

1 Diseño antena

El estudiante debe realizar un notebook donde plantee el problema de optimización a partir del enunciado dado y encuentre la solución optima por medio de técnicas de búsqueda Heurística.

1.1 Descripción

Se requiere estimar los parámetros del diseño de una antena (la cual esta basada en cables doblados) que permita aumentar la ganancia en la frecuencia de resonancia seleccionada.

Table 1: Descripción de los problemas de optimización.

Item	3 segmentos
Función Objetivo	Ganancia máxima
Restricción	No salirse del cubo

Con la frecuencia de resonancia asignada, para el primer grupo se debe poner como restricción que el conjunto de cables doblados no supere el espacio ocupado por un cubo de $\lambda/2$, donde λ es la longitud de onda (no confundirlo con el λ empleado en multiplicadores de Lagrange), como se muestra en la figura 1. Para el segundo grupo, la restricción es que la antena no tenga una longitud mayor a 0.35 metros.

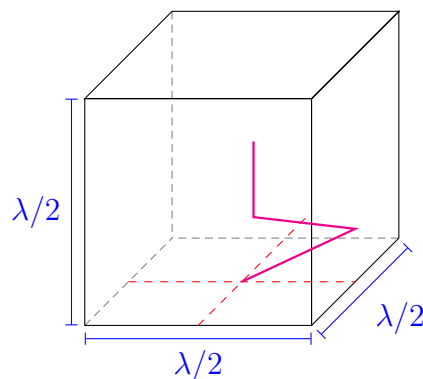


Figure 1: Esquema de diseño de la antena, con restricciones dentro del cubo. En rojo el plano de la tierra, y en magenta un ejemplo de una antena doblada en 3 segmentos.

A continuación se presenta un ejemplo en Python usando la libreria necpp [?].

1.2 Código en Python

```
from PyNEC import *

# Parameters
thickness = 0.002 #Grosor del alambre en metros
freq = 2500000000 #Frecuencia de resonancia de interes
numSegments = 2 #Numero de segmentos a doblar
waveLength = 299792458.0 / freq #calculo de la longitud de onda
tol = 1e-3 #Tolerancia entre segmentos
```

```
directionality = 2 #Seleccionar que funcion optimizar

def ObjFunc(params):
    #Params contiene los elementos x1,y1,z1,x2,y2,z2
    x1 = params[0]
    y1 = params[1]
    z1 = params[2]
    x2 = params[3]
    y2 = params[4]
    z2 = params[5]

    #creation of a nec context
    context=nec_context()
    #get the associated geometry
    geo = context.get_geometry()
    #Sometimes the design crashes
    try:
        #add wires to the geometry
        geo.wire(1, 15, 0.0, 0.0, 0.0, x1, y1, z1, thickness, 1.0, 1.0)
        geo.wire(2, 15, x1+tol, y1+tol, z1+tol, x2, y2, z2, thickness, 1.0, 1.0)
    #finish design
    context.geometry_complete(0)
    #Test signals set-up
    #check: https://pypi.org/project/necpp/
    #Define ground's location
    context.gn_card(1, 0, 0, 0, 0, 0, 0, 0)
    #Define excitation source parameters
    context.ex_card(0, 1, 1, 0, 1.0, 0, 0, 0, 0)
    #Define range of frequencies, this case only 2.5GHz
    context.fr_card(0, 1, freq / 1000000, 0.0)
    #Define radiation pattern parameters
    context.rp_card(0, 17, 45, 0, 5, 0, 0, 0, 0, 5, 8, 0, 0)

    # Objective function output
    if (directionality == 0):
        gain = context.get_gain_max(0)
    elif (directionality==1):
        gain = context.get_gain_min(0)
    else:
        gain = context.get_gain_mean(0)
    except: #If simulation crashes then assign -999 to gain value.
        gain = -999.0
    return gain
```

Procedimiento

1. Descripción matemática y programación del problema de optimización y función objetivo.

2. Descripción matemática del algoritmo Heurístico asignado, como hace exploración y explotación. Recordar que las longitudes de los cables son positivas y no se pueden salir del cubo. El algoritmo se debe diseñar con una clase de Python, con inicialización de los parámetros del algoritmo (límites, cantidad de iteraciones); con método que se encargue del proceso de optimización, y adicionalmente atributos que almacenen el desempeño de algoritmo (mejor solución por iteración).
3. Realizar el proceso de optimización 10 veces para 100 iteraciones y almacenar los resultados en una matriz.
4. Analizar la convergencia del algoritmo graficando las 10 curvas de la función objetivo vs el número de iteraciones. Graficar adicionalmente la media de las 10 curvas.
5. Comparar el desempeño del algoritmo contra el algoritmo de Enjambre de Partículas (PSO) visto en clase.

2 Informe

Desarrollar un notebook en Python, que incluya las siguientes secciones:

1. Introducción al problema.
2. Código y desarrollo de la solución.
3. Análisis de resultados.
4. Conclusiones.
5. Bibliografía.

Referencias

Iglesias, A.; Gálvez, A.; Suárez, P.; Shinya, M.; Yoshida, N.; Otero, C.; Manchado, C.; Gomez-Jauregui, V. Cuckoo Search Algorithm with Lévy Flights for Global-Support Parametric Surface Approximation in Reverse Engineering. *Symmetry* 2018, 10, 58. <https://doi.org/10.3390/sym10030058>