

# GENETIS Bicone Evolution User Manual

Julie Rolla, Max Clowdus, Suren Gourapura, Tom Sinha, Cade Sbrocco

February 22, 2019

This manual was created 1/29/19. Note that the codes involved in this project have the potential to change at any point. We will do as much as possible to keep this up to date; however, make sure you are checking our "Where Our Info Exists" section (section 1.1) – specifically our ELOG– for any comments, concerns, or updates along the way. If you see any issues with this manual, please email [JulieRolla@gmail.com](mailto:JulieRolla@gmail.com).

Others previously involved in this project with OSU: David Liu, Adam Blenk, Hannah Hassan, Jordan Potter.

# Contents

<b>1</b>	<b>Intro: Goal of Code</b>	<b>1</b>
1.1	Where our Info Exists . . . . .	1
<b>2</b>	<b>Setup</b>	<b>2</b>
2.1	Getting a CalPoly Account . . . . .	2
2.2	How to Login into Nutau . . . . .	3
2.3	Navigation of Nutau . . . . .	4
2.4	Running on Personal Machine . . . . .	4
<b>3</b>	<b>The Software</b>	<b>7</b>
3.1	An Introduction to the Loop . . . . .	7
3.1.1	Bash Script . . . . .	9
3.1.2	Genetic Algorithm . . . . .	10
3.1.3	XF Simulation Software . . . . .	10
3.1.4	XF Output Conversion Code . . . . .	11
3.1.5	ARASIM Execution . . . . .	11
3.1.6	Fitness Score Generation . . . . .	12
3.1.7	Plotting . . . . .	12
3.1.8	Editing Variables . . . . .	13
<b>4</b>	<b>Running the Loop</b>	<b>13</b>
4.0.1	Initializing the loop . . . . .	14
4.0.2	Navigating XFtdtd . . . . .	15

# 1 Intro: Goal of Code

There is a future, and a history of this code; both are relevant. Originally in April of 2018, this group presented at April APS displaying (1) the success of the evolution of a quarter wave-length dipole antenna (essentially used as a sanity check to show our software was properly evolving), and (2) that we can evolve physical gain pattern to maximize or minimize gain in a specific direction. The past data, results, and information can be found in the places stated in section 1.1. Note that presentation of these results can be found in the Dropbox – and the old code for the dipole evolution can be found on the Nutau computer in Documents/OSULoop, or at the following GitHub account: <https://github.com/hchasan/XF-Scripts>.

The contents of this manual are focused mainly on the bicone evolution. Our future goal is comprised of three main things (1) show we can evolve more complicated antennas – ie the bicone, and (2) to at some point evolve an antenna rouge – similar to the paper clip ST-5 antenna by NASA, and finally, (3) evolve a gain patter and use that gain pattern as a ”learning data set” for our antenna to evolve. This means that eventually we would be able to create antenna geometry based on it evolving toward a specific set of gain patters. These last two goals are yet to come.

For now, the purpose of the bicone evolution package is to genetically evolve the radius, length, and angle parameters of bicone antennas to maximize neutrino detection. The code uses a roulette algorithm that will be discussed further in section 3.1.1.

## 1.1 Where our Info Exists

As a user note, we primarily prefer to use the ELOG for updates, and Github for our up-to-date software. Additionally, we have a Slack link where the group communicates. However, past information can be found in a few other places. Below are the links where info can be found.

Slack: <https://gpantennas.slack.com/messages>

ELOG: <http://radiorm.physics.ohio-state.edu/elog/GENETIS/>

Github link: <https://github.com/mclowdus/BiconeEvolution>

DropBox link: [https://www.dropbox.com/home/GP\\_Antennas](https://www.dropbox.com/home/GP_Antennas)

Old GitHub for the evolution of the dipole:

<https://github.com/hchasan/XF-Scripts>

## 2 Setup

### 2.1 Getting a CalPoly Account

All of the software for this project is currently being run on a machine at Cal Poly owned by collaborator Stephanie Wissel. In order to gain access, you must have an account on the Cal Poly network. To get an account you must email Amy Connolly at [connolly@physics.osu.edu](mailto:connolly@physics.osu.edu) asking for access to Nutau. The email should contain the following information:

First and last name

Date of birth

Email address you can be reached at

Phone number you can be reached at

Once you receive word that your account has been created, you'll need to call the Cal Poly Service Desk at 805-756-7000. They will ask you for your user name – which you should have received in the email confirming that your account creation. Tell them that you are

an affiliated user and that you need to set your password. You will be asked some basic information (last 4 of SSN, date of birth, etc). The Service Desk will give you a temporary password that will expire 3 days after issued. You should then log in to my.calpoly.edu and set your permanent password via the Personal Info tab.

## **2.2 How to Login into Nutau**

Once you have an account, you can now remote log in. We don't use a VPN client, because it takes up too much computing power from Cal Poly. Instead, we SSH in on a terminal. Note that this requires you to be able to navigate only using Linux commands. To log into Cal Poly, first open terminal (Mac, or Linux).

If you are using a PC, you will need to setup your computer to be able to SSH in. This means you will have to setup something like Bash on Ubuntu on Windows (for Windows 10) or Cygwin (all Windows versions). Here's a link for Bash on Ubuntu on Windows download: <https://www.windowcentral.com/how-install-bash-shell-command-line-windows-10>

As a side note, Bash on Ubuntu on Windows is only available for Windows 10. If you are using a machine with an older version of Windows, you will need to use Cygwin. If possible, it is ideal to have a newer version of Windows and to avoid using Cygwin. Bash on Ubuntu on Windows is a little nicer to use than Cygwin, because Cygwin doesn't let you access the home directory on that machine— only files in its home directory. It instead creates a home directory for Cygwin. All software must then be saved in C:/cygwin64/home/connollylab instead of C:/home/user. Again, if you have a Mac or run Linux none of this will be an issue, and you can just proceed with terminal.

If you are on a Mac or boot Linux, you should already be capable of this without installing any software. Now, open terminal (already on a Mac or Linux), Cygwin (for all Windows versions), or Bash on Ubuntu on Windows (only for Windows 10). First we must login to the Cal Poly general network. After we are in the Cal Poly network, we must log in

to a specific machine on the Cal Poly campus (we use Nutau, which is Stephanie's machine).

To do so, type the following commands:

```
ssh -Y (username)@dirac1.calpoly.edu
```

Now, it will ask you to enter your password you setup with your account.

```
ssh -Y radio@nutau.physics.calpoly.edu
```

Use the general password: iceRadio

You should now be logged into the Cal Poly machine, Nutau. Here is where all of our work is being done.

### **2.3 Navigation of Nutau**

The Bicone Evolution software can be found on Nutau at:

`~/Documents/BiconeEvolution/current_antenna_evo_build/XF_Loop`. Note that to navigate, you will need to know basic Linux commands (see something like this: <https://www.pcsuggest.com/basic-linux-commands/>). More specifically, you must `cd` into this location where the loop exists.

### **2.4 Running on Personal Machine**

In order to run on any machine, first you must have the following programs installed:

Ubuntu

XFdtd

AraSim (and all AraSim prerequisites)

Installation:

1. The current and up-to-date code package for the loop can be found at

<https://github.com/mclowdus/BiconeEvolution>. If you are choosing to install the package on a machine it is not currently on, please use this link.

2. Path names in all of the software must be edited to correctly correspond to the proper locations of code. Note that our files should be set up according to figure 1. The following are the lists of scripts which contain directories that need to be edited:

XF\_Loop.sh

XFintoARA.py

output.xmacro

simulationPECmacroskeleton.txt

simulation\_PEC.xmacro

Note that for the XF\_Loop.sh file (located in XF\_Loop/Evolutionary\_Loop) we must also change the directories for variables XFexec and AraSimExec; these must be edited to directory locations where the local installation are –ie change these to the file path where XFdtl and AraSim are installed on your machine.

3. Once directories have been edited, compile the .cpp file located in the following locations:

(A) XF\_Loop/Evolutionary\_Loop

(B) XF\_Loop/Evolutionary\_Loop/Antenna\_Performance\_Metric

All .cpp files must be compiled using C++11 libraries. The compiled files should be gens-Data.exe, and roulette\_algorithm.exe (in the main directory), and fitnessFunction.exe. Please note that either fitnessFunction\_ARA.cpp or fitnessFunction\_XF.cpp should be compiled to fitnessFunction.exe, not both. The fitness function appended with \_ARA is in the case that you are running loops using AraSim; the function appended \_XF should only be used if AraSim is not being used.

5. The antenna loop should now be correctly installed. Customization can now be done a variety of options are available, but require changes to multiple programs. Remember to recompile any .cpp files that are updated.

A. Change population sizes. In order to do this, the variable NPOP must be adjusted across each of these files roulette\_algorithm.cpp, fitnessFunction.cpp, simulation-PECmacroskeleton.txt, XFintoARA.py, and XF\_Loop.sh.

B. Change number of generations to run for. This can be changed in XF\_Loop.sh, by editing the variable TotalGens in the header.

C. Change initial seeding or mutability. This can be changed simply by editing roulette\_algorithm.cpp. Simply adjust the mutability factors or the initial mean and standard deviation for each gene in the global variables declared in the header.

D. Change frequencies. This can be changed in roulette\_algorithm.cpp by adjusting the frequency minimum, maximum, and step defined in the global variables declared in the header. If changing the number of frequencies, adjustments also need to be made to simulationPECmacroskeleton.txt (variable freqCoefficients) and XFintoARA.py (variable fre-



quency\_number). Please note that the current build of AraSim will crash for anything other than the default number of frequencies and step size.

Execution:

1. Using the terminal (Ubuntu for PC), navigate to `~/XF_Loop/Evolutionary_Loop`.
2. Execute the shell script using the command `./XF_Loop.sh`
3. When prompted, press enter.
4. Import the relevant scripts into the XF project from `~/XF_Loop/Xmacros`
5. Execute the macro simulation `PEC`.
6. When the simulations are all finished executing, execute the macro output.
7. Close XF, and return to the terminal (Ubuntu for PC). Press enter when prompted.
8. The loop should repeat. This will continue for a set number of generations.

## **3 The Software**

### **3.1 An Introduction to the Loop**

The Evolutionary loop is defined to be the software package which contains all of the code responsible for the Genetic algorithm, simulation software, AraSim, and fitness function (as well as plotting software). See figure 2 for a description of each piece of software involved in the loop. The following is our genetic algorithm key. This key tells you what scripts are involved/included in each section of figure 2.

(A)

GA script - `roulette_algorithm.exe`

GA output - `generationDNA.csv`

(B.i)

output.xmacro - output.xmacro

Output macro skeleton - outputmacroskeleton.txt

(B.ii)

simulation\_pec.xmacro - simulation\_pec.xmacro

Skeletons: simulationPECMacroskeleton.txt, and simulationPECMacroskeleton2.txt

(B.iii) Commands to call XF:

./xfui [XF project location] --execute-macro-script=[output.xmacro location]

./xfui [XF project location] --execute-macro-script=[simulation\_pec.xmacro location]

(C)

.uan files - ().uan

.dat files - evol\_antenna\_model\_().dat

(D)

Grabbed .dat files - evol\_antenna\_model\_().dat

Outputted .txt files - AraOut\_[] .txt

(E)

Fitness Score generators (both can create the executable)- fitnessFunction\_ARA.cpp (looks at  $V_{eff}$ ), and fitnessFunction\_XF.cpp (looks at gain)

Executable- fitnessFunction.exe

Fitness score collector- gensData.cpp

Puts max into - maxFitnessScores.csv

(F) Plotting function- FScorePlot.py

### 3.1.1 Bash Script

The bash script is the script named XF\_Loop.sh. Our loop is composed of a multitude of different programs (XFdtd, .cpp, and python) that are all intertwined. In order to run them in a row without user involvement, we use a bash script. In other words, the bash script controls when every software and code gets implemented, controls the movement of data files in the system, and provides the input parameters for every code automatically.

The bash script has many important roles for the user. It starts with a set of three variables to adjust when starting a run. The first variable is called RunName, which is the name given to the project in XF and the name of the folder the project will be made in. The variable TotalGens is the number of generations the loop will run for after the initial generation. NPOP refers to the number of individuals you want to create and evolve with during each generation. Currently, the FREQ variable must be held at 60 since AraSim requires a certain set of frequencies. Another important role of the bash script is to create the .xmacro files for XF. These files are composed on the bash script by stitching skeleton files containing XF formatting and run information that never changes with parameters that can change between different runs. As mentioned earlier, the bash script's main purpose is to run our software and code in order, and move the outputs around automatically. Without this functionality, running the loop would be a fully manual venture. To this end, the bash script runs codes throughout it's length, and files can be seen moving after every code or software run. Passing information to the user is critical for monitoring the loop's progress,

and to this end, the bash script sends updates to the user via terminal at different parts of the code. Also, we send fitness score plots into the RunName folder in real-time as the loop continues, allowing the user to view the progress of the loop at any time.

### 3.1.2 Genetic Algorithm

The genetic algorithm is the process that evolves our antennas between generations. The code uses two algorithms. The first is a roulette algorithm, which assigns each previous antenna a probability based on its fitness score and chooses individuals for parents, evolving based on that score until a convergence is noticed. The second algorithm This will ultimately lead to the desired parameters of the bicone antenna. **Very Important :** There are two options that one can choose for how they want to calculate the fitness scores and you must make sure you compile with the correct one otherwise the code will not run, more will be discussed in section 3.2.2.

The algorithm works as follows. We start with the previous population members (with size NPOP). We randomly draw a set of individuals (with size TOURNEY\_LOTTERY\_SIZE) from the previous population. Then, we choose the individual with the highest fitness score to be our first parent. This process is done  $\text{TOURNEY\_PROPORTION} \times \text{NPOP}$  times to generate some of the future offspring. The remaining offspring are generated using random mutations to keep the genetic diversity high.

### 3.1.3 XF Simulation Software

XFtd (XF) is a computational electromagnetism simulation software developed by REMCOM using the finite difference time domain method for calculations. The antenna and its properties are simulated in XF by hitting an artificial burst of radiation on the antenna to calculate its gain patterns.

There are five pieces of code associated with XF: `simulation.PEC.xmacro` and its two

skeletons and ouput.xmacro and its skeleton. The skeletons are merely just the text that is constant for each script. The simulation\_PEC.xmacro script builds the antenna, creates the waveform, and queues and runs each simulation. The waveform is a sinusoid, which means for 60 frequencies, there is 60 simulation per individual. Note: This takes a long time. The output.xmacro script has XF output the gain vs. angles theta and phi for each frequency in a .uan file (see XFtd manual pg 351 ). This file format is not convenient for ARASIM, so we must convert it to a .dat file.

### 3.1.4 XF Output Conversion Code

In order to run ARASIM we need to make the files that XF outputs readable by ARASIM. This means converting the .uan files from XF into .dat files that ARASIM can read. This is done in the XFintoARA.py file (BiconeEvolution/current\_antenna\_evo\_build/XF\_Loop/ Evolutionary\_Loop/Antenna\_Performance\_Metric/XFintoARA.py). Once this is done we move them into the ARASIM directory (/datapool/projects/ara/AraSim).

### 3.1.5 ARASIM Execution

AraSim generates neutrino events independent of each other, with interaction point locations chosen with a uniform density in the ice. For computational ease, neutrinos are generated within a 3-5 km radius around the center of a single station for neutrino energies from  $E = 1017 \text{ eV}$ - $1021 \text{ eV}$ , with the larger radii used for higher energies. AraSim then performs ray tracing and attenuation on the signal and calculates what electromagnetic radiation reaches the station. At the station, the gain and phase data from XF is used to calculate the sensitivity of the antenna to neutrinos. This value is extracted as an effective sensitive ice volume.

### 3.1.6 Fitness Score Generation

Now that ARASIM has successfully run, we want to take the data recorded and use it to generate fitness scores. The fitness scores determine how well each specific antenna performed; we can use this information to compare it with other antennas with different parameters. This way we can determine which antenna performed the best. The ARASIM data for each generation is concatenated into one text file where each ARASIM antenna output is separated by a space. This data is fed into `fitnessScores.exe` which will generate an individual fitness score for each antenna based on the effective volume of ice observed. Finally, `gensData.py` will extract useful information from the fitness scores and write to `maxFitnessScores.csv` and `gensData.csv` which give results about which performed the best.

A key detail about this code is that it can be compiled to calculate the fitness scores in two different ways. The first fitness function (`fitnessFunction_ARA.cpp`) determines the maximum effective volume output by ARASIM and uses this to figure out which antennas should be parents for the next generation. The other fitness function (`fitnessFunction_XF.cpp`) gets the average gain of the antenna and uses that to report a fitness score. Regardless of which one you choose it is very important that you are consistent and compile the right one and ultimately only have one executable and not two, otherwise you will run into problems when trying to run the software.

### 3.1.7 Plotting

The plotting is done using `FScorePlot.py`, and are made after each generation. The function accesses the saved `i_fitnessScores.csv` files in the `Run_Outputs` directory for its data. These files store the fitness scores of each individual for the *i*'th generation. The program saves a 2D plot of the fitness scores for each generation and the ones previous. This plot is recreated after each generation, with the new generation's data and all previous data. Also, a 3D plot of generations and individuals vs. fitness score pops up on screen. This plot is not saved.

Note: We are in the process of writing a plotting script translating uan files to csv that will then plot gain vs theta and gain vs phi for each individual and each generation to display the gain patterns.

### 3.1.8 Editing Variables

The global variables appear at the top of the bash script. These are "RunName", "TotalGens", "NPOP", and "FREQ". When editing the "RunName" variable, make sure to save the XF project as the same name to avoid errors (see section of XF Usage). Changing "TotalGens" and "NPOP" requires no extra changes to be made in other programs. The "FREQ" variable should not be changed from 60 as ARASIM requires 60 (specific) frequencies.

The variables that deal with locations are "XFexec", "XFProj", and "AraSimExec". "XFexec" is the location of the XF program and where the simulations are ran. "XFProj" is the location the XF project is stored at. "AraSimExec" is the location of the arasim program and where the simulations are ran. There should be no reason to edit these unless moving off of Nutau.

## 4 Running the Loop

Here are the following important things you need to make sure you have set up prior to running the loop:

(1) If you are running the loop on Nutau (the CalPoly computer) the following software will need to be accessible. You must have some X11 forwarding setup on your personal machine, because the graphic user interface (GUI) cannot be suppressed when running XFdttd – ie the user graphics need to be loaded when using XFdttd, and will need to be forwarded from Nutau to your computer.

For a Mac, you should already have XQuartz and will not need to do anything. Similarly,

Linux also already has X11 forwarding (called Xorg), and nothing additional will need to be installed; however, if you have a PC you will have to download an X11 forwarding such as MobaXTerm (<https://mobaxterm.mobatek.net>).

(2) You must have the proper fitness score software compiled. See section 3.1.6 for more details.

(3) You must have all of your variables setup to run. See section 3.1.8 for more details.

#### **4.0.1 Initializing the loop**

Once everything is properly set up on a machine (see sections above), here are the steps to initialize the loop. Note that there is very little that must be done manually outside of XFdttd.

Step 1: To initialize the run, navigate to the folder `BiconeEvolution/current_antenna_evo_build/XF_Loop`

Step 2: Once you are there, type `"ls"` and confirm that the script `XF_Loop.sh` exists.

Step 3: Open the script in an editor the following way. Type `"emacs XF_Loop.sh -nw"`. This should bring up the bash script in an editor. Now, you want to edit the global variable `'RunName'` to a name of your liking.

As an important note, our directories are setup so that all of the outputs (our `.uan` files that have the gain, our fitness scores...etc), our timestamp (says the date and time the run was done), and the XF project (looks like a `.xf` file) get put into the same directory. In the first few lines of the `XF_loop.sh` script (ie our bash script) – under `"LINES TO CHECK OVER WHEN STARTING A NEW RUN"` – there is a variable called `"RunName"`. This will be the name of the folder all of these items will be placed in. Please be sure that you change the name of this variable for each run; otherwise, it will write over previous run data. Note that you must also name your `.xf` project the same name as this variable. We will discuss this in



greater detail later in this section.

Step 4: Save these changes by hitting 'control+x' and then 'control+s'. Then close it by hitting 'control+x' and then 'control+c'.

Step 5: We are now ready to run the loop. To do so, type './XF.Loop.sh'. This will start running the loop. Once the GA is done running (finishing in less than a few seconds), it will ask you to press any key to load XFdttd. You are now ready to move onto the next section. Remember, be patient; XFdttd is very slow.

#### **4.0.2 Navigating XFdttd**

Just to start, there are a few preliminary things to note. When running the loop the first time of a run (ie generation 1), we need to setup XF with the scripts and save the project. In the generations after the initial for any run, you will not need to repeat this process. BE CAREFUL, however; EVERY time you start a new run, you will need to repeat this process for the first generation again.

Now for steps to physically run the XFdttd section of this loop:

Step 1: Once you have run the .sh script (./XF.Loop.sh), the display will ask you to press any key to boot XFdttd (the simulation software made by Remcom). Note that XFdttd's graphics load very slow since its graphics are forwarded. Be patient – don't click around. Now, a pop up will appear saying "Unable to find or open Project.xml in the specific directory!" (See figure 3). Click "no" to proceed.

Step 2: We now need to import scripts. In the left hand column, there is something that

says "scripts". Right click that and select "Import Scripts.." as seen in Figure 4.

Step 3: A window will pop up to select scripts in Nutau. Import the Simulation\_PEC.xmacro and Output.Xmacro scripts as seen in figure 5.

Step 4: Now you must save your project. To do so, click "File", then "Save Project As...". This should pop up your folder it will be trying to save to. This should automatically be the correct folder. To be sure, ONLY "runDate.txt" and "0-generationDNA.csv" should exist in that folder as seen in figure. Note that whatever you name this 'RunName' variable in the XF\_Loop.sh script on step 3 of section 4.0.1, you must also name your XF project the same thing; otherwise, this will throw you errors! 6.

Step 5: We now need to Run Simulation\_PEC.xmacro. To do so, you double click on "Simulation\_PEC" under "Scripts" on the left hand side. Once the script pops up, you will need to hit the green run button (looks like an arrow). See figure 7.

Note that this is the slowest part of the process. This will take quite a while to run. To start, it will draw each geometry first. On the screen you will see it outputting "Successfully created the simulation" continuously until it has completed drawing the geometry for each individual and each frequency for each individual. When we eventually use broadband signals, this will look like just one frequency for each individual. Once it shows "End evaluating macro Simulation\_PEC" you can click on "simulations" on the right hand side to see the progress of each geometry being simulated. This will take a while.

Step 6: Once Simulation\_PEC is done running, we will follow the same process as step 5 to run Output.xmacro. On the left hand side, double click on "output" under "Scripts". The output script should pop up, and you can now hit the green run button (looks like an arrow). Running Output.xmacro should be very fast.

Step 7: Once Output.xmacro has run, you now need to kill XFdttd. To do so, quit your X11 forwarding completely.

Step 8: The bash script will now ask you to press any key to continue. When you do so the loop will continue (see figure 2 for details). After this happens, XFdttd will reboot again. From now on you only need to do steps 5-7 for each generation. Remember, we preset the number of generations as a variable in the bash script. So, you will need to repeat this until the loop stops – ie we've done this the same number of times/generations we set in the bash script.

Once you are done, you're now free to look at your data. Remember that it exists in the folder: BiconeEvolution/current\_antenna\_evo\_build/XF\_Loop/ Evolutionary\_Loop/Run\_Outputs/'RunName'. Remember that this 'RunName' is whatever you called the run in your bash script in step 3 of section 3.2.1. Cheers!

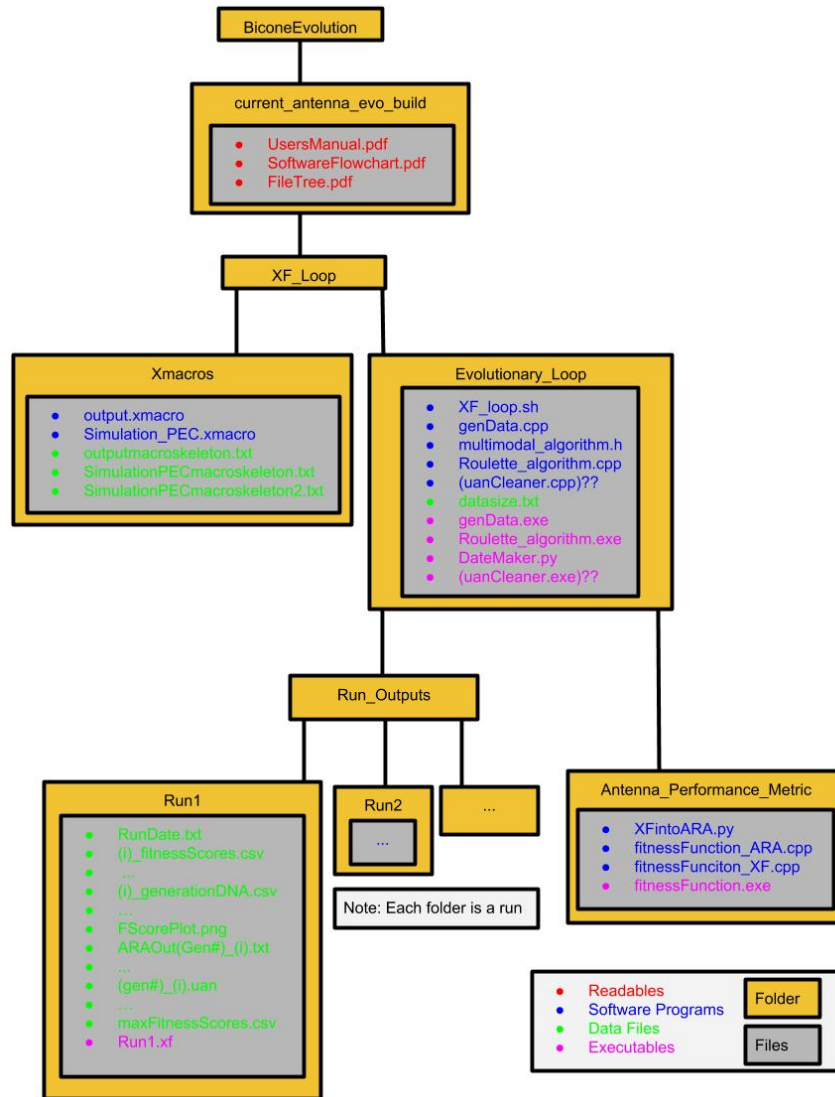


Figure 1: Image of Directory Setup for Genetic Evolution Loop.

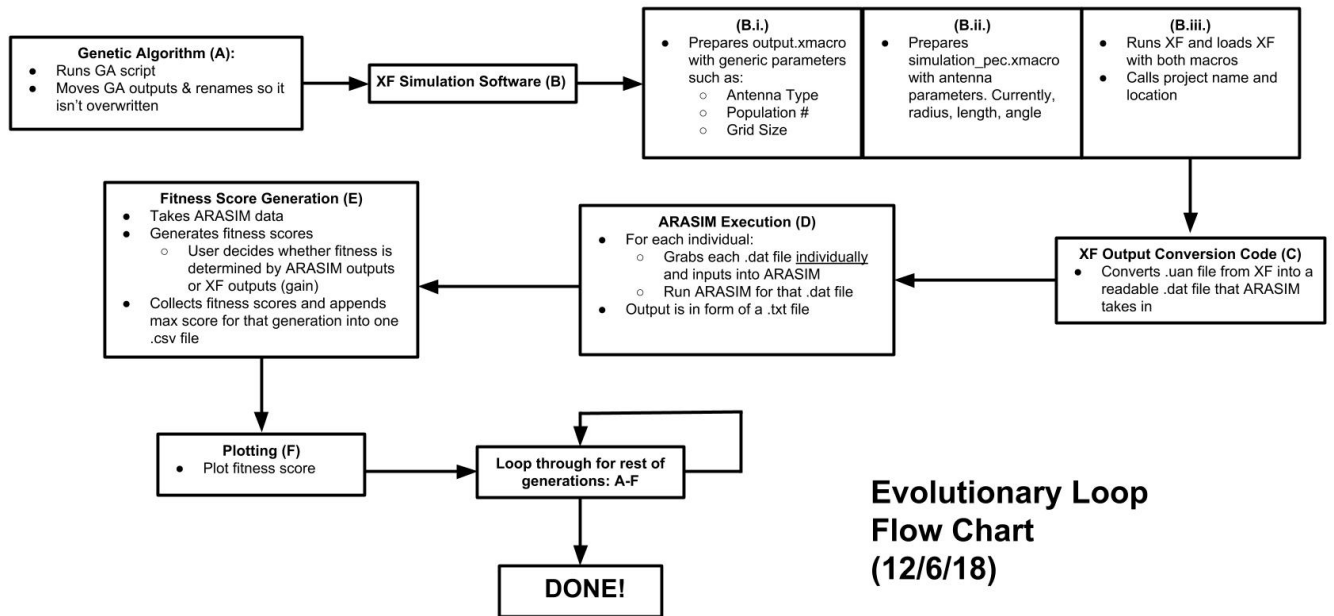


Figure 2: Graphic of Genetic Evolutionary Loop.

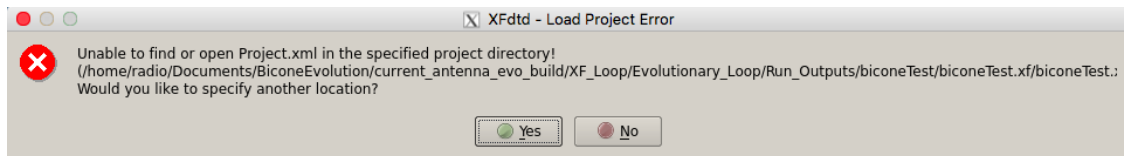


Figure 3: XFtdt Popup.

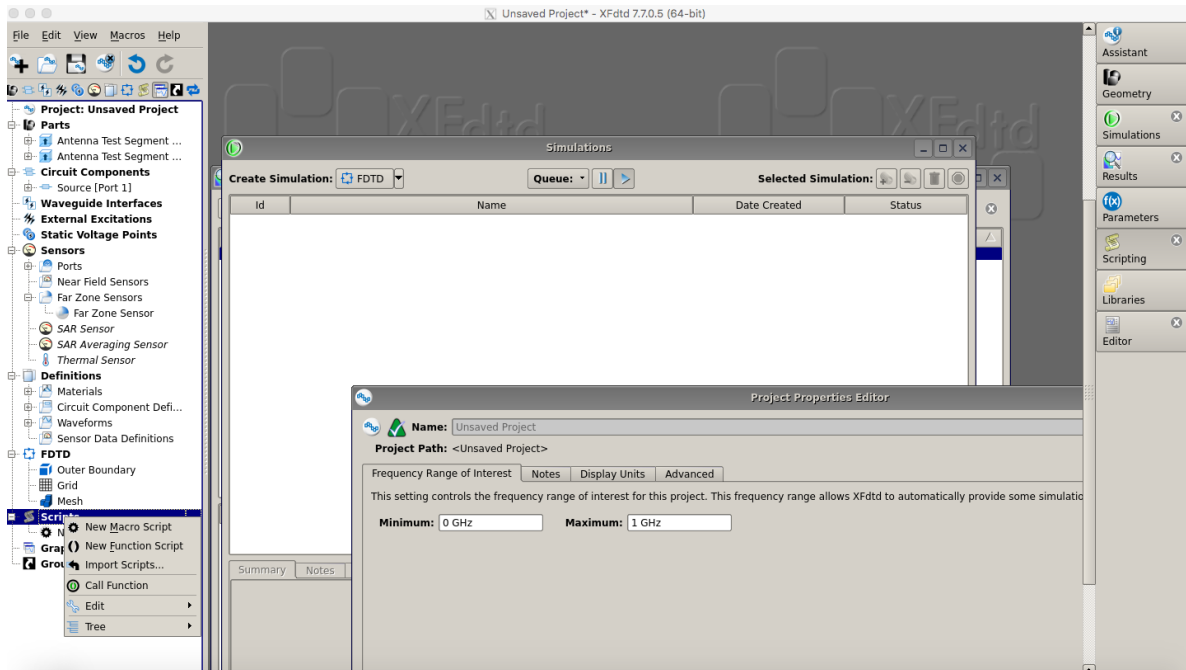


Figure 4: How to Import .xmacro Scripts.

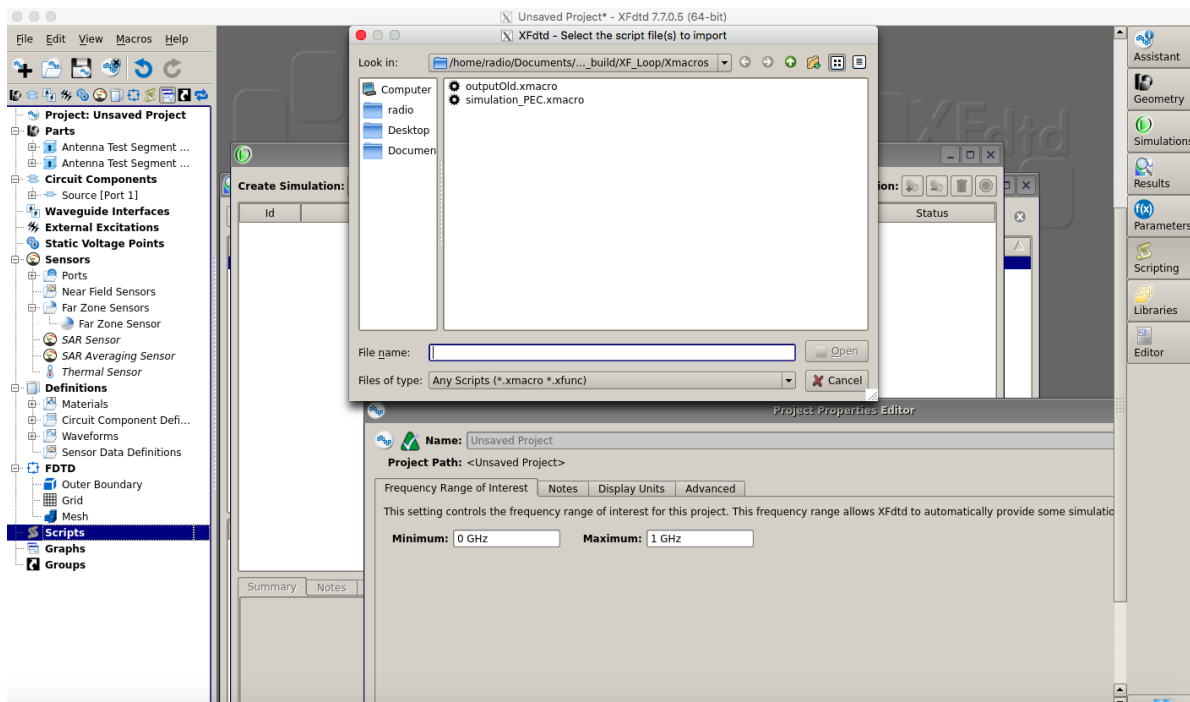


Figure 5: Selecting .xmacro Scripts to Import.

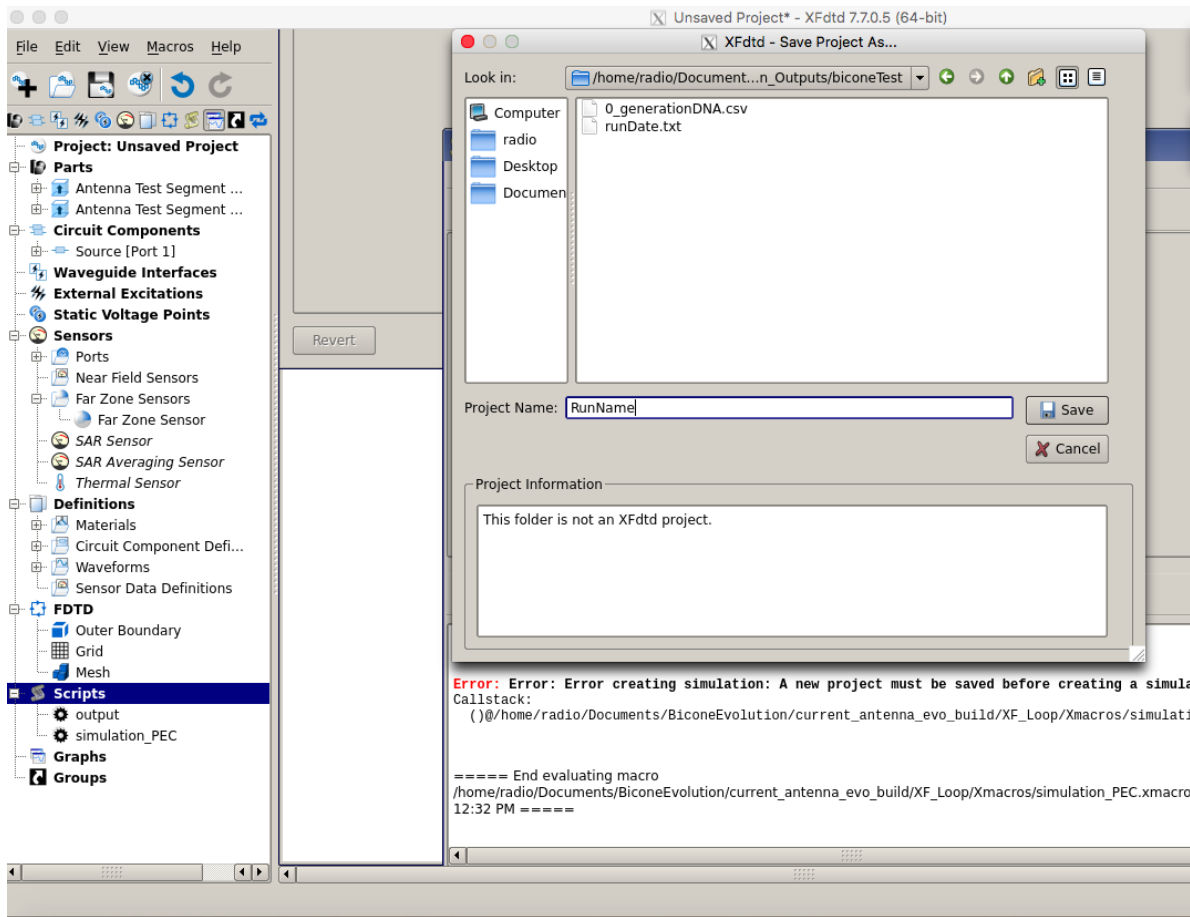


Figure 6: Saving XF Project

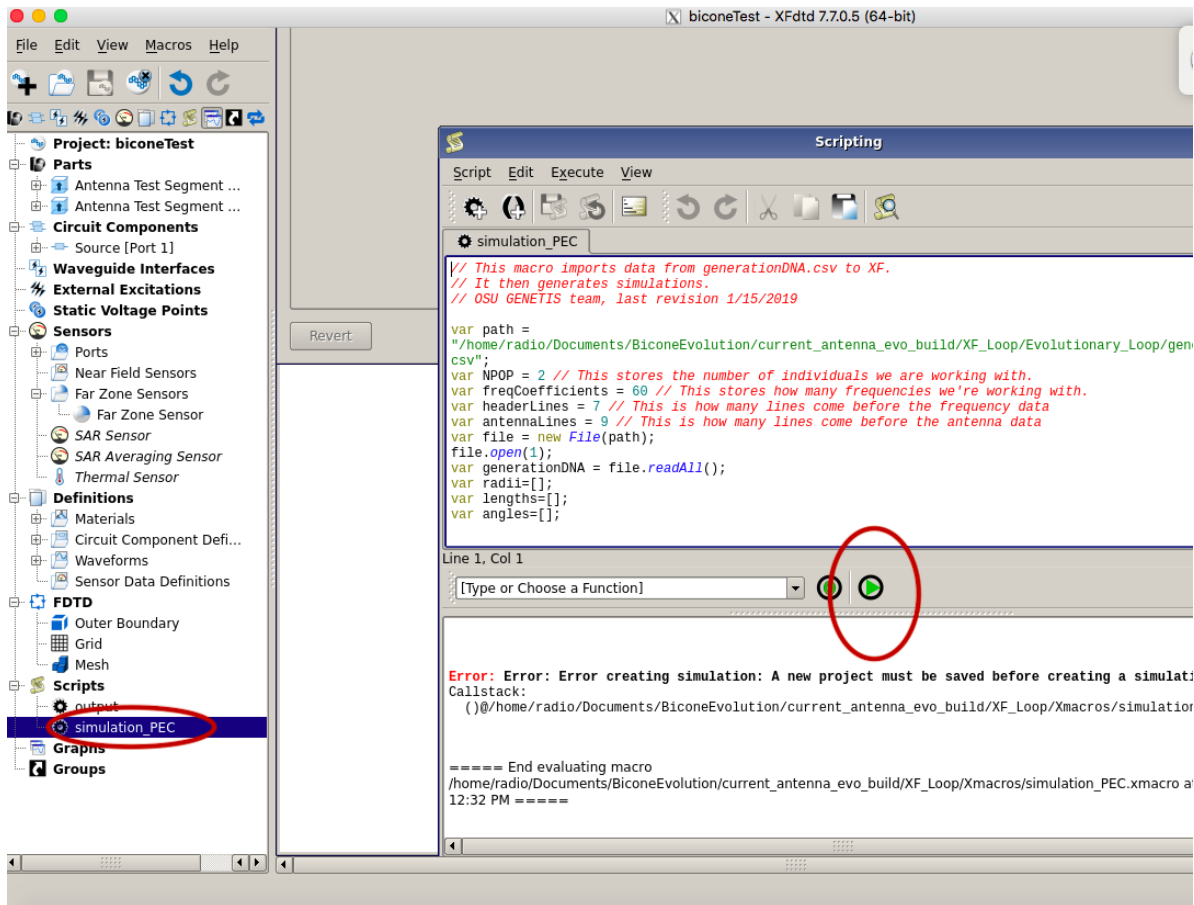


Figure 7: Running Simulation\_PEC.xmacro