PerSeO: PSO Applied to RF Device Design

User Manual

Jaime Ángel, Jorge Cárdenas, John Vera, German Chaparro, Sergio Mora and Oscar Restrepo

Particle Swarm Optimization (PSO) for RF Device Design
https://github.com/ERA-2022/PerSeO

Jaime Ángel, Jorge Cárdenas, John Vera, German Chaparro, Sergio Mora and Oscar Restrepo

Special contributor: Daniela Paez Díaz

First Edition
June 2023
ISBN 978-858-8817-71-2
Bogotá
Colombia

Contents

1 Introduction 5

1.1 General Information 5

2 Set up 7

2.1 Prerequisites for Use 7

2.1.1 Python Version 7

2.1.2 Steps to follow to use the package installed from PyPI 7

2.1.3 Steps to follow to use the package with the files coming from the PerSeO repository 8

2.1.4 Installation Notes and Warnings 8

2.1.5 Ansys HFSS Software and Model 9

2.1.6 Your Custom Script 9

3 Instructions 13

3.1 Execution and Interface 13

3.1.1 Optimize 13

3.1.2 Fitness Function Test 14

3.1.3 Graphics Tools 14

3.1.4 Exit 18

3.2 Examples 18

3.2.1 Patch Antenna 18

4 Annexes 21

1. Introduction

1.1 General Information

The software package presented here provides a powerful tool designed to assist the optimization process of various components such as hybrids or antennas, where electromagnetic performance is heavily reliant on geometric configurations.

Our optimization algorithms are corroborated by HFSS simulations, thereby offering a dependable tool to assess the performance of each proposed geometry and to extract data from each optimization batch.

The current version (v1.0.2) incorporates the following features:

- Package installation using PyPI
- Import and use
- PSO optimization algorithm
- Log file: A comprehensive record of the optimization process
- Data collection in CSV files
- Provision of configuration data through JSON files or directly from the code
- Simulation control by ID
- Includes an example of demonstrating PSO usage in Antennas
- Documentation of all methods and classes of the package

Handling Extended Code Lines: In code sections where lines are excessively long, the line will be continued on the next one. This continuation will start with four points separated by spaces, as shown in the following example.

print("Hi my name is . . . .
. . . . Robert Dakar")

The equivalent continuous line is also provided for reference.

print("Hi my name is Robert Dakar")

2. Set up

2.1 Prerequisites for Use

Before using the package or the files found in the PerSeO repository[1], certain prerequisites need to be fulfilled to ensure seamless operation. These requirements include installing Python along with the perseo-optimizer package or necessary packages, Ansys HFSS 2019, IronPython and the perseo-optimizer package, setting up the file paths appropriately, and preparing your custom Python script.

2.1.1 Python Version

The package requires Python version 3.10.4 to function correctly[2]. Other versions may cause compatibility issues with certain required packages for the optimizer.

2.1.2 Steps to follow to use the package installed from PyPI

you must install the perseo_optimizer package[3], which contains all the files and data for its use. to install the package, use the follow command in your terminal:

```
pip install perseo-optimizer
```

On the other hand, Ansys HFSS uses IronPython to use the scripting option, for this reason, it is also necessary to add the perseo_optimizer package to the IronPython packages. The folder containing the packages is located where Ansys was installed, inside the `./common/IronPython/lib/` folder, the path is usually as follows:

```
C:/Program Files/AnsysEM/AnsysEM19.0/Win64/common/IronPython/Lib/
```

Once you have located the path where the Ironpython packages are located, use the following command in your terminal (it is recommended to open it as administrator):

[1]The repository is available at *https://github.com/ERA-2022/PerSeO*
[2]The python version is available at *https://www.python.org/downloads/release/python-3104/*
[3]The package is available in PyPI at the following link *https://pypi.org/project/perseo-optimizer/*

```
pip install perseo-optimizer -t "YOUR_PATH_TO_ANSYS_IRONPYTHON_PACKAGES/"
```

2.1.3 Steps to follow to use the package with the files coming from the PerSeO repository

If you do not want to use the package by installing it from PyPI, you can use the source files found in the repository, for this, you must clone or download the source files and then install all the necessary dependencies that the perseo_optimizer package needs to work, these packages (along with their versions) are found in the file requirements.txt.

To install all these dependencies from the requirements.txt file, use the follow command in your cmd (be sure to run the command inside the folder where you cloned or downloaded the files from the repository):

```
pip install -r requirements.txt
```

2.1.4 Installation Notes and Warnings

If you use an IDE or source code editor other than Visual Studio Code or the integrated Python IDE, ensure that the package are installed or in case of using the repository files, that all the packages found in requirements.txt are installed.

It is possible that at the moment of installing the package or the dependencies you will find warning messages, this can happen if you have some other version of some of the packages, verify that the versions of the installed packages are the correct ones (you can verify this information in the requirements.txt file).

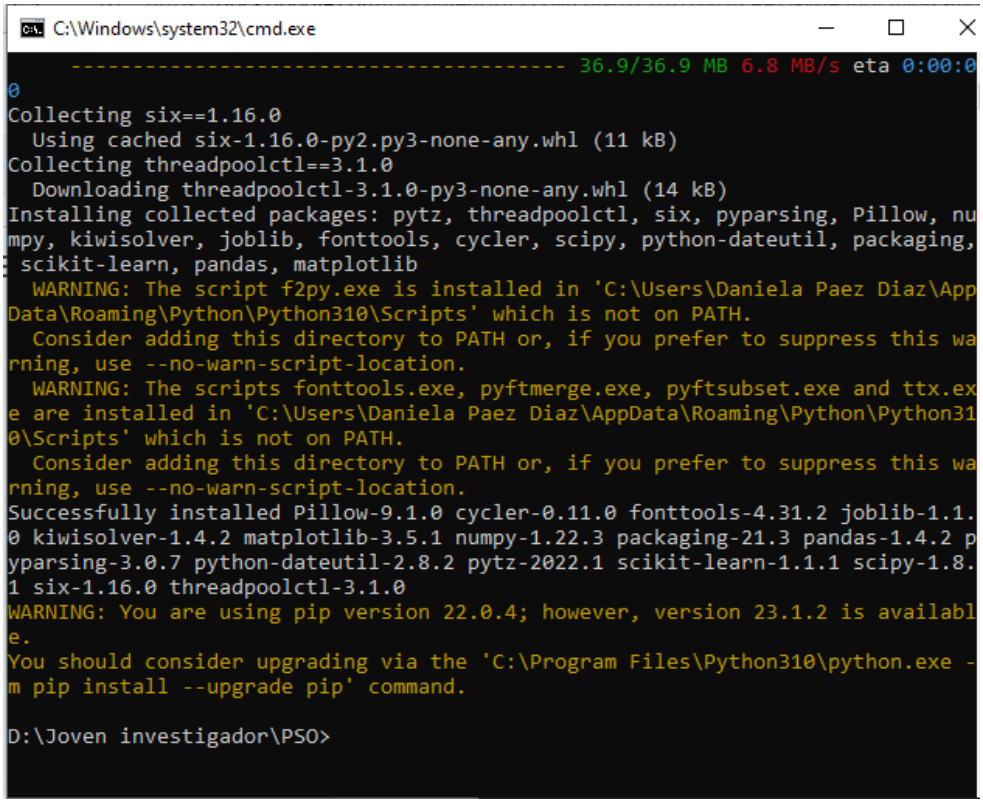

Figure 2.1: Warning message in the console.

2.1.5 Ansys HFSS Software and Model

Installation of Ansys HFSS 2021, 2019, or 2017 is required (compatibility with other versions has not been tested) as the optimizer utilizes it to simulate various models. Furthermore, you should place a Python file containing the geometric model for HFSS in the model's folder located in the root directory. If your model is an .aedt file, add this to the Ansoft folder located in the Documents folder (the default folder created by HFSS). In both cases (.py or .aedt files), the file's name should match the project name.

It's crucial that your model's dimensions are arranged in an array data structure.

The following example reveals a part of a Python file describing a dipole blade antenna geometry. Ensure the name of your file (.py or .aedt) aligns with the line responsible for defining the project's name. The functionality mentioned here will be used later on.

```
# -*- coding: utf-8 -*-
# ----------------------------------------------
# Script Recorded by Ansys Electronics Desktop Student Version 2021.2.0
# 15:07:57  may. 26, 2022
# ----------------------------------------------

import ScriptEnv
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.NewProject()
oProject.Rename("C:/Users/Astrolab/Documents/Ansoft/DIPOLE_BLADE_ANTENNA.
    aedt", True) #This lineis responsible for the path and renaming the
    project
oProject.InsertDesign("HFSS", "HFSSDesign1", "HFSS Modal.Network", "")
oDesign = oProject.SetActiveDesign("HFSSDesign1")
oDesign.RenameDesignInstance("HFSSDesign1", "DESIGN")
```

2.1.6 Your Custom Script

As previously mentioned in Section 2.1, we offer a dedicated Python module for integration into your script. To achieve this, follow the steps outlined below:

- **First step:** Add the following imports to your main script.

```
from perseo_optimizer import commands, interface
```

- **Second step:** Define the following parameters for optimization. It's recommended to store this data in separate variables:

  - Path to ANSYS executable
  - Default save path for ANSYS
  - Structure type (Antenna, Hybrid, Filter, among others)
  - Structure subcategory
  - Project name
  - Design name
  - Variable name or array name holding the dimensions

- Units (of distance)
- Maximum values for the optimization variables
- Minimum values for the optimization variables
- Nominal or default design values
- Number of iterations
- Number of particles
- Number of branches
- Simulation description and relevant information
- Reports required for the fitness function

These data will be used as parameters when initializing the optimization using a later view method. Please refer to the following example:

```
exe = "C:/Program Files/AnsysEM/AnsysEM19.0/Win64/ansysedt.exe"
save = "C:/Users/Astrolab/Documents/Ansoft/"
category = "Antenna"
sub_category = "Dipole blade"
pname = "DIPOLE_BLADE_ANTENNA"
dname = "DESIGN"
vname = "variables"
u = "mm"
ma = [12650, 1300, 1600, 65, 20, 2.5, 1700]
mi = [8200, 750, 950, 30, 20, 2.5, 750]
nom = [8333.33, 813.33, 1043.33, 36.66, 20, 2.5, 866.66]
i = 2
p = 2
b = 0
desc = "100%BW with ideal BW to 80MHz, denominator (or cut-off
    frecuency) in 40MHz working in the frecuency band of 40MHz to 120
    MHz"

reports = {
    "SMN":[(1,1)],
    "gain":[0,90],
    "vswr":[1],
    "zmn":[(1,1)],
    "aditional_data":{
        "fmin":40,
        "points":81,
        "units":"MHz"
    }
}
```

- **Third step:** You need to define your fitness function. This function should accept one parameter (a dictionary containing required report data) and should return the fitness function's value.

```
def fit (dataReports):
    for key in dataReports:
        print(str(key)+"--->"+str(len(dataReports[key])))

    areas_f = []
    areas_d = []
    new_area = True
    for data in dataReports['S11']:
```

```
        if data[1] < -9.8:
            if new_area:
                new_area = False
                areas_f.append([])
                areas_d.append([])
            areas_f[len(areas_f)-1].append(data[0])
            areas_d[len(areas_d)-1].append(data[1])
        else:
            new_area = True

    print(f"\nNumber of areas: {len(areas_f)} || Mat.freq and Mat.db
    with same size: {len(areas_f) == len(areas_d)}\n")

    coefficient = 0
    if len(areas_f)==0:
        coefficient = 20
    else:
        bw = 0
        freq = 0
        for area in range(len(areas_f)):
            temp = areas_f[area][len(areas_f[area])-1] - areas_f[area
    ][0]
            print(f">{temp}")
            if bw < temp:
                bw = temp
                freq = areas_f[area][areas_d[area].index(np.min(areas_d
    [area]))]

        if bw > 0:
            print("INFO PREVIEW")
            print(f"BW: {bw}Mhz")
            print(f"Freq: {freq}Mhz")
            coefficient =  ((80-bw)/80)**2

        else:
            coefficient = 20
    return coefficient
```

In the same way that the before point, this function will use later.

- **Fourth step:** The system should be initialized with the *init_system(...)*. This method is part of the commands imported in the first step, and arguments defined in the second step should be passed to it.(for more information see the Table 4.1).

```
commands.init_system(exe, save, pname, dname,vname, u, ma, mi, nom, i,
    p, b, reports, category, sub_category, desc)
```

- **Fifth step:** The *main_menu(...)* function must be used, which stems from the interface of the second import, and you need to add the fitness function as an argument.

```
interface.main_menu(fit)
```

3. Instructions

3.1 Execution and Interface

Once you have verified all prerequisites and saved your main script, you can execute the script via the command line (cmd), as illustrated below.

```
C:\Users\Astrolab\Documents\Jaime\Temporal>python3 Example_1.py
```

Alternatively, you can execute your code using an Integrated Development Environment (IDE) or source code editor such as VSCode, Spider, or Conda.

To engage with the optimizer's interface, you'll have access to the following options:

```
>>>>
--------->PSO APP<---------
-----\MENU
1> Optimize
2> Fitness function test
3> Graphics tools
4> Exit
Enter an option:
```

3.1.1 Optimize

First and foremost, this option checks if the model exists using the Ansys or Python files. If the software fails to run the files, it will automatically attempt to execute them successfully on a repeated basis. If this does not succeed, the software will alert you about the error and revert to the main menu[1].

[1]This process may involve an average of 22 attempts at successful program execution, indicating that there has been an error. Thus, while the process is underway, please ensure that the path of your design script is correct, avoid having the project open in Ansys unless it has been opened by the optimization process, and that the Python version you're using is compatible.

Once the model is confirmed, the software will inquire if you wish to generate graphics from the reports. Enter 'Y' or 'N' in the console and press Enter. The optimization process begins by generating particles, which represent new random dimensions. These designs are then simulated in ANSYS, and the specified reports are exported. If needed, graphics will be produced and saved to ../ID/figures/. Subsequently, the software assesses the fitness function using the report data and determines the dimensions for new particles. This cycle continues for all iterations and particles until the process is complete.

Upon completion of the optimization process, all obtained results will be saved in the _ output.csv file located in the results folder (.../results/output.csv). In this file, you will find the ID, start and end times, type, category, sub-category, simulation parameters, results, and optimal particles, organized row-wise as iteration summaries. If you require more detailed information of the optimization process, in the control.log file located in the src folder (../src/control.log) you can see details such as the generated dimensions, the calculated value of the fitting function and running errors if found, for each particle of each iteration.

3.1.2 Fitness Function Test

This option allows for the execution of a new simulation based on previous results. It requires the previous simulation's ID and the various report files (.csv), as well as the exact parameters of the previous simulation. Like the Optimization option, you can choose whether to generate graphics.

This option offers a quick test with different fitness functions, as it bypasses the time-consuming simulation step. However, you should understand that the optimization might reveal new dimensions that have not been simulated.

Similarly to the previous option, all obtained data will be saved in the output.csv file located at ../results/output.csv.

3.1.3 Graphics Tools

This option provides a menu that allows for the creation of graphics from the simulated reports, as outlined below.

```
>>>>
-----\Graphics tools
1> Draw one report
2> Draw one complete iteration
3> Draw one complete execution
4> Draw one report comparisons
5> Back
Enter an option:
```

Draw One Report

Initially, this option will request the ID of a previously executed simulation. Then, you will be asked to enter the file name to draw (the '.csv' extension is optional). Following this, you'll need to define the x-axis label, which is associated with magnitude in frequency (not applicable to the gain phi graphics). The process will conclude with a message indicating that the process has ended.

```
>>>>
-----\Graphics tools
```

```
1> Draw one report
2> Draw one complete iteration
3> Draw one complete execution
4> Draw one report comparisons
5> Back
Enter option: 1
Enter a previously simulate ID: b2466c28-8fe8-4b71-af6c-fd435b6e5418
Enter the file name: datosS11_1_0.csv
Enter the magnitude of frequency (for example Hz, KHz, MHz, GHz,. . . .
. . . .among others): MHz

The process has ended,  verify that the draw graphic is in. . . .
. . . .'figures' folder in the entered ID folder
Press intro to continue...
```

The graphics generated will be stored in the figures folder within the previously requested ID folder.

Draw One Complete Iteration

As in the previous option, this will request a previously simulated ID followed by the number of iterations to draw and the label of the x-axis (not applicable to the gain phi graphics). Next, while the software draws the graphics, the screen will show the drawn files' names, and the process will finish with a notification on the screen.

```
>>>>
-----\Graphics tools
1> Draw one report
2> Draw one complete iteration
3> Draw one complete execution
4> Draw one report comparisons
5> Back
Enter option: 2
Enter a previously simulate ID: b2466c28-8fe8-4b71-af6c-fd435b6e5418
Enter the iteration number: 2
Enter the magnitude of frequency (for example, Hz, kHz, MHz, GHz,. . . .
. . . .among others): MHz
Drawing a graphic ---> datosGananciaPhi0_2_0.csv
Drawing a graphic ---> datosGananciaPhi0_2_1.csv
Drawing a graphic ---> datosGananciaPhi90_2_0.csv
Drawing a graphic ---> datosGananciaPhi90_2_1.csv
Drawing a graphic ---> datosS11_2_0.csv
Drawing a graphic ---> datosS11_2_1.csv
Drawing a graphic ---> datosVSWR(1)_2_0.csv
Drawing a graphic ---> datosVSWR(1)_2_1.csv
Drawing a graphic ---> datosZ11_2_0.csv
Drawing a graphic ---> datosZ11_2_1.csv

The process has ended. Verify that the drawn graphic is in the. . . .
```

```
. . . .'figures' folder in the entered ID folder
Press intro to continue...
```

The graphics drawn will be saved in the figures folder between the previously requested ID folder.

Draw One Complete Execution

Similarly to the previous options, this will request a previously simulated ID and the x-axis label (not applicable to the gain phi graphics). As the software generates the graphics, the screen will display the names of the files being drawn, and the process will conclude with a tally of files read and drawn, followed by an on-screen notification that the process has finished.

```
>>>>
-----\Graphics tools
1> Draw one report
2> Draw one complete iteration
3> Draw one complete execution
4> Draw one report comparisons
5> Back
Enter option: 3
Enter a previously simulate ID: b2466c28-8fe8-4b71-af6c-fd435b6e5418
Enter the magnitude of frequency (for example, Hz, kHz, MHz, GHz,. . . .
. . . .among others): MHz
Drawing a graphic ---> datosGananciaPhi0_0_0.csv
Drawing a graphic ---> datosGananciaPhi0_0_1.csv
Drawing a graphic ---> datosGananciaPhi0_1_0.csv
Drawing a graphic ---> datosGananciaPhi0_1_1.csv
Drawing a graphic ---> datosGananciaPhi0_2_0.csv
Drawing a graphic ---> datosGananciaPhi0_2_1.csv
Drawing a graphic ---> datosGananciaPhi90_0_0.csv
Drawing a graphic ---> datosGananciaPhi90_0_1.csv
Drawing a graphic ---> datosGananciaPhi90_1_0.csv
Drawing a graphic ---> datosGananciaPhi90_1_1.csv
Drawing a graphic ---> datosGananciaPhi90_2_0.csv
Drawing a graphic ---> datosGananciaPhi90_2_1.csv
Drawing a graphic ---> datosS11_0_0.csv
Drawing a graphic ---> datosS11_0_1.csv
Drawing a graphic ---> datosS11_1_0.csv
Drawing a graphic ---> datosS11_1_1.csv
Drawing a graphic ---> datosS11_2_0.csv
Drawing a graphic ---> datosS11_2_1.csv
Drawing a graphic ---> datosVSWR(1)_0_0.csv
Drawing a graphic ---> datosVSWR(1)_0_1.csv
Drawing a graphic ---> datosVSWR(1)_1_0.csv
Drawing a graphic ---> datosVSWR(1)_1_1.csv
Drawing a graphic ---> datosVSWR(1)_2_0.csv
Drawing a graphic ---> datosVSWR(1)_2_1.csv
Drawing a graphic ---> datosZ11_0_0.csv
```

```
Drawing a graphic ---> datosZ11_0_1.csv
Drawing a graphic ---> datosZ11_1_0.csv
Drawing a graphic ---> datosZ11_1_1.csv
Drawing a graphic ---> datosZ11_2_0.csv
Drawing a graphic ---> datosZ11_2_1.csv
Files read and drawn:30

The process has ended. Verify that the drawn graphic is in the. . . .
. . . .'figures' folder in the entered ID folder.
Press intro to continue...
```

The graphics generated will be stored in the figures folder within the previously requested ID folder.

Draw One Report Comparison

For this option, it's crucial that the two files have identical magnitudes, otherwise the graphic may appear erroneous.

Initially, you'll need to provide the path (a full path is recommended, not a relative path) of the first file. Similarly, you will be asked for the path of the second file. You'll then be asked to specify the save path; if you press Enter with this field blank, the save path will default to ../results/comparison graphics/

```
>>>>
-----\Graphics tools
1> Draw one report
2> Draw one complete iteration
3> Draw one complete execution
4> Draw one report comparisons
5> Back
Enter option: 4
Enter the path file 1: C:\Users\ESTACION\Documents\GitHub\. . . .
. . . .PSO_for_hybrids_and_antennas\results\. . . .
. . . .b2466c28-8fe8-4b71-af6c-fd435b6e5418\files\datosS11_0_1.csv
Enter the path file 2: C:\Users\ESTACION\Documents\GitHub\. . . .
. . . .PSO_for_hybrids_and_antennas\results\. . . .
. . . .b2466c28-8fe8-4b71-af6c-fd435b6e5418\files\datosS11_2_1.csv
```

Next, you will be prompted for the save path. However, if you press Enter while this field is empty, the default save path will be set to ../results/comparison graphics/

- **Here is an example with a specific save path:**

  ```
  >>>>
  Note: if you press Intro without adding nothing path, this will. . . .
  . . . .save in ../results/comparison graphics/ by default
  Enter the save path: C:\Users\ESTACION\Documents\Jaime\Comparaciones
  ```

- **Here is an example with a specific save path:**

```
>>>>
Note: if you press Intro without adding nothing path, this will. . . .
. . . .save in ../results/comparison graphics/ by default
Enter the save path:
```

Subsequently, you'll be asked for the labels of the x and y-axis, the Title of the graphic, which will be identical to the save file name (this name cannot contain periods), and the label of each file that will be displayed in the graphic. Once the process is complete, a notification will be issued.

```
>>>>
Note: if you press intro without adding nothing path, this will. . . .
. . . .save in ../results/comparison graphics/ by default
Enter the save path: C:\Users\Astrolab\Documents\Jaime\Comparaciones
Enter the axis x label: Frequency (MHz)
Enter the axis y label: dB
Enter the graphic title: S11 initial particle vs final particle
Enter the label of the  data to file 1: Initial particle
Enter the label of the  data to file 2: Final particle

The process has ended. Verify that the drawn graphic is in the. . . .
. . . .entered path or the default path.
Press intro to continue...
```

3.1.4 Exit

This option allows you to terminate the currently running script.

3.2 Examples

In the PerSeO repository, an 'examples' folder is available. This folder contains several sub-folders with specific examples. All examples follow the same structure: a Python file named after the RF device that contains the design for execution in Ansys, a main.py file that sets up the PSO for optimizing the said design, and a README.md file providing information about the example's usage.

3.2.1 Patch Antenna

This example includes a script with a design for a patch antenna (Figure 3.1) operating at 1.7GHz (Figure 3.2), a script illustrating the usage of PerSeO package with a fitness function aiming to tune the antenna to 2.6GHz with a minimum of -10dB adaptation, and a README.md file with usage information.

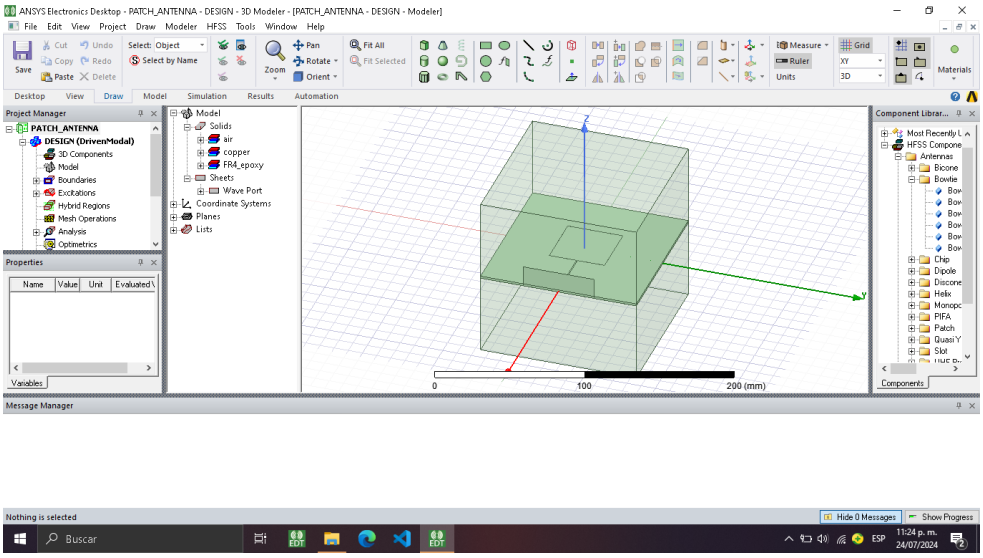

Figure 3.1: view of the patch antenna model.

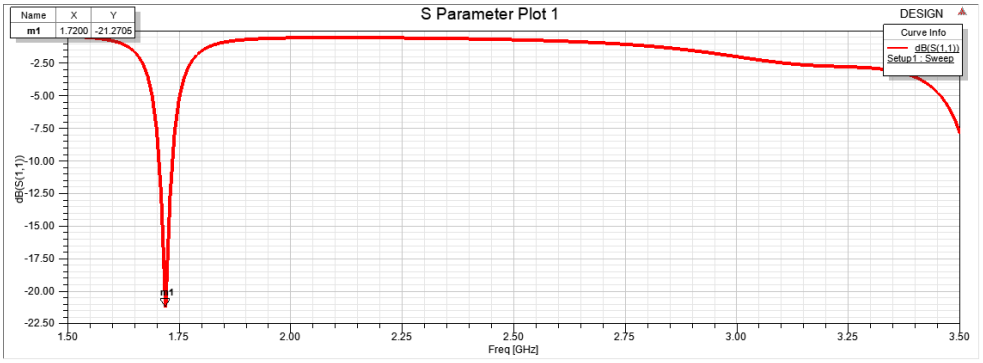

Figure 3.2: Patch Antenna Parameter S11

How to Use

1. You must have the perseo_optimizer package installed or you have the source code.
2. Next, you must copy the main.py and paste to the root of your project and run the initialize of system.
3. Later you must copy the PATCH_ANTENNA.py and paste to models folder located in the root of your project
4. In the PATCH_ANTENNA.py file, modify the path appearing in the project rename as shown below.

   ```
   oProject.Rename("C:/Users/INVESTIGACIÓN/Documents/Ansoft/. . . .
   . . . .PATCH_ANTENNA.aedt", True)
   ```

   Ensure you only change the path and not the project's name. This should be updated to your default Ansys save path (which should resemble the one provided in the example, barring the user name folder).

5. In the main file, update the *ansys_path* and *save_path* variables with your corresponding values. The former should contain the path to the Ansys executable file, while the latter should point to the Ansys default save path.
6. In the main file, uncomment the last line of code and execute the script. In the main menu, select option one, choose whether you want to generate graphics, and wait for end the optimization process.

4. Annexes

Position	argument name	data type	description
1	ansys_exe	str	containing the path to the Ansys HFSS executable, usually found in C:/program files/
2	ansys_save_def	str	containing the path where the Ansys model file is located, it is recommended to leave the default path Ansys uses to save its files
3	project_name	str	containing the name of the project within the Ansys model
4	design_name	str	containing the name of the design within the Ansys model
5	variable_name	str	containing the name of the array name of the model dimensions
6	units	str	containing the unit of measure of the model, e.g. mm (millimeters)
7	max	list[float or int]	List of numbers with the maximum values that the PSO can take to modify the model's dimensions
8	min	list[float or int]	List of numbers with the minimum values the PSO can take to modify the model's dimensions.
9	nominals	list[float or int]	List of numbers with the nominal or default values that the model has in its dimensions.
10	iterations	int	number of iterations to be executed by the PSO
11	particles	int	number of particles to be executed by the PSO
12	branches	int	number of branches (only for hybrids with branches)
13	reports	dict	Dictionary with the reports to be generated by ansys. Among these reports are: parameter Smn, parameter Zmn, Gain phi x angle, Amplitude imbalance, phase imbalance, and VSWR(x port)
14	category	str	containing the category you want to give to the model or optimization, for example, "Antenna", "Hybrid", "Filter", among others
15	sub_category	str	containing the sub-category you want to give to the model or optimization, for example, "Dipole blade", "Low pass", "8 branches", among others
16	description	str	containing the description of the model, the setting function, additional information

Table 4.1: Arguments information of init_system method

Links of interest:

1. Link to PerSeO repository: https://github.com/ERA-2022/PerSeO
2. Link to PerSeO in PyPI: https://pypi.org/project/perseo-optimizer/