and set it properly (keywords for the header, axis to compress, overcan regions, ...)

2. A part from the process to do this plot, set also the one to average the skipper image (be carefull with the order)
3. Set the config values to display only the 1st, 2nd, and 159th skip measurements (among the 2000 we have for each pixel in this skipper)

panaSKImg (see slide 3)

4. Launch panaSKImg using the command line -s (instead of using the parameter sequence in the json file) and display the output by setting both --display and --verbose
5. Takle a look into the output messages and plots ...

Ex 4.6 **panaSKImg** for setups as a DQM

Several skipper images from the Compton setup can be found at

`/sps/hep/damic/school/waders_exercises/compton_run107`

Where you can find

- 209 images with 64 skips: `Image_Am241_Source_107_<id>.fits`
- 1 image with 2000 skips: `Image_Am241_Source_107_skip_1.fits`

A ME (Monitor Element) is a process to create a plot to control your data taken.

Goal: *Run the* ***DQM*** *over the full set of images (over the full run 107) to create the averaged images, and the full set of plots to control the data taken of your setup.*

https://ncastell.web.cern.ch/ncastell/compton_web_site/

Remember: use and independent folder to work (see slide 4)

1. You can re-use the same configuration file than in the previous ex

Configure the JSON file

2. Configure the JSON file only to compress your images using the "mean" function

panaSKImg (see slide 3)

3. *Run* ***panaSKImg*** *as DQM using option* ***--dqm***
    1. *set the directory where data is via* ***--dqm-data-dir***
    2. *point option* ***--image-HR*** *to the name of the image with the highest resolution (available in your run). In this example this image is the one with 2000 skips. You can use just a pattern *skip_1* to identify this image within the previous directory (no need to use the full name). This image is used to get the calibration constant in the first place which is used in the following ME*
    3. *use option --run to set the run number (needed to create the structure of folders for the outputs)*
    4. *Use options* ***--save-img*** *to save all compressed images*
    5. *Set the output directory with option* ***-o*** *and remember do not use the same directory where your input data is (keep things separately and specially when input data is shared with other collaborators!)*

This will create a set of directories with images and plots under runXXX (being XXX the number you use in option –run).