INFORME DE INVESTIGACIÓN

Softwares de Simulación para Tecnología Electrónica y otras cátedras de Ingeniera.

Compatibilidad Electromagnética

Gonzalo Nahuel Vaca

Departamento de Ingeniería e Investigaciones Tecnológicas
Universidad Nacional de la Matanza
Argentina
9 de agosto de 2022

1. Introducción

Este documento se propone detallar todos los aspectos referidos a la evaluación de programas para el cálculo de compatibilidad electromagnética.

2. OpenEMS

En esta sección se encuentran los detalles de la evaluación del programa OpenEMS.

2.1. Instalación

A continuación se detallan los pasos a seguir para tener el programa instalado en el ordenador.

1. Satisfacer las dependencias del programa, a continuación se muestran los comandos para el sistema Ubuntu 22.04 LTS:

   - Dependencias mínimas obligatorias:
     `sudo apt-get install`
     `build-essential cmake git libhdf5-dev libvtk7-dev`
     `libboost-all-dev libcgal-dev libtinyxml-dev`
     `qtbase5-dev libvtk7-qt-dev`
   - GNU Octave (equivalente GNU de Matlab, opcional):
     `sudo apt-get install octave liboctave-dev`
   - Interfaz Python (opcional):
     `sudo pip3 install matplotlib cython h5py`

2. Clonar e instalar

   - Clonar el repositorio y sus submódulos:
     `git clone --recursive`
     `https://github.com/thliebig/openEMS-Project.git`
   - Dentro del repositorio ejecutar:

     ```
     ./update_openEMS.sh ~/opt/openEMS --python
     ```

2.2. Verificación

La prueba más simple para verificar la instalación es ejecutar el binario de openEMS. Para lograrlo debe navegar hasta la carpeta de la instalación, la cuál normalmente es `opt/openEMS/bin` dentro de su usuario; luego se debe ejecutar el comando `./openEMS` y deberá observar una salida como la siguiente:

```
------------------------------------------------------
| openEMS 64bit -- version v0.0.35-76-gd4448fa
| (C) 2010-2018 Thorsten Liebig <thorsten.liebig@gmx.de>
| GPL license
------------------------------------------------------
Used external libraries:
CSXCAD -- Version: v0.6.2-109-gcd9decb
hdf5   -- Version: 1.10.7
compiled against: HDF5 library version: 1.10.7
tinyxml -- compiled against: 2.6.2
fparser
boost  -- compiled against: 1_74
vtk -- Version: 7.1.1
compiled against: 7.1.1
```

A continuación se detallan los pasos a seguir para verificar la correcta instalación de openEMS con GNU Octave.

Como se muestra en la figura 1 se debe colocar el *path* de openEMS en la ventana de comandos.

Los comandos para agregar los *path* son:

```
addpath('~/opt/openEMS/share/openEMS/matlab');
addpath('~/opt/openEMS/share/CSXCAD/matlab');
```

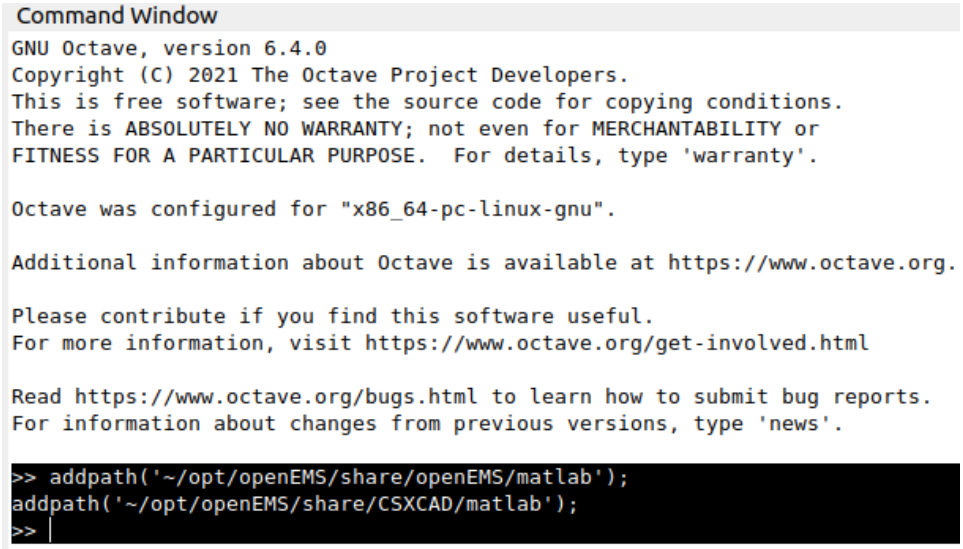

Figura 1: Configuración de *path* en GNU Octave.

El siguiente paso es verificar la interfaz CXCAD, a continuación se muestra el comando y respuesta en la ventana de comandos de GNU Octave:

```
>> InitCSX
ans =
Properties: []
```

Con la interfaz inicializada se puede realizar una verificación de sus funciones como se muestra a continuación:

```
>> InitFDTD('NrTS', 0, 'EndCriteria', 0)
ans =
    ATTRIBUTE: [1x1 struct]
```

El siguiente paso es verificar el correcto funcionamiento del simulador como se muestra a continuación:

```
>> RunOpenEMS( '.', 'nonexistant.xml', '' )
[...]
Read openEMS xml file: nonexistant.xml ...
openEMS: Error File-Loading failed!!! File: nonexistant.xml
```

El error que se muestra significa que el simulador funciona correctamente ya que su argumento es un archivo inexistente. Sin embargo, el binario del simulador se invocó sin problemas.

Es importante verificar que el programa de visualización de definiciones de geometría funcione de forma correcta, esto se logra con el siguiente comando:

```
CSXGeomPlot('nonexistant.xml')
```

La ventana de comandos debe arrojar un error ya que el archivo de geometría no existe, sin embargo se debe ejecutar el visualizador como se muestra en la figura 2.

Llegado a este punto se puede concluir que openEMS fue instalado de forma correcta.

Se recomienda instalar el programa ParaView para tener una herramienta para visualizar las simulaciones. Los ejemplos de este documento utilizan ParaView para representar de forma gráfica los resultados.

2.3. Ejemplos de uso

2.3.1. Guía de onda superficial

Las instrucciones para generar la geometría, su excitación y la definición de las variables a observar son las siguientes:

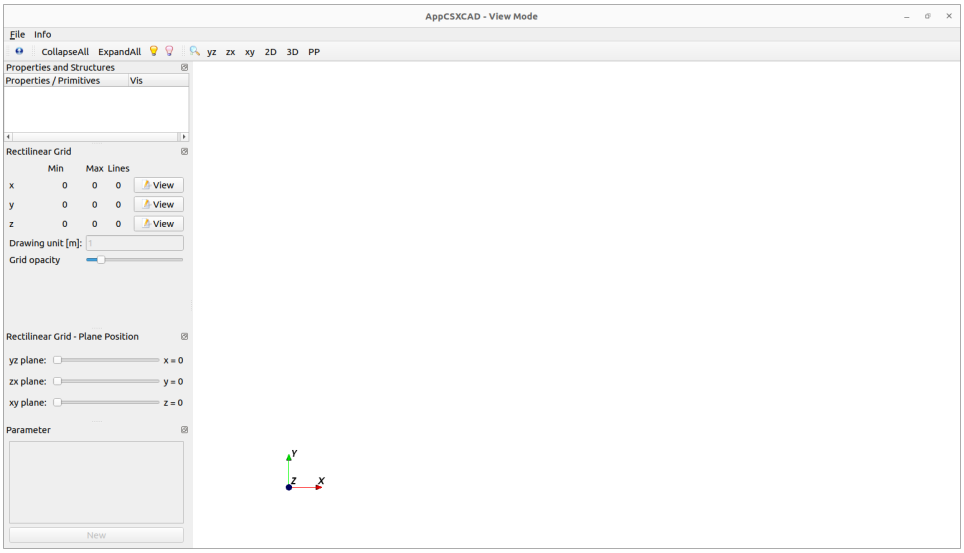

Figura 2: Programa de visualización de geometrías.

```
close all
clear
clc
FDTD = InitFDTD('NrTS',100, 'EndCriteria',0, 'OverSampling',50);
FDTD = SetSinusExcite(FDTD,10e6);
FDTD = SetBoundaryCond(FDTD,{'PMC' 'PMC' 'PEC' 'PEC' 'MUR' 'MUR'});
CSX = InitCSX();
mesh.x = -10:10;
mesh.y = -10:10;
mesh.z = -10:30;
CSX = DefineRectGrid(CSX, 1, mesh);
CSX = AddExcitation(CSX,'excitation',0,[0 1 0]);
CSX = AddBox(CSX,'excitation',0,[-10 -10 0],[10 10 0]);
CSX = AddDump(CSX,'Et');
CSX = AddBox(CSX,'Et',0,[-10 0 -10],[10 0 30]);
mkdir('tmp');
WriteOpenEMS('tmp/tmp.xml',FDTD,CSX);
```

Antes de lanzar la simulación se procede a verificar la geometría generada con los siguientes comandos:

```
CSXGeomPlot( 'tmp/tmp.xml' );
```

En la figura 3 se puede observar el plano que simula una guía de onda.

Una vez satisfecho con la geometría, se procede a lanzar la simulación. Para eso, primero debe cerrar el visualizador de geometría y escribir el siguiente comando:

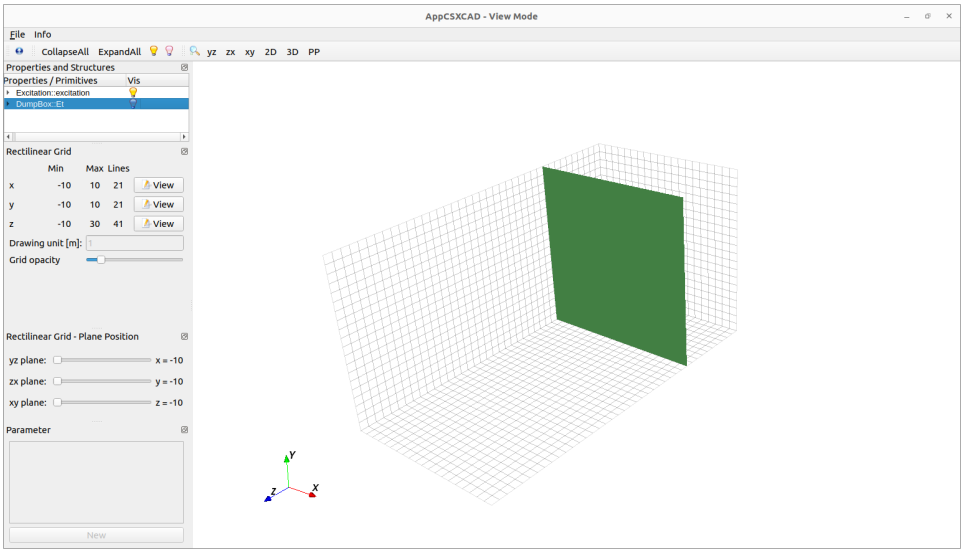

Figura 3: Geometría del ejemplo 1.

```
RunOpenEMS('tmp','tmp.xml','');
```

Para visualizar la simulación se puede usar el programa ParaView. Para eso se debe abrir el archivo `Et..vtr`, luego debe ingresar al inspector de objetos y seleccionar que el color se asocie al *E-Field* además de indicar el eje *Y*. Seguidamente, puede hacer click en el boton de *play*. Se verá una animación con la variación del potencial eléctrico en el tiempo, sin embargo los colores se encuentran sobre el plano y no se puede apreciar con facilidad la onda.

Para mejorar la representación se debe colocar el filtro *warp-by-vector* que deformará la representación según la intensidad del campo, esto genera una representación en donde se puede observar con facilidad la onda en la guía.

En la figura 4 se puede observar la guía de onda simulada en un plano.

2.3.2. Esfera metálica dispersa un frente de onda

En este ejemplo se simula la dispersión que produce una esfera metálica cuando una onda incide sobre ella. Para tal fin se generó una geometría esférica con la propiedad física de ser un conductor eléctrico. Se la puede ver en la figura 5.

Luego se definió un cubo como volumen de simulación lo suficientemente grande para que el cálculo numérico sea lo más preciso posible pero que los tiempos de cómputo sean aceptables. El volumen se puede observar en la figura 6.

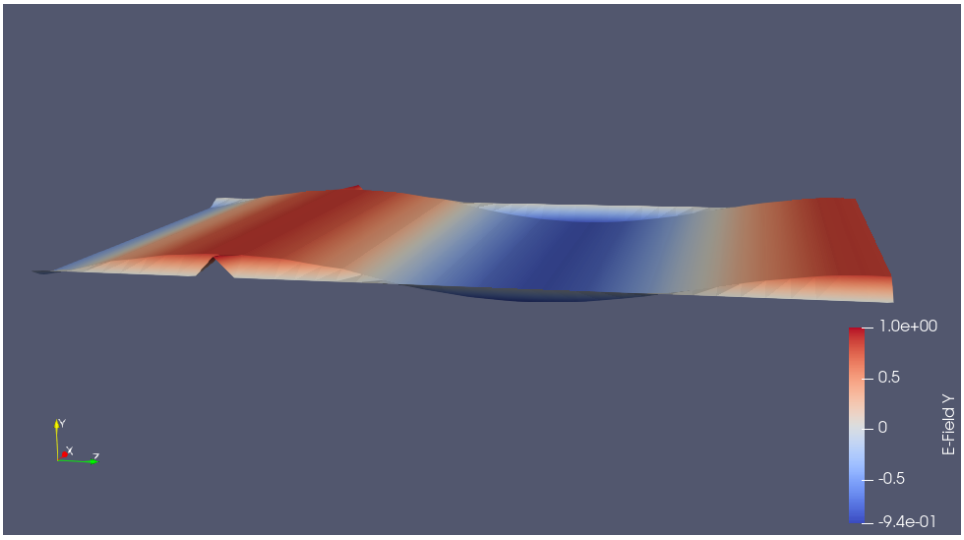

Figura 4: Simulación del ejemplo 1.

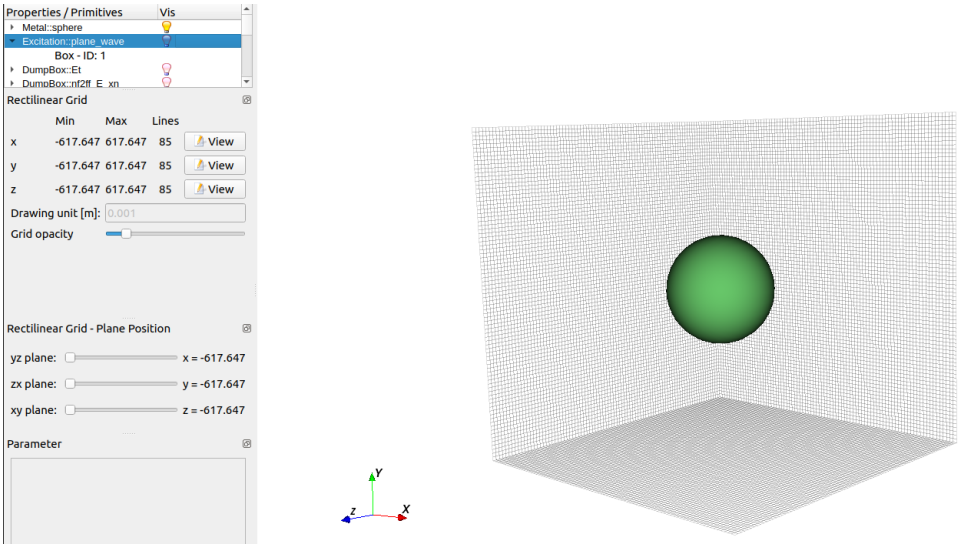

Figura 5: Esfera metálica del ejemplo 2.

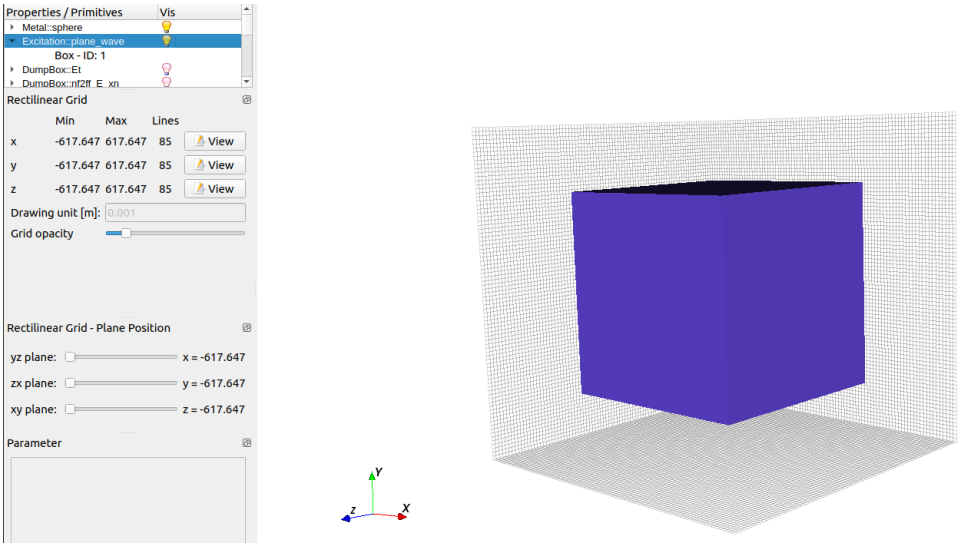

Figura 6: Volumen de simulación del ejemplo 2.

Luego se corrió la simulación y se la visualizó con ParaView. Se tomaron capturas en 3 momentos distintos y se los muestran en 2 y 3 dimensiones. Se pueden observar en las figuras 7,8,9,10,11 y 12.

A continuación se describe el código para generar el ejemplo:

```
% Si es la primera ejecución de octave de la sesión de trabajo
% agregar openEMS al path, sino physical_constants no va a estar
% definido
addpath('~/opt/openEMS/share/openEMS/matlab');
addpath('~/opt/openEMS/share/CSXCAD/matlab');

close all
clear
clc

%% Se configura la simulación
physical_constants;
unit = 1e-3; % longitudes en mm

sphere.rad = 200;

inc_angle = 0 /180*pi; % ángulo incidente (x-axis) en rad

% Tamaño de la caja de simulación
SimBox = 1000;
PW_Box = 750;
```

```
%% configuración FDTD parámetros y función de excitación
f_start =  50e6; % frecuencia de inicio
f_stop = 1000e6; % frecuencia de fin
f0 = 500e6;

FDTD = InitFDTD( );
FDTD = SetGaussExcite( FDTD, 0.5*(f_start+f_stop), 0.5*(f_stop-f_start) );
BC = [1 1 1 1 1 1]*3;  % set boundary conditions
FDTD = SetBoundaryCond( FDTD, BC );

%% configurar CSXCAD geometría y malla
max_res = c0 / f_stop / unit / 20; % cell size: lambda/20
CSX = InitCSX();

% creación de malla
smooth_mesh = SmoothMeshLines([0 SimBox/2], max_res);
mesh.x = unique([-smooth_mesh smooth_mesh]);
mesh.y = mesh.x;
mesh.z = mesh.x;

%% creación de esfera de metal
CSX = AddMetal( CSX, 'sphere' ); % se crea un conductor perfecto (PEC)
CSX = AddSphere(CSX,'sphere',10,[0 0 0],sphere.rad);

%% excitación de onda planar
k_dir = [cos(inc_angle) sin(inc_angle) 0]; % dirección de la onda planar
E_dir = [0 0 1]; % polarización de la onda planar --> E_z

CSX = AddPlaneWaveExcite(CSX, 'plane_wave', k_dir, E_dir, f0);
start = [-PW_Box/2 -PW_Box/2 -PW_Box/2];
stop  = -start;
CSX = AddBox(CSX, 'plane_wave', 0, start, stop);

CSX = AddDump(CSX, 'Et');
start = [mesh.x(1)   mesh.y(1)   0];
stop  = [mesh.x(end) mesh.y(end) 0];
CSX = AddBox(CSX, 'Et', 0, start, stop);

%% cálculo nf2ff
start = [mesh.x(1)     mesh.y(1)     mesh.z(1)];
stop  = [mesh.x(end) mesh.y(end) mesh.z(end)];
[CSX nf2ff] = CreateNF2FFBox(CSX, 'nf2ff', start, stop);
```

```
mesh = AddPML(mesh,8);

CSX = DefineRectGrid( CSX, unit, mesh );

%% carpeta de simulación
Sim_Path = 'Sphere_RCS';
Sim_CSX = 'Sphere_RCS.xml';

[status, message, messageid] = rmdir( Sim_Path, 's' );
[status, message, messageid] = mkdir( Sim_Path );

WriteOpenEMS( [Sim_Path '/' Sim_CSX], FDTD, CSX );

%% Se muestra la geometría
CSXGeomPlot( [Sim_Path '/' Sim_CSX] );

%% se corre la simulación
RunOpenEMS( Sim_Path, Sim_CSX);
```

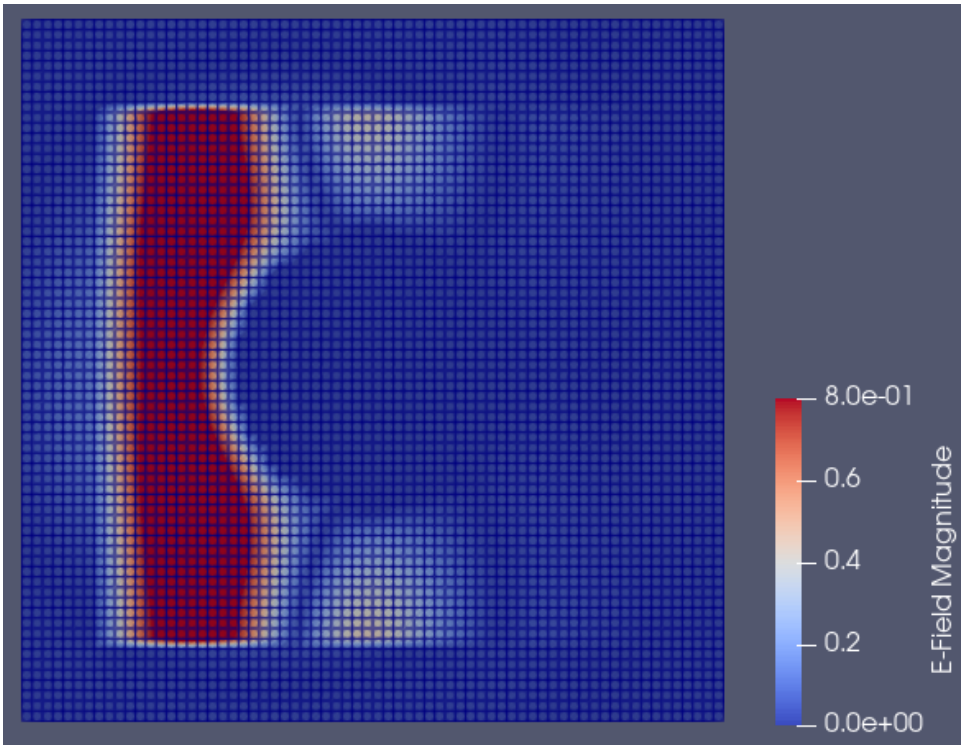

Figura 7: Simulación momento 1.

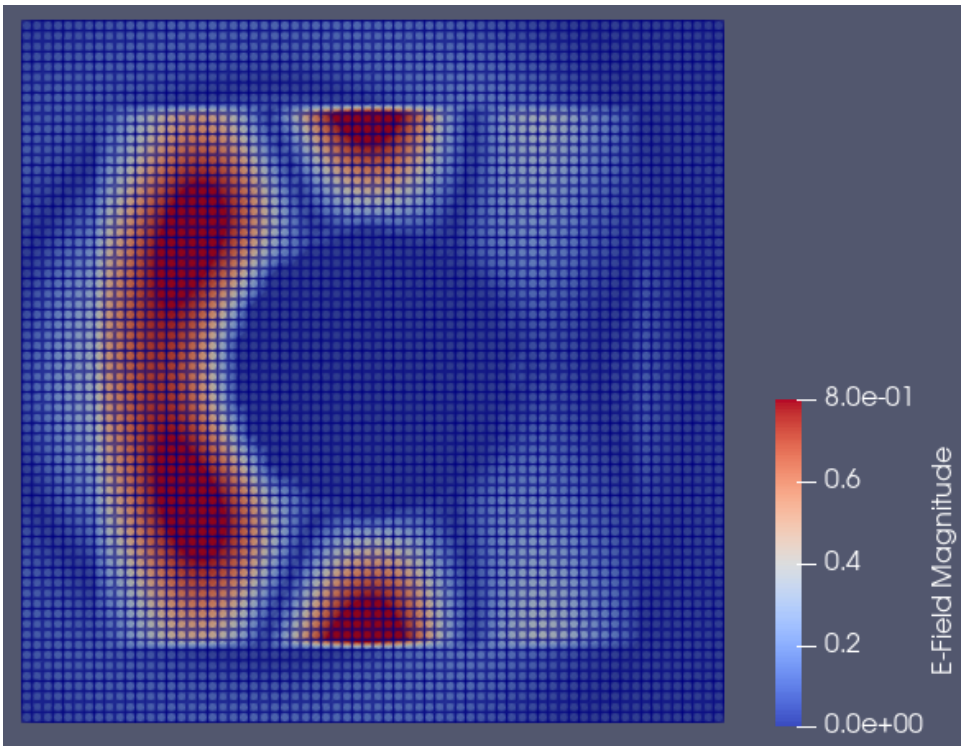

Figura 8: Simulación momento 2.

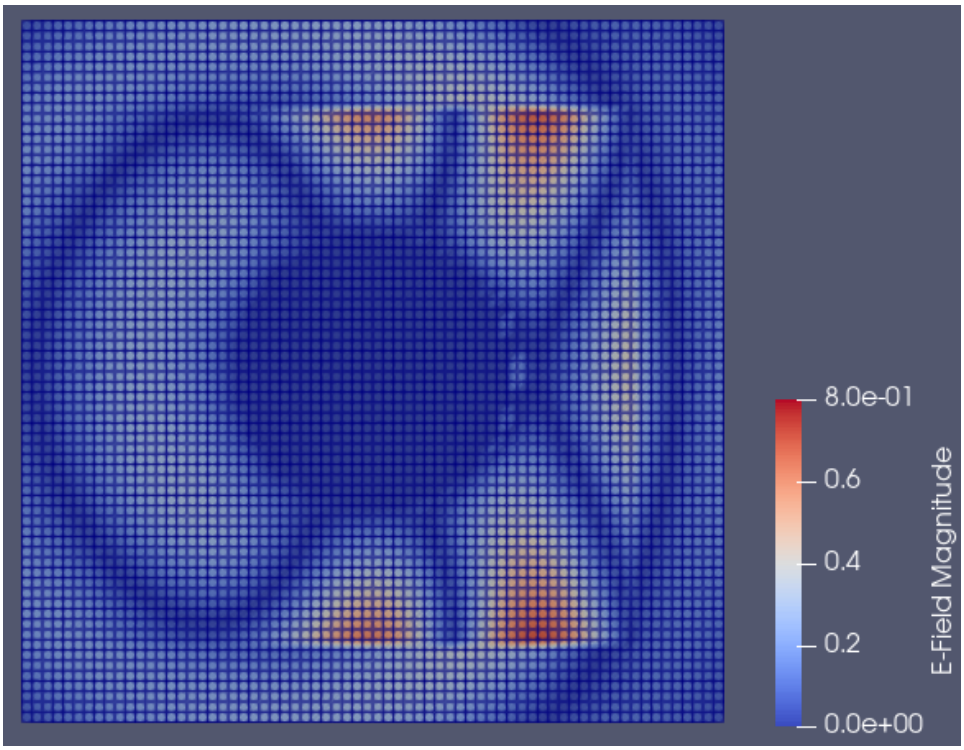

Figura 9: Simulación momento 3.

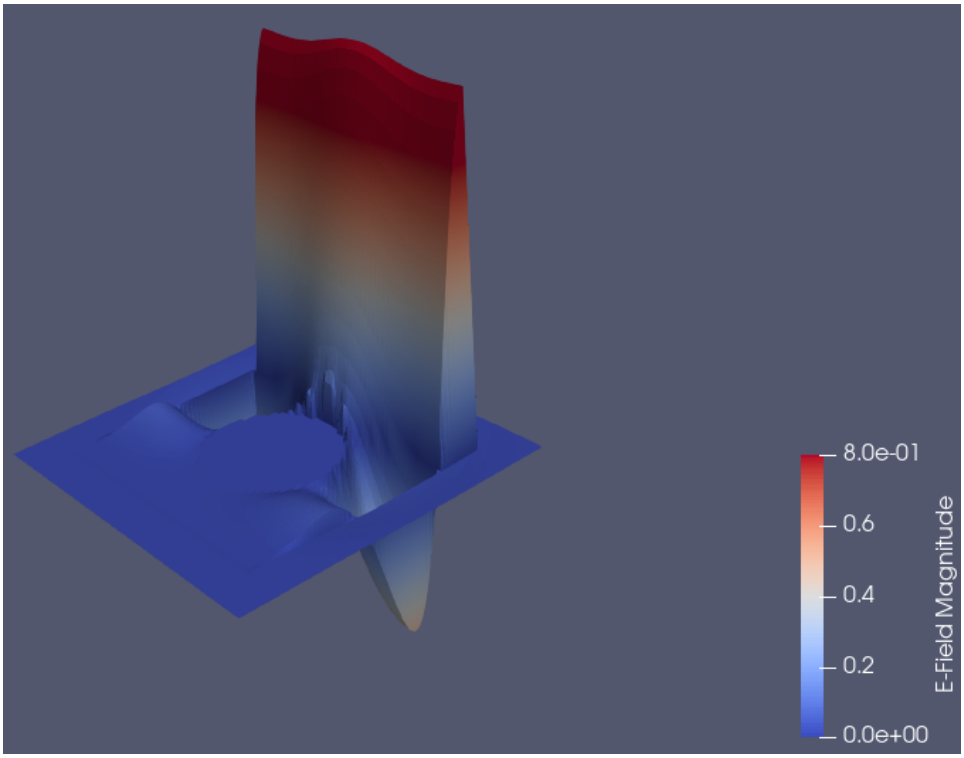

Figura 10: Simulación momento 1 en 3D.

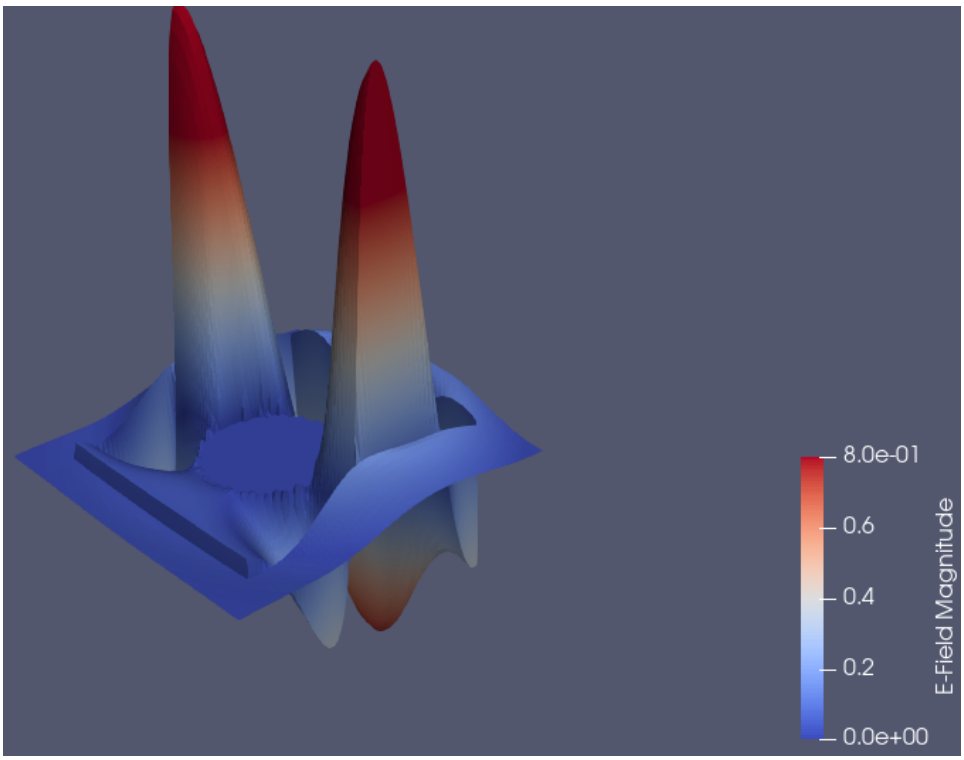

Figura 11: Simulación momento 2 en 3D.

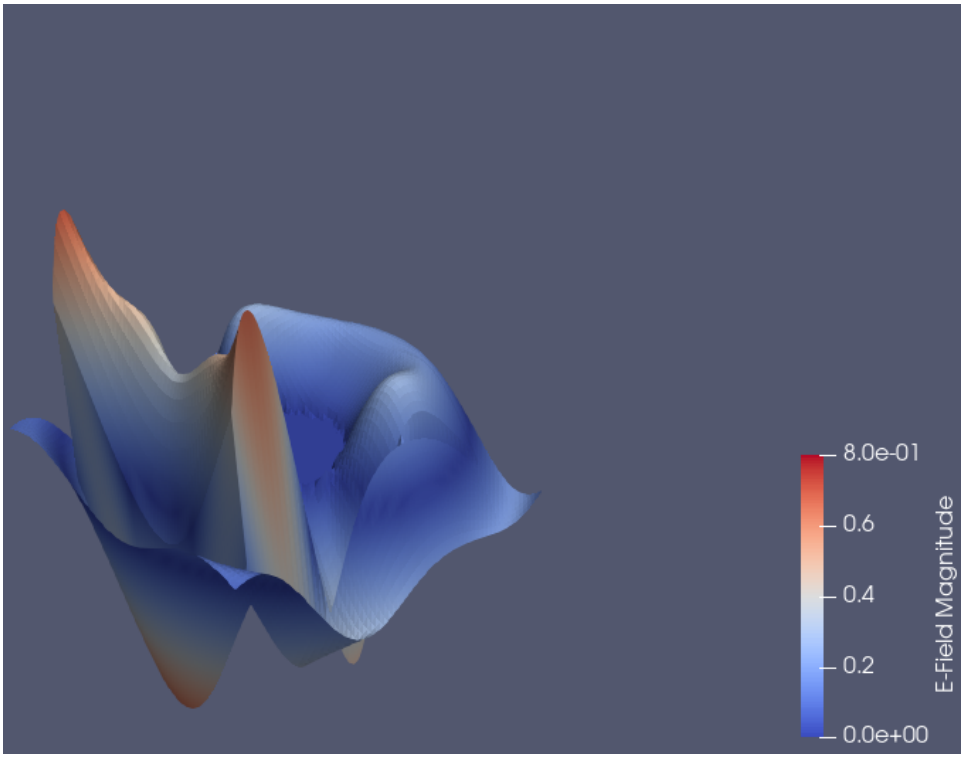

Figura 12: Simulación momento 3 en 3D.

2.4. Primeros intentos de simular circuitos impresos

OpenEMS utiliza un *parser* para transformar los archivos *HyperLynx* que exportan programas como KiCad e Eagle en un formato que el motor de matlab pueda entender. Sin embargo, el *parser* hyp2mat es un programa cuyo único autor dió por finalizado y quedaron deprecadas las recetas de Autotools y las bibliotecas que utiliza para *linkear*.

Existen tres posibilidades en este punto:

- Crear un dockerfile que genere una imagen con Octave y OpenEMS en base a una versión de Ubuntu que soporte la última versión de hyp2mat que se publicó.
- Utilizar un método alternativo para importar los circuitos impresos.
- Abandonar OpenEMS como herramienta viable.

2.5. Montaje de GNU/Octave y OpenEMS en docker

Se pudo crear una serie de recetas que generan una imagen y contenedor con Octave y OpenEMS.

Dockerfile:

```
# Universidad Nacional de la Matanza.

# Los Dockerfiles comienzan a partir de una imagen base
FROM ubuntu:focal

# En esta sección se actualiza la base de datos del gestor
# de paquetes (apt) y se procede a instalar las herramientas
# y sus dependencias
RUN apt-get update &&  DEBIAN_FRONTEND=noninteractive apt-get install -y -q \
  octave-common \
  openems \
  octave-openems \
  libcsxcad0

# Se genera un usuario de desarrollo (dev) con permisos de
# administrador y sin necesidad de ingresar su contraseña
RUN adduser --disabled-password --gecos '' dev && \
  usermod -aG sudo dev && \
  echo "dev ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers

# Se hace /bin/sh symlink a bash en vez de dash:
RUN echo "dash dash/sh boolean false" | debconf-set-selections
RUN DEBIAN_FRONTEND=noninteractive dpkg-reconfigure dash
```

```
# Cuando se ingrese al contenedor se ingresara como dev
USER dev
ENV HOME /home/dev
ENV LANG en_US.UTF-8

# Se crea una carpeta de trabajo y se la configura como
# workdir
RUN mkdir /home/dev/workspace
WORKDIR /home/dev/workspace
```

Receta de creación:

```
#!/bin/bash
docker build \
    -t openems .
```

Receta de lanzamiento del contenedor:

```
#!/bin/bash

# El comentario que comienza con un signo de exclamacion
# tiene la finalidad de indicarle al sistema con que binario
# debe ejecutar este archivo, en nuestro caso el intérprete de
# BASH

# Se define el directorio del host como "path to working directory"
# (PWD), esto es, la carpeta desde donde lanzamos este script.
DIRECTORIO_DE_MI_MAQUINA=PWD

docker run -ti \
    -e DISPLAY=$DISPLAY \
    --net="host" \
    --name="UNLaM" \
    --hostname="UNLAM" \
    -v $DIRECTORIO_DE_MI_MAQUINA:/home/dev/workspace \
    openems
    /bin/bash
```

```
Step 9/10 : RUN mkdir /home/dev/workspace
 ---> Running in ff216454d37b
Removing intermediate container ff216454d37b
 ---> e2e99e68430f
Step 10/10 : WORKDIR /home/dev/workspace
 ---> Running in b9d1405df162
Removing intermediate container b9d1405df162
 ---> 2ca4330c923b
Successfully built 2ca4330c923b
Successfully tagged openems:latest
```

Figura 13: Construcción de imagen

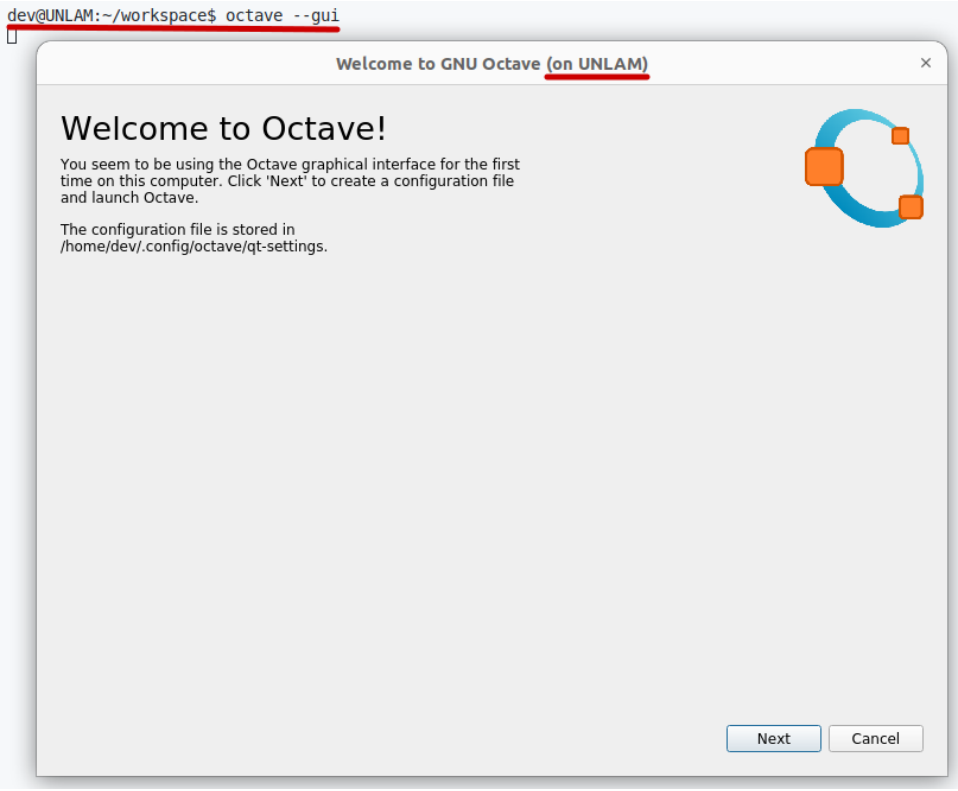

Figura 14: Ejecución del contenedor