Africalim 2024 Project

November 2024

This project is designed to teach you some essentials developer skills while designing your own toy selfcal pipeline. You should, while leaning heavily on the Documentation, make use of the functions and examples that are available in the codex-africanus repository. Feel free to use any resources whatsoever, including LLM's like ChatGPT, to achieve the goals set out below.

You will need an empty measurement set for these tasks. You can either simulate a small one of your own using the simms tool or use the one available here. Have a look at the conftest in pfb-imaging to see how this MS can be downloaded automatically for use within your test suite (you may find a few other helpful pointers there too).

Working as part of a team is part of the development process. Your best chance of getting through all the steps below is to assign specific tasks to team members and use pull requests (PRs) to incorporate features as they are completed. Ideally PRs should be reviewed by other members of the team and all features should include tests to make sure they work as expected. So, during the course of the week, while you are waiting for jobs to complete, or when we are fixing the inevitable bug, try to implement the following steps:

1. Choose a name for your group project and create a GitHub repository on the africalim organisation called `group-name`. Add a .gitignore file for python and use the MIT licence. It is best if you make the repository public.

2. Make the package installable using pyproject.toml file

3. Create three modules called apps, utils and tests.

4. Use Click to create an app called `hello` which prints "Hello name" to the terminal where name is an input passed in from the command line

5. Modify the app to use `clickify_parameters` from stimela and create a stimela cab so that you can run the app using stimela.

6. Create an app that uses the DFT in codex-africanus populate a measurement set column with the model data corresponding to a source model. You can start with simple source models (e.g. a text file containing the right ascension, declination, flux and spectral index of your sources) but

a stretch goal would be to add support for tigger local sky models. You should pytest to invent a test which verifies that the column is correctly populated.

7. Set up continuous integration (CI) using GitHub actions to only allow merging in PR's once all tests have passed. As you add apps to you repo you will add more tests. It is probably easiest to set up the CI by looking at some examples.

8. Create and app that corrupts model visibilities with a set of random phase only gains and writes it to a column called DATA in the MS. Add an option to corrupt the visibilities with i.i.d. noise of a given standard deviation (default to zero). Design a test for your app and add it to your CI.

9. Create an app that implements a phase only solver to calibrate your data and write the corrected data to a column in the measurement set. Design a test to make sure it works as expected and integrate it with your CI.

10. Create an app to render the corrected data to a dirty image using the wgridder in ducc. Design a test to verify that it produces the expected result by imaging a single point source with a known flux and integrate this into your CI. Also have it write the PSF to a fits file.

11. Write an app which uses single scale clean to deconvolve your image. Write a test to ensure it reconstructs the correct flux for your single component model.

12. Now that you have all the pieces required for a simple selfcal pipeline, use stimela to write a selfcal recipe.