pcherm42/LIGO (github.com)

Overall Structure:

*FoldingAnalysis* Process → *FoldFold* Process →*ChannelVisualizer* Process → *YearCompare* Process →*YearDotshare* Process → *Dotfilter* Process

*FoldingAnalysis*: Generates the original .mat files from the frame files, each of these is a fold over one part of one day across a whole year, these will need to be averaged together.

*FoldFold*: This script goes through and uses the output from *FoldingAnalysis* to fold together each day, then each month, then each year to create yearly summary files and the proper graphs for each day, month, and year.

*ChannelVisualizer:* This takes the *FoldFold* output that was compiled across a day, takes the norm, and plots it on a vertical axis, repeating this process for every day. This creates a time vs date graph with the darkness of each pixel representing the norm of that day. This allows the user to see changes in the behavior of channels across a whole year right at your fingertips.

*YearCompare*: This treats each day's data as a function to be then dotted together in a pseudo-Hilbert space. This creates a "dot product" vs time graph which is helpful for assessing the correlation between an environmental channel and the strain channel after the folding process.

*YearDotshare*: This peers under the hood of *YearCompare*, giving us data on which parts of the data contribute to the dot product. This lets us see which behavior appears in both the environmental channel and the strain channel.

*Dotfilter*: This process first ranks all of the environmental channels based on their dot products, then filters out by euclidean orthogonal projection the highest ranked channel–the channel most polluting the strain channel. You can think of this process as running a comb through hair. The first passthrough gets all the rough edges out, then subsequent runs remove finer knots.

pcherm42/LIGO (github.com)

**How to run from start to finish:**

1. Run createDir.sh <directory_name>
    a. You will need to rename the proper files to their original names to "initialize" the directory
2. Copy over your own version of O3_submitter.py and edit the years and months to the ones you want to submit
    a. Fixed a binary error with the text from process.communicate spitting out utf-8 encoded binary which needs to be re-encoded, annoying issue fixed with one line of code, but lookout now everywhere we used this supposedly 'better' command line reader
3. Add the channels you want to submit to channels.txt
    a. You can always change the days of the month you wish to submit by changing the beginning and end days in the main loop of DAG_maker_and_submitter.py
4. Run your version of O3_submitter.py to submit your jobs to condor.
5. You should now have some uncleaned .mat files. We will need to run some topical scripts to ensure that the foldfold part of the process can run cleanly.
6. The first of these is err_Checker.py, which ensures that no .err files were created by the *FoldingAnalysis*.
7. Then there is O3_Checker.py, which will need to be re-created to an O4 or Ox checker by changing the date information at the beginning of the script.
8. The next is day_day_resubmit_maker.txt which will take a list of days and channels and write them to a DAG to then be submitted to Condor.
9. It is now time to start the *FoldFold* process. This begins with running the gen_foldfold.py script to create a DAG for *FoldFold*.
    a. I found FoldFold very hard to get working consistently from Condor. The script lends itself very well to being first compiled and then run from the command line.
10. There may be some errors which stop the *FoldFold* process, to solve these we have some topical scripts which fix some common errors.
11. List out the faulty channels in the format required for foldfold_resubmitter.py and allow the script to do the proper cleaning and resubmitting.
12. Once the *FoldFold* process is completed, you should have all of the basic plots of all the folded data for each day, month, and year.
13. All of the following scripts spit out files into the matlab directory and will require an `mv *.png desired_path` command to move the files where you want.
14. It is now time to start the *Channelvisulaizer* process. To do this, simply run the proper command is channels.txt to generate the data right from the

command line. No Condor functionality is implemented yet. Move the files into your desired directory

15. Next is the *YearCompare* process. This should run with a command from commands.txt. Make sure to check the data by moving it to a subdirectory of ~/public_html and viewing the data.
16. Next is the *YearDotshare* Process which runs with a command from commands.txt.
17. Next is the *Dotfilter* Process which runs with a simple command from commands.txt. It is recommended to run this with passthrough level 1-3.

**How to make changes to the matfiles.**

If you discover an error, you'll need to make a change. To do this, you make a change to the .m file of the script in which you discover an error. Then you'll need to compile the new file with the correct `mcc <> -a <> …` command available in commands.txt. Don't bother removing things, the mcc command will simply write over the old file. PYTHON DOES NOT NEED TO BE RECOMPILED OR COMPLIED IN THE FIRST PLACE!!

**Audiomaker.m**

This is a cool script which turns a .mat file into an audio file. If your ears deceive you, try using this to find patterns with nature's FFT – your brain!

**Line Simulations**

As part of an effort to understand how each line peak would behave, simulation matlab scripts were created in the Line Simulation folder on the github. pcherm42/LIGO (github.com)

These simulations can be run on your local install of matlab.

**Workflow and Process**

I found it very helpful to get a working version of VisualStudioCode Visual Studio Code - Code Editing. Redefined
I would make a change on VSCode, then put that change onto the LDAS grid, then recompile the matlab. Wash, rinse, and repeat.

**Using the SSH Terminal**

You'll need to learn how to use the UNIX command line. I downloaded a virtual terminal like terminus to then use the ssh feature. On a basic level, you can go into cmd and use `ssh Albert.Einstein@ssh.ligo.org`. You then want to type 2 for CIT then select the machine which is best for whatever job you are running (I Like g. ldas-pcdev6 - Large mem/post procs 1.5T Intel Gold 6154 72 )