XF Loop Setup and Usage Instructions

Hannah Hasan

The Ohio State University

May 18, 2018

1 Introduction

This document details how to set up and run genetic algorithms to evolve dipole antennas using XFdtd. After the initial setup, runs are fairly automated and require minimal extra configuration between runs. As a result the **Setup** section will be considerably more lengthy than the **Compiling and Running the Loop** section. However, please do not neglect the information in the second section, for it is equally as important.

2 Setup

This section includes instructions for setting up the necessary files and directories for the algorithm.

2.1 Download

Ensure that you have the following files (available at `https://github.com/hchasan/XF-Scripts`):

- `Evolved_Dipole_CMD.cpp`
- `Roulette_Select.cpp`
- `XF_Loop.sh`
- `dipolePECmacroskeleton.txt`
- `dipolePECmacroskeleton2.txt`

- dipolePECmacroskeleton3.txt
- gainPlot.py
- gensData.cpp
- handflail.cpp
- makehandshook.cpp
- outputmacroskeleton.txt
- plotLR.py
- uanCleaner.cpp

Place all of these files into a single directory. For purposes of this guide we will refer to this overarching directory as XF_Scripts.

2.2 Directory Setup

1. Move into the XF_Scripts directory. [cd XF_Scripts]
2. Create a directory named Xmacros. [mkdir Xmacros]
3. Create a directory named Projects. [mkdir Projects]
4. Create a directory named Evolved_Dipole. [mkdir Evolved_Dipole]
5. At this point, the contents of XF_Scripts should be the files listed in Section 2.1 and the three directories: Xmacros, Projects, and Evolved_Dipole.
6. Move all non-directory files into Evolved_Dipole. [mv (filename) Evolved_Dipole]
    - Evolved_Dipole_CMD.cpp
    - Roulette_Select.cpp
    - XF_Loop.sh
    - dipolePECmacroskeleton.txt
    - dipolePECmacroskeleton2.txt
    - dipolePECmacroskeleton3.txt
    - gainPlot.py
    - gensData.cpp

- handflail.cpp
- makehandshook.cpp
- outputmacroskeleton.txt
- plotLR.py
- uanCleaner.cpp

This may be easier to do by moving similar files in one go; for example, all text files can be moved at once with `mv *txt Evolved_Dipole`

7. Move into the `Evolved_Dipole` directory [`cd Evolved_Dipole`].
8. Create a directory called `hands` [`mkdir hands`].
9. Create a directory called `data`
10. Move the following files to the `data` directory:

    - dipolePECmacroskeleton.txt
    - dipolePECmacroskeleton2.txt
    - dipolePECmacroskeleton3.txt
    - makehandshook.cpp
    - outputmacroskeleton.txt

    To make things easier, move the skeleton files in a single command by typing `mv *skeleton* data`

11. Still in `Evolved_Dipole`, create a file called `watch.txt` that contains just the number `"0"`.
12. Now the contents of `Evolved_Dipole` should be the directory `data` (which itself contains `makehandshook.cpp` and the 4 skeleton files), and the following files:

    - Evolved_Dipole_CMD.cpp
    - Roulette_Select.cpp
    - XF_Loop.sh
    - gainPlot.py
    - gensData.cpp

- handflail.cpp
- plotLR.py
- uanCleaner.cpp

2.3 Configuring Scripts and Code

Open XF_Loop.sh and make the following changes. Note that line numbers may not be exact due to updates made after the time of writing:

- On line 18, set the variable XFexec to the location of the XF executable on your machine. Use Linux format.
- On line 29, set the end of the variable lastline to the path for the data directory. If you are on Windows, the path will be from your C: drive. If you are on Linux (perhaps running on NuTau), this should be in Linux format.

3 Compiling and Running the Loop

3.1 Compiling

All .cpp files must be compiled before use. To do this, move to the directory of the file and use the following commands:
For the following files, type g++ (name).cpp -o (name).exe:

- Evolved_Dipole_CMD
- Roulette_Select
- gensData
- uanCleaner

For these files, type g++ (name).cpp -std=c++11 -o (name).exe:

- makehandshook
- handflail

3.2 Running

Before a new run, be sure to:

1. Open `XF_Loop.sh` and on line 9 set the variable `RunName` to whatever you want to name this run.
2. Choose the number of generations to run after the parent generation on line 10 of `XF_Loop.sh`.
3. Select which mutation program to use by un-commenting *one* of the programs at line 42, at line 151, and at line 267 of `XF_Loop.sh`
4. Check that all files have the variable `NPOP` set to the same number. This is the number of individuals per generation. In `handflail`, make sure the sum of the variables `ROUL_X_NO, ROUL_MUT_NO, TOUR_X_NO, TOUR_MUT_NO` is equal to `NPOP`. Re-compile if the value is changed for any `cpp` files.

Once these have been accounted for, begin the loop:

1. Type `./XF_Loop`
2. If running on Windows, plug in the USB software key. Then press a key, as prompted.
3. Once XF is open (this will take a while if you are running on NuTau), import the `dipole_PEC.xmacro` and `output.xmacro` scripts, which should be located in the `Xmacros` directory created in Section 2.2.
4. Save the project in the `Projects` directory, also created in Section 2.2. Be sure to give it the same name as RunName.
5. Run `dipole_PEC.xmacro`. This should create and begin simulations for the individuals of the generation.
6. If you are on Windows, open the "Simulations" window and click the "output" tab. Check here from time to time, because the program will lose connection to the license key at times. If this happens, you can remove the key and plug it back in, then manually queue the unfinished simulations.
7. After all simulations are complete, run `output.xmacro`.
8. Close XF and give a keypress as prompted.
9. Repeat from step 5.

4 Notes

Just some extra tidbits:

- If using `Roulette_Select.cpp` as the mutation algorithm, you can run `XF_Loop` until the dipoles are of similar length, rather than running for a set number of generations.

  To do this, make sure `Roulette_Select.exe` is selected as the mutation program in `XF_Loop.sh`.

  In `XF_Loop.sh`, comment out the `for` loop on line 146 and un-comment the `while` loop in the next line.

  Choose a value for a convergence threshold on line 28 of `Roulette_Select.cpp`. *Convergence* will be when the lengths of a generation's dipoles have a standard deviation less than this value.

  Compile the program and run the loop as outlined in Section 3.2.