# 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
github.com/ERA-2022/PSO_for_hybrids_and_ antennas

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

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

# Contents

# 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 leverage HFSS (High Frequency Structure Simulator) simulations, supplying a dependable resource for assessing the performance of every alternative geometry. Additionally, they serve as a method to gather data from each optimization batch, enhancing the overall process.

The current version (v0.91) incorporates the following features:

- Import and use
- PSO optimization algorithm
- Log file: A comprehensive record of the optimization process can be found here
- Data collection in CSV files
- Provision of configuration data through JSON files or directly from the code
- Simulation control by ID
- Two demonstrative examples for applying the PSO to Hybrids and Antennas

**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 deploying the optimizer, certain prerequisites need to be fulfilled to ensure seamless operation. These requirements include installing Python along with necessary libraries, Ansys HFSS 2019, setting up the file paths appropriately, and preparing your custom Python script.

### 2.1.1 Python Version

The software requires Python version 3.10.4 to function correctly [1]. Other versions may cause compatibility issues with certain required packages for the optimizer.

### 2.1.2 Requirement.txt File

This file contains all the necessary Python libraries for the optimizer to function. These libraries include:

- cycler==0.11.0
- fonttools==4.31.2
- joblib==1.1.0
- kiwisolver==1.4.2
- matplotlib==3.5.1
- numpy==1.22.3
- packaging==21.3
- pandas==1.4.2
- Pillow==9.1.0
- pyparsing==3.0.7
- python-dateutil==2.8.2
- pytz==2022.1

[1] In the repository, under the "README.md" section, you will find a direct link to download this version.

- scikit-learn==1.1.1
- scipy==1.8.1
- six==1.16.0
- threadpoolctl==3.1.0

To install all these dependencies from the requirements.txt file, use your cmd interface:

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

### 2.1.3 Installation Notes and Warnings

If you use an IDE other than Visual Studio Code or the integrated Python IDE, ensure that these libraries are installed.

You may receive a warning message after installing the requirements.txt file indicating that some files could not be found in the Path. Do not be alarmed by this; it is merely a warning and can safely be ignored as the PSO will function regardless.

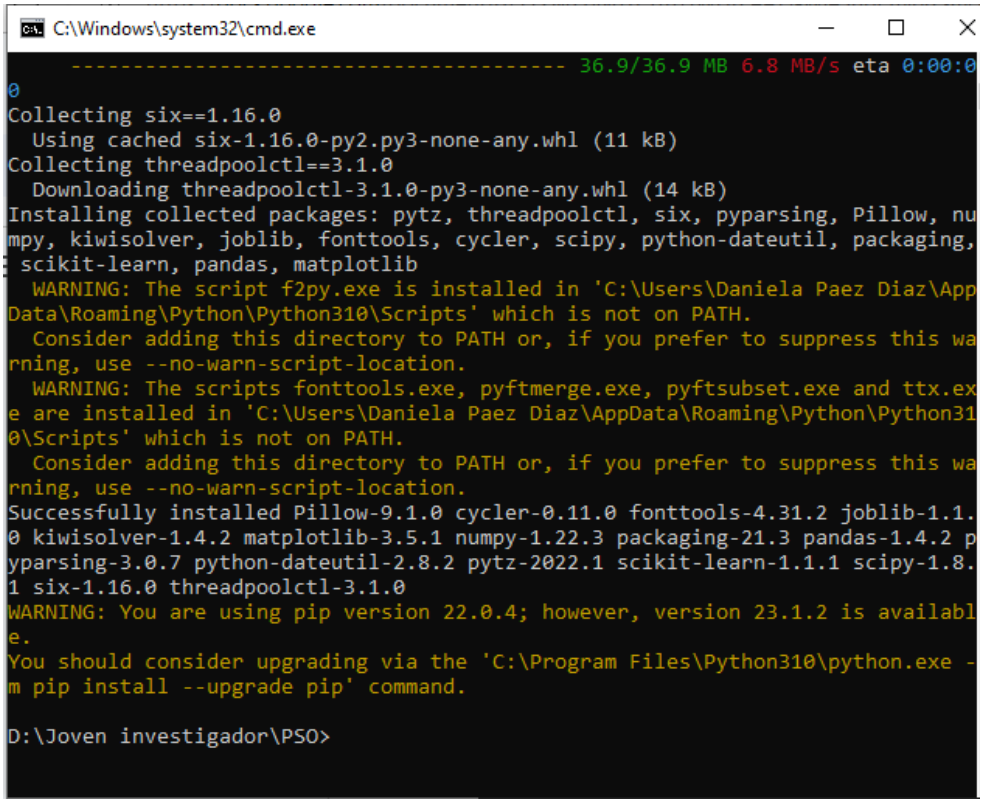


Figure 2.1: Warning message in the console.

### 2.1.4 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")
```

### 2.1.5 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 PSO_core import commands
from PSO_functions import Interfaz
```

- **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"\nNumero de areas: {len(areas_f)} || Mat.freq y Mat.db con mismo tama o: {len(areas_f) == len(areas_d)}\n")

    coeficiente = 0
    if len(areas_f)==0:
        coeficiente = 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")
            coeficiente =  ((80-bw)/80)**2

        else:
            coeficiente = 20

    print(f"Valor funci n de merito: {coeficiente}")
    print("-------------------------------------------------\n\n")
    return coeficiente
```

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.

  `Interfaz.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) 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> Optimizate
2> Fitness function test
3> Graphics tools
4> Exit
Enter an option:
```

### 3.1.1 Optimization

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 30 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, 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.

### 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 comparitions
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 comparitions
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 comparitions
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 comparitions
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 comparison
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 PSO package, 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 antenna 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 operating at 1.7GHz, a script illustrating the usage of PSO 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.

#### How to Use

1. Copy the main.py file and paste it at the root of your project, then initiate the system.
2. Next, copy the PATCH_ANTENNA.py file and paste it into the models folder located at the root of your project.
3. 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).

4. 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.
5. 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 then wait for the PSO to complete the optimization.

# 4. Annexes

| Position | Argument name | Data type | Description |
|---|---|---|---|
| 1 | ansys_exe | str | A string that saves the executable path of Ansys |
| 2 | ansys_save_def | str | A string that saves the default save path of Ansys |
| 3 | project_name | str | A string that saves the project's name |
| 4 | design_name | str | A string that saves the designs's name |
| 5 | variable_name | str | Variable name or name of the array where are the dimensions |
| 6 | units | str | A string that contains the units of dimensions (in the distance) of the design |
| 7 | max | lst | A list with maximum values that can take the PSO for the particles. These values must be numbers |
| 8 | min | lst | A list with minimum values that can take the PSO for the particles. These values must be numbers |
| 9 | nomilas | lst | A list with default values that can take the PSO for the particles. These values must be numbers |
| 10 | iterations | int | An integer that defines how many iterations will execute the code |
| 11 | particles | int | An integer that defines how many particles will be created |
| 12 | branches | int | An integer that defines how many branches have the hybrid |
| 13 | reports | dict | A dictionary containing the required reports will be used in the fitness function. Among these reports are:parameter Smn, parameter Zmn, Gain phi x angle, Amplitude imbalance, phase imbalance, and VSWR(x port) |
| 14 | category | str | A string that contains the category of design, for example, "Antenna", "Hybrid", "Filter", among others |
| 15 | sub_category | str | A string that contains the category of design, for example, "Dipole blade", "Low pass", "8 branches", among others. |
| 16 | description | str | A string with add/relevant information on design |

Table 4.1: Arguments information of init_system method