GENETIS To Do: With Detailed Instructions!

Julie Cade Alex P. Alex M. Mitchell

(1) Add ARA Constants to plots:

- [ ] Make Software that plots "θ versus Generation"

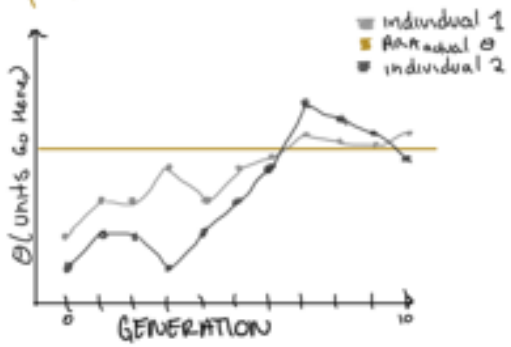
- First, reference "LR-plot.py" to see how to write this code.

Step (1) instead of reading in L R, we want to read in θ from the exact same file! ie generation DNA files.

Step (2) Add in proper labels for the plots
  - → Units
  - → axis labels
  - → individuals

Step (3) Add a constant, θ, that is equal to the value of ARA's current actual θ value Plot that on top of the plot every time

- The plot should look like:

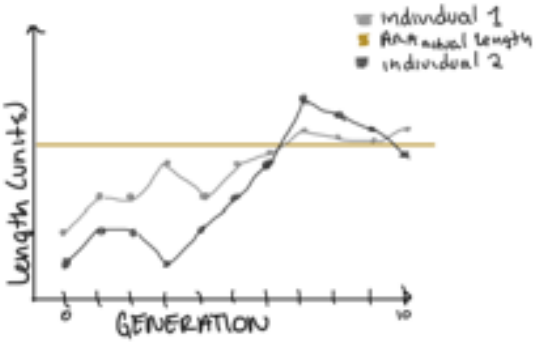


☐ Fix Current L and R plots

Step ① Correct Both plots to have labeled units for each Axis, and a key. This L and R code is "LRplot.py".

- Google This!

Step ② Go into "LRplot.py" and add current Length and radius for ARA_{actual} Similarly to the instructions above.



... and Similarly shown for R.

Note: this should plot for ${gen} individuals. The example I've plotted is assuming gen = 2.10 - there are only 2 individuals

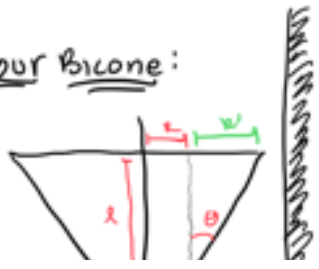
Confirm ARA_{actual} Bicone Values via:

1. Ask Amy?
2. Look at Brian Clark's thesis.
3. Look at papers for diagrams.

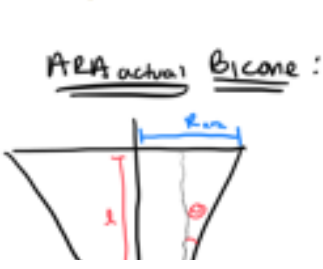
Step ③ Fix <u>our</u> "R" values.

I do believe we may be plotting the incorrect "R". Here is what I think is happening:

<u>our Bicone</u>:



<u>ARA_{actual} Bicone</u>:



Thus, $R_{ara} = R + R'$.

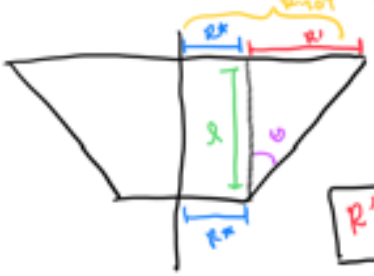
Similarly, To do this:

Confirm ARA_{actual} Bicone Values via:

1. Ask Amy?
2. Look at Brian Clark's thesis.
3. Look at papers for diagrams.

Step ④ If we are wrong, we must change the "R vs Gen" part in "LRplot.py" code so that we:

1. Read in R, L, θ for each generation.
   - See how this is already done for reading in R, and repeat for L and θ.
2. Do trig to calculate the same "R" as ARA_{actual} Bicone defines. [code this calculation into "LRplot.py"].



$\tan\theta = \frac{R'}{l}$

$l \tan\theta = R'$

$R' + R^* = R_{TOT}$

3. Store the newly calculated "R" as an Array, and plot That each time

4. Don't forget to add the ARA_{actual} Bicone Radius as a constant line.

☐ **Find ARA_{actual} Bicone Veff Given the Energy of the Current Run.**

Goal: Put ARA_{actual} Bicone through AraSim to see Veff at the same energy as the individuals.

Step ① Add a line in gen 0 of the Bash Script That Submits another job for the ARA actual Bicone using the input file below.

Explanation: There's a current file that exists for ARA actual Bicone that has the gain patterns for the actual antenna dimensions — ie AraSim input that XF creates.

This file is properly formatted and ready to go! When we create the line in our Bash Script (XF_loop.sh) that would Submit an extra job, This file would be our input. It exists here:

BiconeEvolutionOSG/AraSim/Ara_Bicone_bin_outputs.txt

Thus, we don't need to run the actual ARA actual Bicone parameters through XF To get a Veff at the same Energy of our current Run. We just need To use this already created file above and run it through AraSim.

Note: We only need to get Veff at our Run's Energy once! Once we have it, it can be stored and reused when we replot every generation, since ARA actual Bicone isn't evolving & is just a Reference point for Veff.

Step ② Correct "wait" command for AraSim job Submission.

- "wait" command in Bash Script on the 1st iteration [because we only Run ARA actual Bicone on generation 0] must be edited to wait for

  $NPop+1 (instead of $NPop)

Step ③ Move ARA actual Bicone AraSim outputs to the {$RunName} directory.

- edit in Bash Script in AraSim Section

note:
- Make Sure that this location is either:

1) Moved after we plot "Veff vs gen" and calculate the fitness Function.

OR

2) We account for this directory/location move when we Plot "Veff vs gen" and calculate the fitness Score.

☐ Add Plot for "Veff vs. Generation" With Ara's Veff.

Step① Start by following the way "LRplot.py" is made. -ie copy it!

1. However, in this case, we need to take files:

fs/project/PAS0654/BiconeEvolutionOSC/BiconeEvolution/current_antenna_evo_build/XF_Loop/Evolutionary_Loop/Run_Outputs/$RunName/AraOut${gen}_${I}.txt

ie the AraSim outputs for each individual in All Generations (gen 0 - ${gens}) - including the one we create in generation 0 for Ara for ARA actual Bicone and plot Veff from there.

Step② Read in Veff values (and Veff error).

The error data is labeled "and Veff (water eq.)" at the bottom of these AraSim output files we are already reading in for Veff.

1. note: to know how to read in Veff and errors you can look at how we are reading in that *Exact* line from the AraSim output files in on the fitnessfunction_Ara.cpp code existing in the

Antenna_Performance_Metric directory, since we are doing it there

2. Since we're plotting veff vs. generation, we will either have to read in all of the files AraOut${gen}_${i}.txt (up to how many ${gen} we are in) every time -- (including the ARAcadia Bicone one --) or we can append them all to one .txt file and read that in. It's your choice.

3. Make sure you Nicely label this like the plots above! 😊

② Fix ARA Hole constraint:

note: we've made a mistake when implementing the radius constraint.

What we did: In fitnessFunction_ARA.cpp we added...

If $R_{individual} > R_{Hole}$

Then

$$fitnessScore = fSx\left(e^{-n\left[\frac{R_x - R_{hole}}{R_x}\right]^2}\right)$$

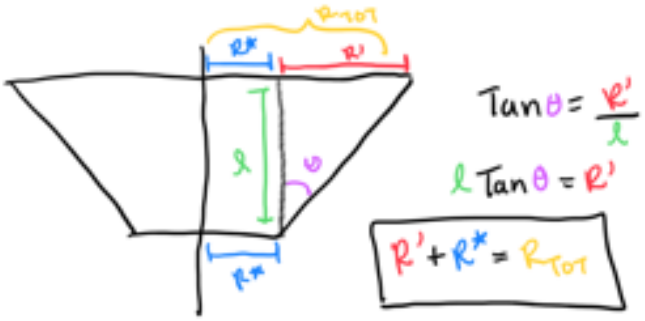
else

fitnessScore = fS.

However, $R_{individual}$ isn't what we thought it was.

- [ ] Edit fitnessFunction_ARA.cpp to find R.

Step① Add the following Math to this code after we've read in our R* value from generationDNA.CSV.



1) Read in L, G from generationDNA.CSV.
2) Use them to find R'.
3) Add already read in R^* to R' and get R_{TOT}.

Step② Save this new R_{TOT} as an Array and Make:

if $R_{TOT} > R_{Hole}$

then

$$FitnessScore = FS \times \left[e^{-\pi \left[\frac{R_{TOT} - R_H}{R_{TOT}} \right]^2} \right]$$

else ....

$FitnessScore = FS$

③ other Random Tasks:

☐ **Make Energy a variable in Setup.txt**

Step① Exactly how we've made "nnu" in Setup.txt a variable, we should be able to Make Energy a variable. Either:

① Google if you can use the "sed" command in Bash Script is how

We're currently using it] to replace 2 different words with 2 other different words somehow.

OR

(2) write from Setup_dummy.txt to a new file called Something like

"Setup_dummy_2.txt"

To replace variable "Innu". Then, use the "sed" command to replace Energy in "Setup_dummy_2.txt" file and write that to:

Setup.txt

-- which is what AraSim reads! :)

If you choose this/ figure it's the best option, Do the following:

1) cp Setup.txt Setup_dummy_2.txt

2) Change "Energy" at the top of Setup_dummy_2.txt to be equal to a random (not ever used anywhere else in the document) string of text such as:

Energy = VarEnergy

3) Change current "sed" lines in bash script (in Gen 0 part and looping part) To say:

sed 's/new-nnn/$NNT Setup_dummy.txt > Setup_dummy_2.txt

not Setup.txt

4) add new line just Below it (in gen 0 & looping part) that says:

sed 's/VarEnergy/$Energy' Setup_dummy_2.txt > Setup.txt

5) at the top of the Bash Script in the variables Section, add

Energy = 17 # This variable is the exponent energy. ie 17 is 10^17.

- [ ] Save CAD Models in XF
- [ ] STOP plots from popping up during Run
- [ ] Add Cade's "Save state" idea to the loop.
  - This Saves where we are in the Loop So we can Continue a Run!
- [ ] Push to github
  - we need to git ignore Big files like .Rad files and everything in $RunName.
  - we have a .gitignore made but we don't understand why its not working.
- [ ] figure out why fitnessfunction_Ara.cpp