Contents

1 Thu **3**
A Bash . . . 3

2 Fri **4**

3 Sat **5**
A Meeting with Julie . . . 5
B Regex . . . 5

4 Sun **6**
A Meeting with Julie . . . 6

5 Mon **7**
A Meeting with Julie . . . 7
B SLURM . . . 7

6 Tue **8**
A Meeting with Julie . . . 8

7 Wed **9**
A Meeting with Julie . . . 9
B Bash . . . 9

8 Thu **10**
A Meeting with Julie . . . 10

9 Fri **11**
A Meeting with Julie . . . 11
B Bash . . . 11

10 Sat **12**
A Meeting with Amy . . . 12
B Python . . . 12
(a) XFintoARA.py . . . 12

12 Mon **13**
A Python . . . 13
(a) XFintoARA.py . . . 13
(b) General . . . 13

13 Tue **14**
A Meeting with Julie . . . 14
B Bash . . . 14

14 Wed **15**
A Meeting with Julie 15
B AraSim Job Submission Script 15
(a) AraSimCall_Array.sh 15
(b) AraSimCall_Array.sh: my attempt 16

15 Thu **17**
A AraSim Job Submission Script 17
(a) AraSimCall_Array.sh: my attempt 17
B Meeting with Julie 17

16 Fri **18**
A Birefringence Resources from Justin 18
B Meeting with Julie 18
(a) Notes from today's meeting 18
(b) Second pass of the Loop continued 18
C Bash 19

17 Sat **20**
A Meeting with Julie 20
(a) Part_D2_Array.sh Second Pass 20

18 Sun **22**
A Meeting with Julie 22
(a) Part_D2_Array.sh Second Pass Continued 22

19 Mon **23**
A Bash 23
B Meeting with Julie 23
(a) Part_D2_Array.sh Second Pass Continued 23

20 Tue **24**
A Meeting with Julie 24
(a) Part_E_Asym.sh second pass 24

21 Wed **25**
A Meeing with Julie 25
(a) Part_E_Asym.sh second pass continued 25

1 Thu

A Bash

a) A little bit of Bash each day keeps the doctor away. In the main loop, around line 140, we have

```
read -p "Starting generation ${gen} at location ${state}.\
          Press any key to continue... " -n1 -s
```

Option –`p <prompt>` outputs the `prompt` string before reading user input.
Option –`n <number>` returns after reading the specified `number` of chars.
Option –`s` does not echo the user's input.

2 Fri

On a flight home. Didn't do much except started going over Part B1 again

3 Sat

A Meeting with Julie

a) The meeting this Wednesday was cancelled. In the meantime, Julie suggests reading the Appendix and her candidacy paper. I have largely finished reading the relevant part of the Appendix, so the rest of this week will most likely be spent reading the candidacy paper and checking out the `Loop Parts` in more detail.

b) Part B1 second pass: will also check out the job submission script this time around (final line of `Part_B_GPU_job_1.sh`, but still not going to look into the xmacros).

B Regex

a) Started reading about regular expression, although perhaps not directly relevant to anything.

4 Sun

A Meeting with Julie

a) Currently a bit confused by the job submission script `GPU_XF_job.sh`

b) I think this is because I am not familiar enough with the file structure yet.

c) Also, I am not entirely sure what the `individual number` is for in this script.

5 Mon

A Meeting with Julie

a) Still on the job submission script of `Part B1`. I am confused about when exactly are the `indiv_parent_dir` created (the ones in $`XFproj/Simulations`)

B SLURM

Filename replacement are as follows:

a) `%x` - Job name, which can be given as `--job-name=<name>`

b) `%a` - Job array ID (index) number

c) `%A` - Job Array's master job allocation number

d) More options here: `https://slurm.schedmd.com/sbatch.html`

Note that the following are not the same:

a) $`SLURM_ARRAY_JOB_ID`: this is the `%A` from above

b) $`SLURM_ARRAY_TASK_ID`: this is the index (`%a` from above)

6 Tue

A Meeting with Julie

a) Started reading Julie's candidacy paper.

7 Wed

A Meeting with Julie

a) Read everything aside from Chapter 5.

b) **TODO:** Go over Chapter 3 again before reading chapter 5.

c) Ask Julie about Figure 6,7 8 and 9 (a discussion on all of chapter 3.1.2 if possible)

d) **Zenith Angle:** The angle at which the muon traveled into the IceCube detector, with respect to the vertical. Source: https://user-web.icecube.wisc.edu/~krosenau/index.html

B Bash

a) A backtick is not a quotation mark. Instance: `Part_B_GPU_job1.sh` line 37

```
for i in `seq 1 $NPOP`
```

Everything we type inside the backticks is executed before the main command (such as `chown`), and the output of the backticked-command is then read by the main command

8 Thu

A Meeting with Julie

a) Regarding Part A of the June 5th entry, from Alex:

> those are created by XF. It's like a backend thing, we never make them ourselves but by virtue of simulating antennas they get made by XF as the location to store the antenna simulation data

In other words, we do not ourselves `mkdir` the $`indiv_dir_parent`.

b) Meeting with Julie today moved to Monday next week.

c) ~~It seems unlikely to be a bug given how long this script has been used, but I am getting an error from the for loop through all frequencies in freqlist. Will ask Alex about it.~~ I'm an indiot. I need to initialize $`GeoFactor` first.

d) Finished `Loop_Parts/Part_B1` second pass, moving on to `Part_B2`. Adding line breaks in the process; **Be careful with the trailing whitespace!** *Mayhap it's better to just leave them as single long lines?*

e) **TODO:** sometime in the future I'll need to figure out what `simulation_PEC.xmacro` and `output.xmacro` do to really understand what `Part_B1` and `Part_B2` are about.

9 Fri

A Meeting with Julie

a) ~~Probably trivial, but it seems like the uan files are already being moved to the correct directory at the end of Part_B2, so I am not sure why we do it again at the beginning of Part_C.~~

b) One of the `mv` command might be extra; will check with Alex.

c) `Part_C` second pass: digging into the python codes; should be able to finish by tomorrow.

B Bash

a) The dollar sign works inside the double quotes so there is no need for string concatenation in the example below:

```
$a=2;$b=2
echo "a * b = $(($a*$b))"
```

The above outputs `a * b = 4`. Note that the $((...)) part is for arithmetics.

b) At some point I should try to figure out exactly what double quotes are for in shell scripts; they seem to be more than just strings?

10 Sat

A Meeting with Amy

a) Get in touch with Nicholas to see what he's up to.

b) Go to Monday's collaboration meeting this week if possible to see if there's a project for me.

B Python

(a) XFintoARA.py

a) instead of using the `%`'s for string interpolation, we could use `f−strings`.

b) inside the curly braces we can call variables, for instance {`g.WorkingDir`} in line 66:

```
uanName = f'{g.WorkingDir}/.../{g.gen}_{indiv}_{freqNum}.uan'
```

c) The `g` above is from line 102: `g = parser.parse_args()`

d) `Line 73`: `mat = [["0" for x in range(n)] for y in range(m)]` is simply python's way to do `mat=zeros(m,n)` in Matlab. (`List Comprehension`)

12 Mon

A Python

(a) XFintoARA.py

The following pertains `line 81 \& 82`

a) `line 81` first turns the third entry in the list `lineList` into a float.

b) `line 82` contains the following: `"%.2f" % 10` which turns `10` into `10.00`. This is NOT modulo operation; this is probably more like string interpolation.

c) Mostly finished reading this python script. Moving on to `Part D` tomorrow.

(b) General

a) To access the `help` message of `argparse`'s `add_argument` function, one can run the command `python3 <filename>.py -h` at the terminal

b) For more info on the `argparse` module one can look through the documentation of `argparse` on https://docs.python.org/3/library/argparse.html and the tutorial at https://towardsdatascience.com/a-simple-guide-to-command-line-arguments-with-argparse-6824c30ab1c3

c) ~~Reviewed file opening (with open, etc)~~

d) mode `w` and `w+`: `w` is write whereas `w+` is read and write

e) apparently with python3, when using `os.chmod` one needs to add `0o` (zero-oh) in front of `777` to grant all read-write-execute, etc. For instance:
`os.chmod("<filename>", 0o770)`
This is because

> In unix conventions, written numbers are assumed to be decimal unless they are prefixed with a `0x` (or `0X`) in which case they are hexadecimal

(https://stackoverflow.com/questions/32729309/what-is-the-purpose-of-the-octal-digit-0-permission) and according to the error message we "use an 0o prefix for octal integers".

13 Tue

A Meeting with Julie

a) Meeting with Julie today didn't happen.

b) Around line 24 of `Part_D1_Array.sh` the comment says to "make a directory to hold the AraSim output and error files *for each generation*" but it seems like we are only making the directory for the zeroth gen?

B Bash

a) option `-e` of `sed` allows for multiple commands at once; for instance,
```
sed -e "s/world/universe/" -e "s/hello/goodbye/" ./temp > ./newtemp
```
first replaces the word "world" in `temp` with "universe" and then "hello" with "goodbye" and then pipe these changes to a new file `newtemp`.

14 Wed

A Meeting with Julie

a) Continuing `Part_D1` second pass.

b) Looking into the job submission script `Batch_Jobs/AraSimCall_Array.sh`

c) Sort of annoying, but it seems like SLURM directives simply cannot be broken into multiple lines with backslash, so I'll just leave them.

d) ~~what do num and seed in AraSimCall_Array.sh refer to?~~ `Seed` is defined at the very beginning of the main loop and passed to the scripts along the way. See page 155 (Appendix A) of Julie's dissertation.

e) Note: haven't looked into `setup.txt` yet

B AraSim Job Submission Script

(a) AraSimCall_Array.sh

```
#!/bin/bash
## This job is designed to be submitted by an array batch submission
## Here's the command:
## sbatch --array=1-NPOP*SEEDS%max --export=ALL,(variables) AraSimC...
#SBATCH -A PAS1960
#SBATCH -t 18:00:00
#SBATCH -N 1
#SBATCH -n 8
#SBATCH --output=/fs/ess/PAS1960/BiconeEvolutionOSC/BiconeEvolution/cur...
#SBATCH --error=/fs/ess/PAS1960/BiconeEvolutionOSC/BiconeEvolution/cur...

source /fs/ess/PAS1960/BiconeEvolutionOSC/new_root/new_root_setup.sh
cd $AraSimDir
num=$(($((${SLURM_ARRAY_TASK_ID}-1))/${Seeds}+1))
seed=$(($((${SLURM_ARRAY_TASK_ID}-1))%${Seeds}+1))
echo a_${num}_${seed}.txt

chmod -R 777 $AraSimDir/outputs/
./AraSim setup.txt ${SLURM_ARRAY_TASK_ID} $TMPDIR a_${num}.txt > \
 $TMPDIR/AraOut_${gen}_${num}_${seed}.txt
cd $TMPDIR
echo "Let's see what's in TMPDIR:"
ls -alrt

echo $gen > $TMPDIR/${num}_${seed}.txt
echo $num >> $TMPDIR/${num}_${seed}.txt
echo $seed >> $TMPDIR/${num}_${seed}.txt
```

```
mv AraOut.setup.txt.run${SLURM_ARRAY_TASK_ID}.root\
 $WorkingDir/Antenna_Performance_Metric/AraOut_${gen}_${num}_${seed}.root
mv AraOut_${gen}_${num}_${seed}.txt $WorkingDir/Antenna_Performance_Metric/
mv ${num}_${seed}.txt $WorkingDir/Run_Outputs/$RunName/AraSimFlags

## This part appears unnecessary now
: << 'END'
...
...
END
```

(b) AraSimCall_Array.sh: my attempt

This is an attempt to understand `AraSimCall_Array.sh`. My own edited version of the job submission script as attached at the end of this section. Run-on lines are mainly just SLURM directives which are unimportant here, so they are ignored here in the log.

a) This is for `line 15`. There's really no need to start $`SLURM_ARRAY_TASK_ID` at 1, but if that's what we've been doing then I'll leave it for consistency. That is, we could have submitted the job array analogous to

   ```
   sbatch --array=0-8%4 foo.sh
   ```

   in which case the first index would have been 0.

15 Thu

A AraSim Job Submission Script

(a) AraSimCall_Array.sh: my attempt

a) `Seeds` is the number of AraSim jobs of an individual. So `num` on `line 15` is essentially just (ignoring the pesky index issue) `index` divided by `Seeds`.

b) As an example, consider a task 33 in the array of tasks. Suppose `Seeds` is 8; that is, for each antenna, we do 8 AraSim runs. Since $33 - 1 = 32$ divided by 8 is 4, and then we add one to get 5, `num` in this case is 5. In other words, task 33 is a task for antenna **num**ber 5.

c) Similarly, right below `num`, `seed` refers to the "seed index" for that particular antenna. Continuing with the example above, task 33 will be seed number 1 of antenna 5 (start counting from 1, as usual).

d) Pretty sure `line 16`: `echo a_${num}_${seed}.txt` is unnecessary, and the file name being `echo`ed is probably a typo as well.

e) Line 18 is also extra? Seems like we are not using this directory anymore?

f) `Line 19` is likely the line that says "run AraSim" (with the appropriate setup and parameters) and then redirecting the output of the run to the (local) scratch space of the cluster, ie. $`TMPDIR`. (see osc documentation regarding parallel and local scratch space. Basically, `TMPDIR` is the fastest.)

g) `Line 26` is like another "SaveState" file.

h) `Line 30` to `33` move the AraSim output files from the scratch space back to the directories under `GE60`.

B Meeting with Julie

a) Meeting with Julie today rescheduled to same time tomorrow.

b) Finished going through the job submission script; back to `Part_D1`

c) Skipping the `DEBUG_MODE` of `Part_D1` for now.

d) Changed `line 98 and 99` of `Part_D1` to use $`WorkingDir` to shorten the lines.

16 Fri

A Birefringence Resources from Justin

Putting these here because apparently Slack hides messages older than 90 days. Boo.

a) Amy's paper: https://arxiv.org/abs/2110.09015

b) Paper by a collaboration member who studies the birefringence in ice: https://arxiv.org/abs/1910.01471

c) "[A] decent paper on the theory, but it's a little math heavy":
https://link.springer.com/article/10.1007/s00371-011-0619-2
https://arxiv.org/abs/2110.09015
https://arxiv.org/abs/1910.01471

B Meeting with Julie

(a) Notes from today's meeting

a) hpol – I will probably be using AraSim as is, without having to modify its code.

b) Dylan built the PUEO software by modifying PAEA; Julie had a to-do list of instruction for him to do it which she will modify and send to me.

c) Birefringence – Julie said that it sounds like what Amy wants is for hpol to be optimized for birefringence (?), but I don't need to worry about learning all about birefringence for now, since I will be mainly just working on scripts of the Loop.

d) Charged current interaction

$$\overline{\nu}_l + N \to l^{\pm} + X$$

The lepton people see is usually muon because it is much more stable than tau (longer lifetime). We don't see electrons either because they just get reabsorbed immediately into the ice atoms. (Consequently, muons produce tracks whereas the other two create spheres)

e) Julie recommends Dick, Chris Hirata, and Antonio Boveia for candidacy committee.

(b) Second pass of the Loop continued

a) Continuing with `Part_D1_Array.sh`

b) As reported a few weeks ago on Slack, `line 116` will never be executed. Alex set the impossible condition to keep the code, but I'll just comment it out.

c) finished `Part_D1`, moving on to `Part_D2`.

C Bash

a) be sure not to include whitespace when assigning values to variables in bash. For instance, `a=2` is correct but not `a = 3`.

b) Example usage of `expr`: `totPop=$( expr $NPOP \* $Seeds )` *Note the whitespace! It matters here whether or not there are spaces around the multiplication operator.*

c) But what is the difference between `expr` and simply using double parentheses? For instanace `totPop=$(( $NPOP * $Seeds ))` (in this case it seems like bash doesn't care as much about the whitespace around $*$)?

17 Sat

A Meeting with Julie

(a) Part_D2_Array.sh Second Pass

a) Might be able to shorten `Line 22`:

```
nFiles=$(ls -1 --file-type ../AraSimConfirmed | grep -v '/$' | wc -l)
```

using just `ls | wc –l`. Will test this tomorrow.

b) Option `1`(one) of `ls` "[forces] output to be one entry per line"

c) `-file-typ` makes it so that all the directories end with a `/`, and soft links end with a `@`, etc. https://stackoverflow.com/questions/51952975/what-is-the-purpose-of-file-type-in-ls-command Files don't have anything attached.

d) Thus, as an example, inside $`WorkingDir/Run_Outputs/2023_02_20_Symmetric_Run`, if we issue `ls` we get

```
2023_02_20_Symmetric_Run.xf   Fitness_Scores_RG.png   uan_files
Antenna_Images                FScorePlot2D.png        Veffectives_RG.png
AraOut                        Gain_Plots              Veff_plot.png
AraSimConfirmed               Generation_Data         Violin_Plot.png
AraSim_Errors                 GPUFlags                XF_Errors
AraSimFlags                   Root_Files              XFGPUOutputs
AraSim_Outputs                runDate.txt             XF_Outputs
Evolution_Plots               run_details.txt
```

and with `ls –1` we have

```
2023_02_20_Symmetric_Run.xf
Antenna_Images
AraOut
AraSimConfirmed
AraSim_Errors
AraSimFlags
AraSim_Outputs
Evolution_Plots
Fitness_Scores_RG.png
FScorePlot2D.png
Gain_Plots
Generation_Data
GPUFlags
Root_Files
runDate.txt
```

```
run_details.txt
uan_files
Veffectives_RG.png
Veff_plot.png
Violin_Plot.png
XF_Errors
XFGPUOutputs
XF_Outputs
```

Lastly, with `ls -1 --file-type` (on OSC, not on Mac), we get

```
2023_02_20_Symmetric_Run.xf/
Antenna_Images/
AraOut/
AraSimConfirmed/
AraSim_Errors/
AraSimFlags/
AraSim_Outputs/
Evolution_Plots/
Fitness_Scores_RG.png
FScorePlot2D.png
Gain_Plots/
Generation_Data/
GPUFlags/
Root_Files/
runDate.txt
run_details.txt
uan_files/
Veffectives_RG.png
Veff_plot.png
Violin_Plot.png
XF_Errors/
XFGPUOutputs/
XF_Outputs/
```

e) option `v` of `grep` is "invert-match", which acts like a `not`-gate. It selectes all entries that do not match. In this case we are trying to match all entries that has a forward slash `/` right before the end-of-line character `$`; that is, we are trying to match all the directories. And then, `v` makes sure that we select everything that is not a directory, ie. files. Finally, we pipe it to `wc -l` to count as usual.

f) I'll check with Alex to see if we really need to be this careful, because in `Part_B_GPU_job2_asym_array.sh line 56` it seems like we decided to simply use `ls | wc -l`, which is much cleaner.

18 Sun

A Meeting with Julie

(a) Part_D2_Array.sh Second Pass Continued

a) Regarding the `for` loop around `line 28`, consider the following script:

```
#!/bin/bash
cd ~/Desktop/temp

for file in *
do
  echo $file
done
```

If there is *no* file inside `temp/`, then the output would be $*$. This is what the comment around `line 32` is talking about.

19 Mon

A Bash

a) We can break a line inside double quotes. For example, the following is legal

```
#!/bin/bash
pat="/users/PAS2137/unmovingcastle\
/temp"
out_name=$pat/%x.out
err_name=$pat/%x.error
sbatch --job-name=whatever --output=$out_name --error=$err_name a.sh
```

b) Normally we don't even need the backslash if we are just `echo`ing the stuff that is inside the double quotes. But if we are going to access the path later with `$pat` then it appears that the backslash *is* necessary.

c) `wait` waits for a process to finish; `sleep` sleeps for a certain amount of seconds.

B Meeting with Julie

(a) Part_D2_Array.sh Second Pass Continued

a) Used `$WorkingDir` to shorten `line 60` & `61`

b) The final `if` block is effectively a comment, so I'll comment it out.

c) Finsihed `Part_D2_Array.sh`

20 Tue

A Meeting with Julie

(a) Part_E_Asym.sh second pass

a) The curly braces around `$10`, `$11` and `$12` are *necessary*.

21 Wed

A Meeing with Julie

(a) Part_E_Asym.sh second pass continued

a) To understand the `for` block around `line 37`, consider the following example

```
for i in ‘seq 1 4‘
do
  InputFiles="${InputFiles}out${i}.txt "
done
echo $InputFiles
```

b) The output is
`out1.txt out2.txt out3.txt out4.txt`

c) **TODO:** haven't looked into `fitnessFunction_ARA.cpp` yet.