Pérdidas_Microstrip

January 22, 2020

1 Pérdidas óhmicas y dieléctricas en Microstrip para PCBs.

1.0.1 Author: Sonia Gómez Gijón email

Version 0.2

Date, 18/11/2019

1.0.2 Introducción. Fundamento Teórico

Este programa calcula el efecto pelicular, la dispersión y las pérdidas del dieléctrico en función de la frecuencia en la atenuación de una onda cuadrada en una línea microstrip de longitud L_o.

La impedancia característica de la linea de transmisión viene dada por cociente de voltaje y corriente en la línea. Para una línea uniforme, infinita con propiedades eléctricas distribuidads la ecuación es

$$Z_o\,(f) = \sqrt{\frac{R(f) + j \cdot 2 \cdot \pi \cdot f \cdot L}{G(f) + j \cdot 2 \cdot \pi \cdot f \cdot C}} \tag{1}$$

Donde R(f) es la resistencia por unidad de longitud, L es la inductancia por unidad de longitud, G(f) es la conductancia por unidad de longitud y C es la capacitancia por unidad de longitud. El factor de propagación de la señal es $\Gamma(f)$:

$$\Gamma(f) = \sqrt{(R(f) + j \cdot 2 \cdot \pi \cdot L) \cdot (G(f) \cdot j \cdot 2 \cdot \pi \cdot f \cdot C)} \tag{2}$$

El voltaje de la señal se propaga en la dirección Z de acuerto con la siguiente ecuación:

$$V(f, z, t) = e^{Re[\Gamma(f)] \cdot (z/v - t)} \cdot e^{Im[\Gamma(f)] \cdot (z/v - t)} \tag{3}$$

Aquí vemos que la atenuación de la señal está dada por la parte real de $\Gamma(f)$ y la propagación está determinada por la parte imaginaria de $\Gamma(f)$

Parámetros generales
Resistividad del cobre:

$$\rho = 6.585 \cdot 10^{-7} \Omega \cdot in \tag{4}$$

Constante dieléctrica del vacío:

$$\varepsilon_0 = 8.8542 \cdot 10^{12} \cdot \frac{farad}{m} \tag{5}$$

Permeabilidad magnética del vacío:

$$\mu_0 = 4 \cdot \pi \cdot 10^{-7} kg \cdot m \cdot could^{-2} \tag{6}$$

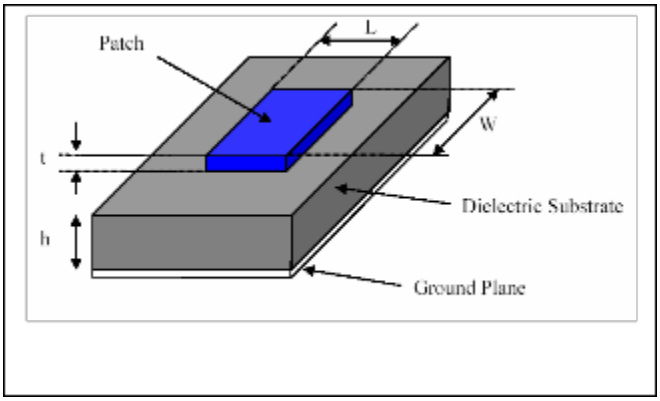
1.0.3 Parámetros de la PCB

Estos parámetros son para NELCO 6000-SI

Parámetros Microstrip:
w:=0.012·in

t:= 0.001·in

h:= 0.006·in



1.0.4 Resolución

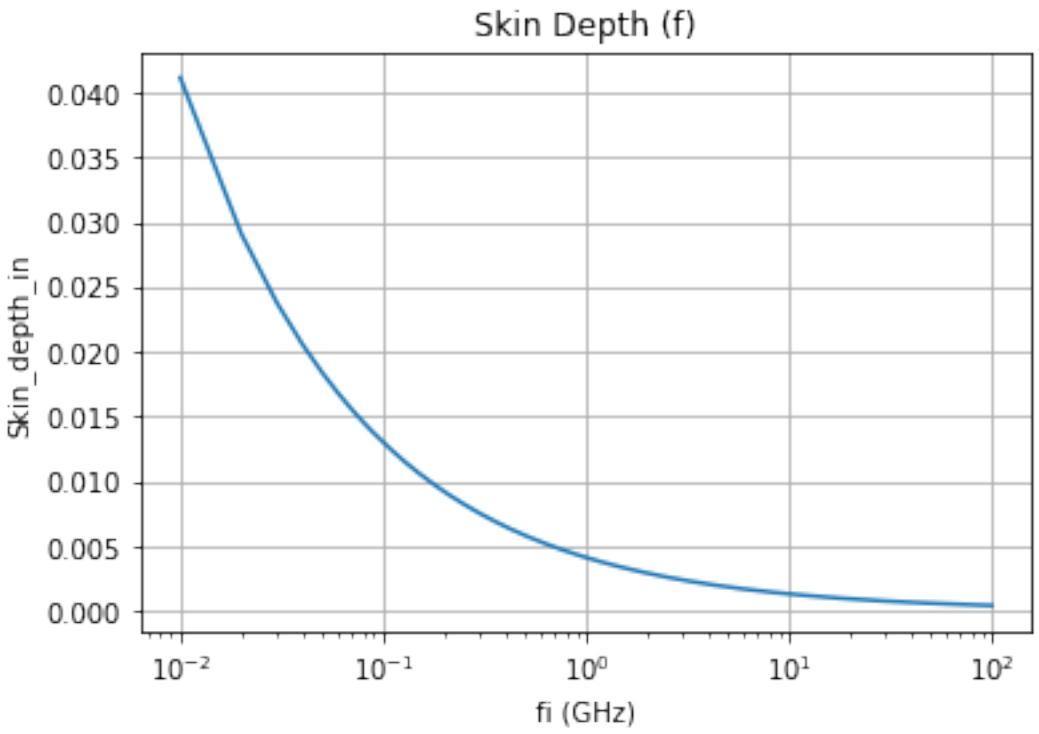
[84]:
```
%matplotlib inline
import matplotlib.pylab as mp
import numpy as np
from numpy import sqrt
from matplotlib.pylab import plot
from numpy import pi
```

[85]:
```
#Defininmos variables
from scipy.constants import mu_0          # Definida en m
from scipy.constants import epsilon_0     # Definida en m

mu_in=mu_0*39.3701
mu_mm=mu_0/1000

epsilon_in=epsilon_0/39.3701

c_m=1/sqrt(epsilon_0*mu_0)
c_in=c_m*39.3701
```

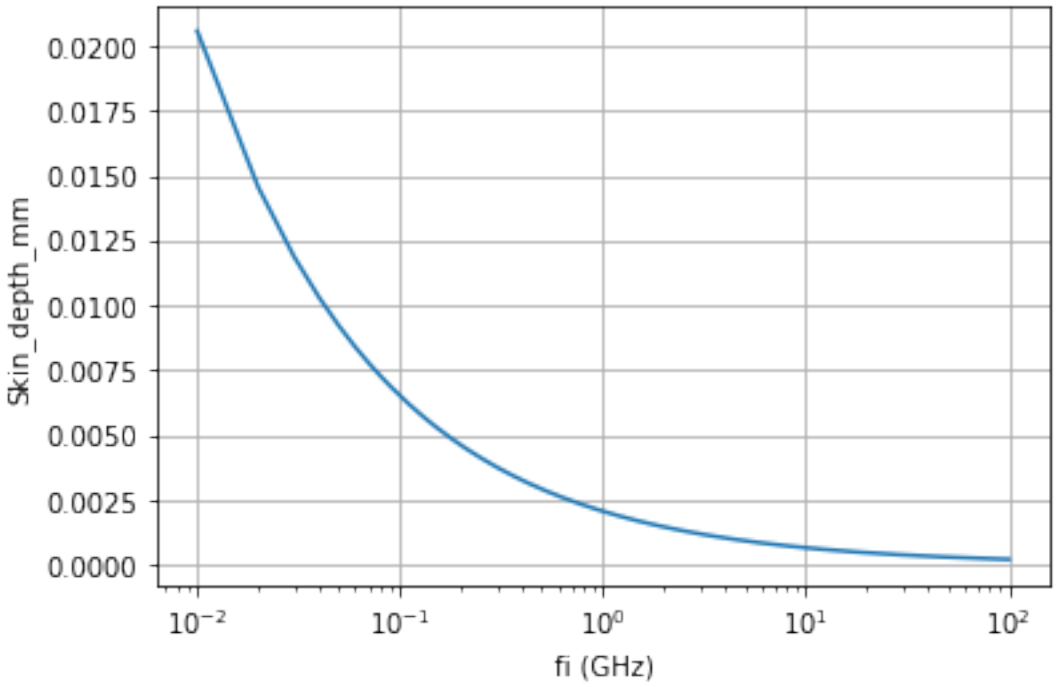
```
rho_in=6.585e-7
rho_mm=25.4*rho_in
rho_m=rho_mm/1000

#Vector frecuencias
i=np.arange(1,10001,1)
fi=i*10e6

#Definimos los parémtros para la pcb FR-4
w_in=0.01
w_mm=25.4*w_in
w_m=w_mm/1000

t_in=0.001
t_mm=25.4*t_in
t_m=t_mm/1000

h_in=0.006
h_mm=25.4*h_in
h_m=h_mm/1000

tan_delta=0.022+0.0075*fi/1e9
er=4.4
L0_in=8
L0_m=L0_in/39.37
```

[86]:
```
#Creamos el vector frecuencias que va de 1 a 10000000 Hz
skin_depth=sqrt((2*rho_in)/(2*pi*fi*mu_in))
skin_depth=skin_depth/(t_in/2)
skin_depth_mm=sqrt((2*rho_mm)/(2*pi*fi*mu_mm))
```

[87]:
```
mp.plot(fi/1e9,skin_depth)
mp.xscale("log")
mp.title('Skin Depth (f)')
mp.xlabel('fi (GHz)')
mp.ylabel('Skin_depth_in')
mp.grid()
```



[88]:
```
mp.plot(fi/1e9,skin_depth_mm)
mp.xscale("log")
mp.xlabel('fi (GHz)')
mp.ylabel('Skin_depth_mm')
mp.grid()
```



Tomamos los siguientes valores Io=0.1mA y f=1GHz

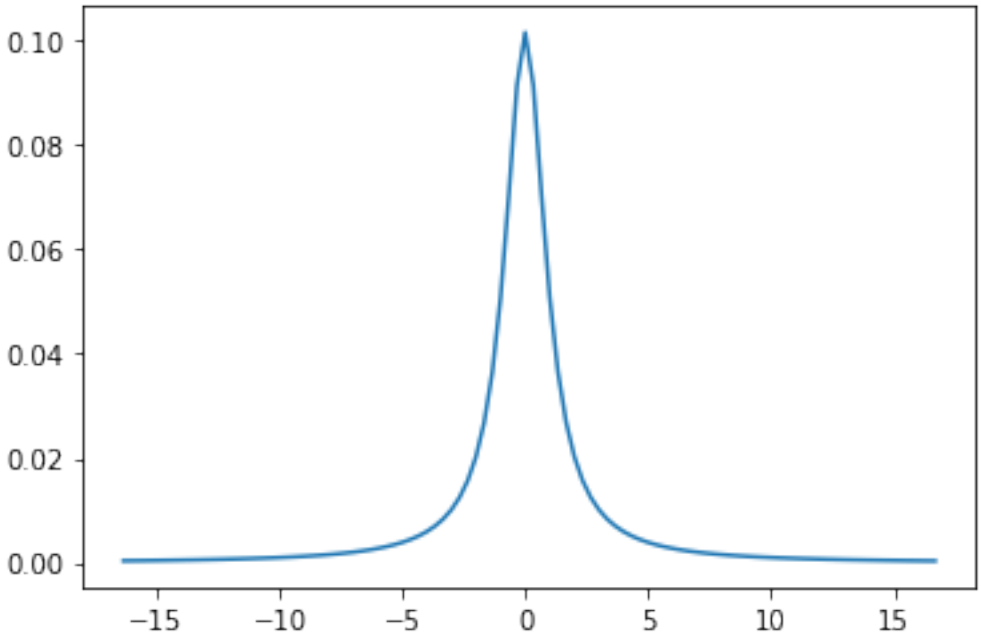
La corriente en función del efecto pelicular se puede calcular con la siguiente expresión:

$$I_x = \frac{Io}{\pi \cdot h \cdot \delta(f) \cdot [1 + (\frac{x}{h})^2]} \tag{7}$$

A una frecuencia de 1 GHz tendremos los siguientes valores de efecto pelicular

[89]:
```
# Este calculo en mm
fo=1e9
Io=0.1e-3
delta_mm=sqrt((2*rho_mm)/(2*pi*fo*mu_mm))
delta_m=sqrt((2*rho_m)/(2*pi*fo*mu_0))

i=np.arange(1,101,1)
x_mm=(i-50)*h_mm/3
x_m=(i-50)*h_m/3
x_in=(i-50)*h_in/3

I=Io/(np.pi*h_mm*delta_mm*(1+np.power((x_mm/h_mm),2))) #0.00015424
eje_x=x_mm/h_mm
mp.plot(eje_x,I)
```

[89]:
```
[<matplotlib.lines.Line2D at 0x151e790a10>]
```



La corriente arriba es Io, I(x) es la corriente de vuelta (la que va por el plano de masa) Queremos calcular el ancho del plano inferior que ocupa la corriente, cuando la corriente de abajo es igual al 95% de la corriente de arriba.

$$\int_{-a\cdot h}^{a\cdot h} I(x) - 0.95 I_0 = 0 \tag{8}$$

$$\int_{-a\cdot h}^{a\cdot h} \frac{Io}{\pi \cdot h \cdot \delta(f) \cdot [1 + (\frac{x}{h})^2]} - 0.95 I_0 = 0 \tag{9}$$

Provisionalmente para poder seguir a=12.7

[90]:
```
import sympy as sp
import scipy.integrate as integrate
from numpy import power
from sympy import cos,pi
from sympy import simplify
from sympy import *

x_m,a=symbols('x_m,a')
integral=integrate(Io/(np.pi*h_m*delta_m*(1+(x_m/h_m)**2)),(x_m,-a*h_m,a*h_m))
integral

a=list(solveset(delta_m*integral-0.95*Io,a,domain=S.Reals))
a=float(a[0])
a
```

[90]: 12.706204736174731

La resistencia en DC está dada por la siguiente expresión:

$$R_{DC} = \frac{\rho}{w \cdot t} + \frac{\rho}{2 \cdot a \cdot h \cdot t} \tag{10}$$

[91]:
```
# Todo en las mismas unidades (Hay que pasas t a pulgadas)
#a=12.7062047361747
Rdc_in=(rho_in/(w_in*t_in))+(rho_in/(2*a*h_in*t_in))
Rdc_m=(rho_m/(w_m*t_m))+(rho_m/(2*a*h_m*t_m))
Rdc_in
```

[91]: 0.07016875616200093

The RF skin resistance is given by:

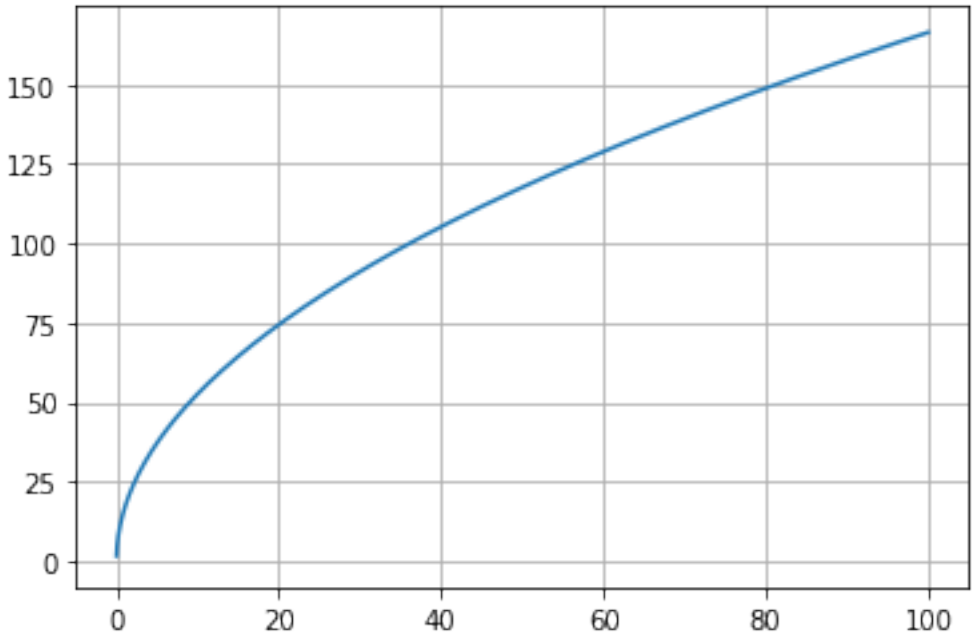
$$R_{skin}(f) = \frac{\sqrt{\rho \cdot \pi \cdot \mu_0 \cdot f}}{2 \cdot a \cdot h} \tag{11}$$

[92]:
```
#Definida en m
R_skin_m=(np.sqrt(np.pi*rho_m*mu_0*fo)/(2*(w_m+t_m)))+(np.sqrt(np.
 ↪pi*rho_m*mu_0*fo)/(2*a*h_m))
R_skin_m
```

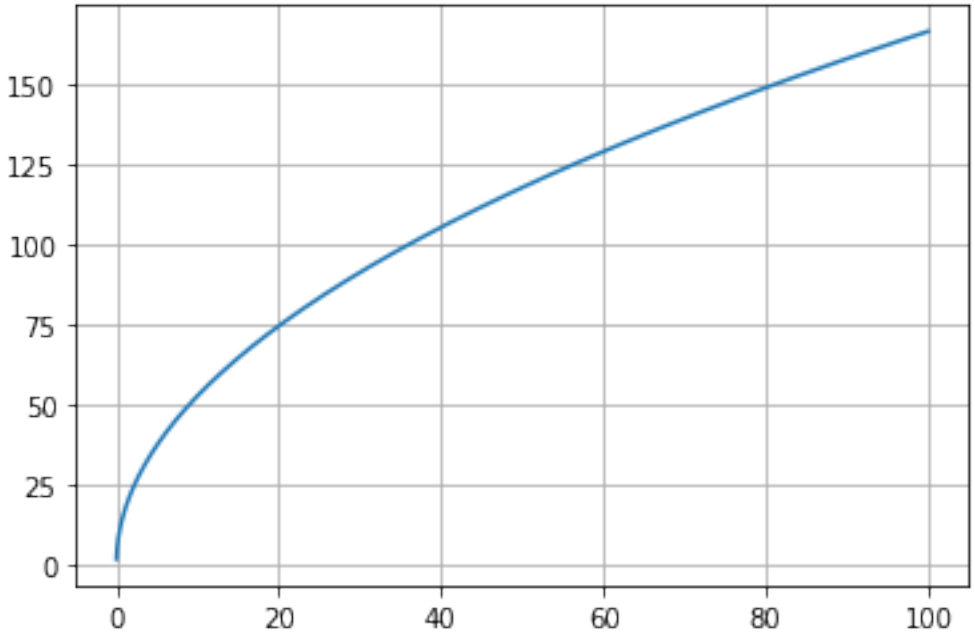
[92]: 16.63998685667905

Las dos últimas ecuaciones siguientes son la contribución desde la tierra return path??????

[93]:
```
#GRAFICAS DE LAS RESISTENCIAS EN FUNCION DE LA FRECUENCIA
#Calculo ahora en pulgadas
R_skin_in=(np.sqrt(np.pi*rho_in*mu_in*fi)/(2*(w_in+t_in)))+(np.sqrt(np.
 ↪pi*rho_in*mu_in*fi)/(2*a*h_in))
R_skin_m=(np.sqrt(np.pi*rho_m*mu_0*fi)/(2*(w_m+t_m)))+(np.sqrt(np.
 ↪pi*rho_m*mu_0*fi)/(2*a*h_m))
R_in=R_skin_in+Rdc_in
R_m=R_skin_m+Rdc_m
mp.plot(fi/1e9,R_skin_in,Rdc_in)
mp.grid()
```



[94]:
```
mp.plot(fi/1e9,R_in)
mp.grid()
```



La ecuación para el voltaje de la onda viajera viene dada por:

$$V(f,z,t) = e^{Re[\Gamma(f)]\cdot(\sqrt{\varepsilon_r}\cdot\frac{z}{c}\cdot t)} \cdot e^{Im[\Gamma(f)]\cdot(\sqrt{\varepsilon_r}\cdot\frac{z}{c}\cdot t)} \tag{12}$$

La constante dieléctrica es compleja:

$$\varepsilon(f) = \varepsilon_r \cdot (1 + j \cdot tan\delta(f)) \tag{13}$$

Para Microstrip tenemos:

$$Z_0 = \frac{87\Omega}{\sqrt{\varepsilon_r + 1.41}} \cdot ln\left[\frac{5.98 \cdot h}{(0.8 \cdot w) + t)}\right] \tag{14}$$

[95]:
```
#Calculo de Z0 con aproximación e_r = e_r(f)
Z0=(87/(np.sqrt(er+1.41)))*np.log(np.absolute((5.98*h_in)/((0.8*w_in)+t_in)))
Z0
```

[95]:
```
49.91595444094371
```

[96]:
```
#Calculo de L y C de la línea microstrip
c_m=1/np.sqrt(epsilon_0*mu_0)
c_in=c_m/0.0254
L=np.sqrt(er)*Z0/c_in
C=np.sqrt(er)/(c_in*Z0)
```

La relacion entre la conductancia y la tangente de pérdidas es la siguiente:

$$G(f) = \frac{2 \cdot \pi \cdot f \cdot \sqrt{\varepsilon_r}}{c \cdot Z_0} \cdot tan\delta(f) \tag{15}$$

$$\Gamma(f) = \sqrt{(R(f) + j \cdot 2 \cdot \pi \cdot L) \cdot (G(f) \cdot j \cdot 2 \cdot \pi \cdot f \cdot C)} \tag{16}$$

Buscando la parte real e imaginaria de $\Gamma(f)$ y definimos α(f) y β(f) como se muestra a continuación:

$$\alpha(f) = \frac{R(f)}{2 \cdot Z_0} + \frac{G(f) \cdot Z_0}{2} \tag{17}$$

$$\beta(f) = 2 \cdot \pi \cdot f \cdot \sqrt{L \cdot C} \cdot (1 - \frac{R(f) \cdot G(f)}{16 \cdot \pi^2 \cdot f^2 \cdot L \cdot C} \tag{18}$$

$$V(f,z,t) = e^{-\alpha(f)\cdot(\sqrt{\varepsilon_r}\cdot\frac{z}{c}\cdot t)} \cdot e^{j\cdot\beta(f)\cdot(\sqrt{\varepsilon_r}\cdot\frac{z}{c}\cdot t)} \tag{19}$$

Separamos el factor de atenuación en la atenuación resistiva (principalmente efecto skin) y la atenuación dieléctrica.

$$\alpha_{skin}(f) = \frac{R(f)}{2 \cdot |Z_0|} \tag{20}$$

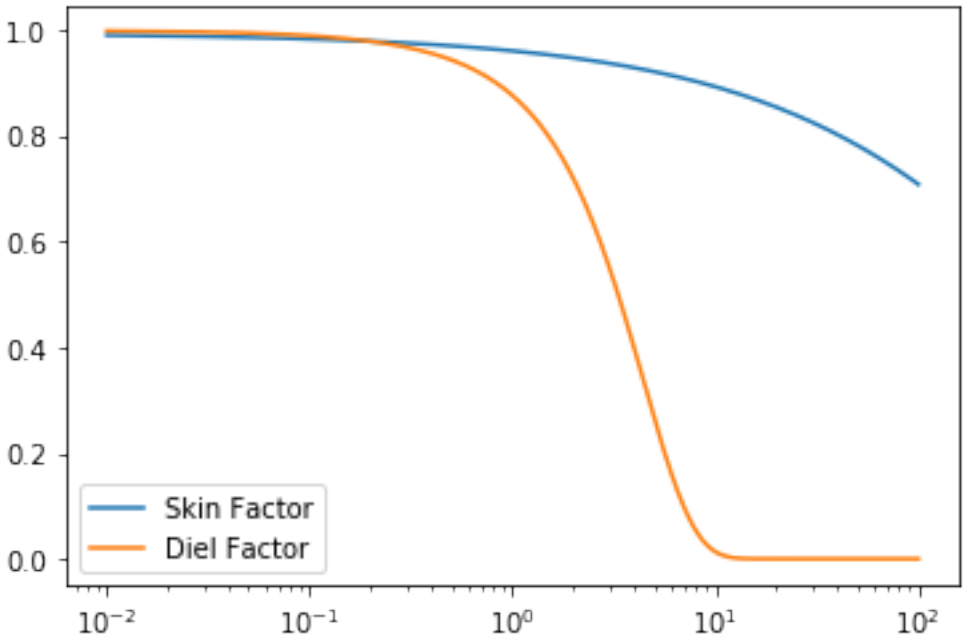
$$skin_factor(f) = e^{-\alpha_{skin(f)} \cdot L_0} \tag{21}$$

$$\alpha_{diel}(f) = \frac{G(f) \cdot Z_0}{2} \tag{22}$$

$$skin_{factor}(f) = e^{-\alpha_{diel(f)} \cdot L_0} \tag{23}$$

[97]:
```
#Calculamos todo esto

#Calculamos la Conductancia
G_in=(2*np.pi*fi*np.sqrt(er)*tan_delta)/(c_in*Z0)
G_m=(2*np.pi*fi*np.sqrt(er)*tan_delta)/(c_m*Z0)

#Partes reales e imaginarias
alpha=(R_in/(2*Z0)) + (G_in*Z0/2)
beta=2*np.pi*fi*np.sqrt(L*C)*(1-((R_in*G_in)/(16*np.pi**2*fi**2*L*C)))

# Factor de atenuación por efecto skin  y  pérdidas dieléctrico
alpha_skin=R_m/(2*np.absolute(Z0))
alpha_diel=G_m*Z0/2

skin_factor=pow(np.e,(-alpha_skin*L0_m))
diel_factor=pow(np.e,(-alpha_diel*L0_m))

# Grafica
mp.plot(fi/1e9,skin_factor, label = 'Skin Factor')
mp.plot(fi/1e9,diel_factor, label = 'Diel Factor')
mp.xscale("log")
mp.legend()
```

[97]:
```
<matplotlib.legend.Legend at 0x151d3db1d0>
```



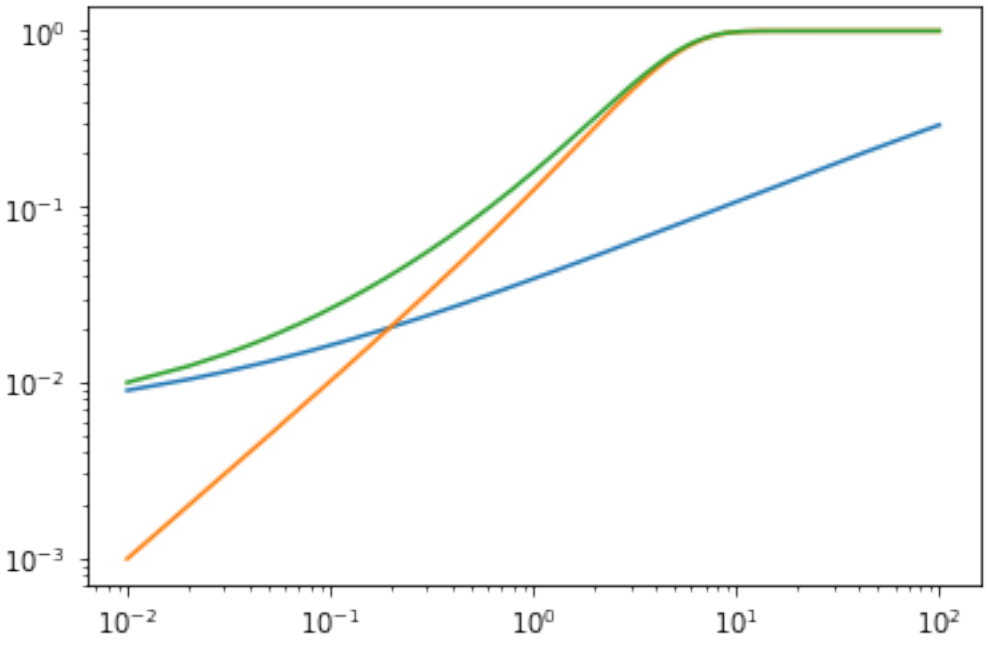
Calculamos el factor de perdidas LF de la señal

$$LF_{skin}(f) = (1 - skin\_factor(f)) \tag{24}$$

$$LF_{diel}(f) = (1 - diel\_factor(f)) \tag{25}$$

$$LF_{tot}(f) = (1 - skin\_factor(f) \cdot diel\_factor(f)) \tag{26}$$

[98]:
```
#Factor de atenuacion
LF_skin=1-skin_factor
LF_diel=1-diel_factor
LF_tot=(1-skin_factor*diel_factor)
mp.plot(fi/1e9,LF_skin)
mp.plot(fi/1e9,LF_diel)
mp.plot(fi/1e9,LF_tot)
mp.xscale("log")
mp.yscale("log")
```



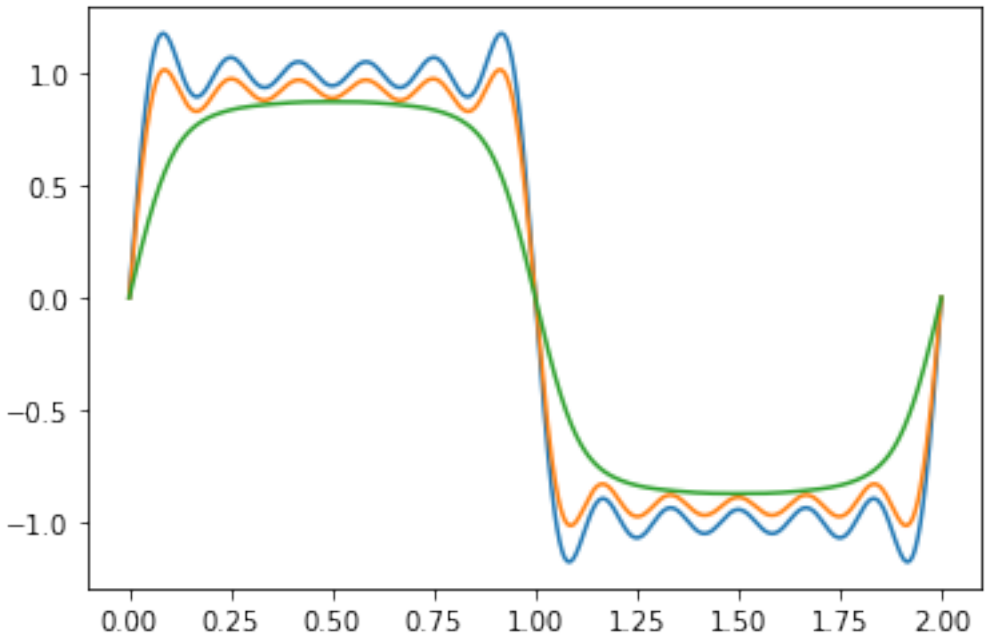
Construimos una onda cuadrada para visualizar estas pérdidas en el pulso:

[99]:
```
from numpy import pi
from numpy import sin,cos
from numpy import sqrt
```

[100]:
```
# Hay que calcular las pérdidas nuevamente para ese rango de frecuencias
k=np.arange(1,13,2)
Bk=2/k

j=np.arange(0,1001,1)
fc=10e9
T0=1/fc
Fk=k/(2*T0)
tj=((j/500)*T0) + (L0_m*sqrt(er)/c_m)

#RESISTENCIAS EN FUNCION DE LA FRECUENCIA
R_skin_m=(np.sqrt(np.pi*rho_m*mu_0*Fk)/(2*(w_m+t_m)))+(np.sqrt(np.
 ↪pi*rho_m*mu_0*Fk)/(2*a*h_m))
R_m=R_skin_m+Rdc_m
tan_delta=0.004+0.0002*Fk/1e9
G_m=(2*np.pi*Fk*np.sqrt(er)*tan_delta)/(c_m*Z0)
alpha_skin=R_m/(2*np.absolute(Z0))
alpha_diel=G_m*Z0/2
```

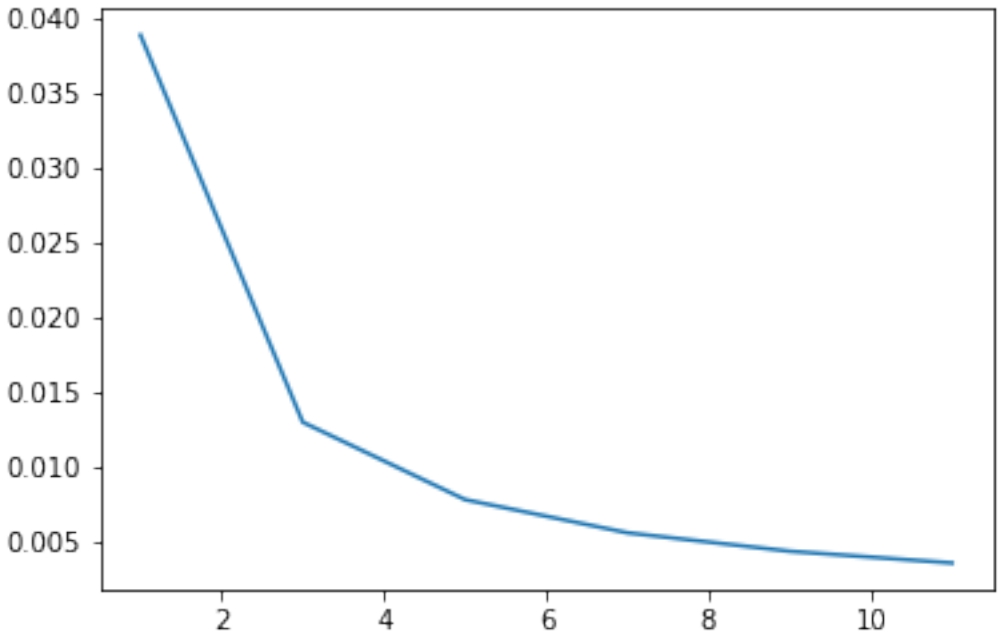
```
skin_factor=pow(np.e,(-alpha_skin*L0_m))
diel_factor=pow(np.e,(-alpha_diel*L0_m))

# 6 señales en función del tiempo que sumamos para obtener la señal cuadrada
So=0
So_skin=0
So_tot=0
for i in [0,1,2,3,4,5]:
    temp=(2/pi)*Bk[i]*sin(2*pi*Fk[i]*(tj-(L0_m*sqrt(er)/c_m)))
    So=temp+So

# Señal cuadrada con perdidas ohmicas
for i in [0,1,2,3,4,5]:
    Bk_skin=(2/k[i])*skin_factor[i]
    temp=(2/pi)*Bk_skin*sin(2*pi*Fk[i]*(tj-(L0_m*sqrt(er)/c_m)))
    So_skin=temp+So_skin

# Señal cuadrada con perdidas ohmicas y dielectricas
for i in [0,1,2,3,4,5]:
    Bk_tot=(2/k)*skin_factor[i]*diel_factor[i]
    temp=(2/pi)*Bk_tot[i]*sin(2*pi*Fk[i]*(tj-(L0_m*sqrt(er)/c_m)))
    So_tot=temp+So_tot

mp.plot((tj-(L0_in*np.sqrt(er)/c_in))/T0,So, label='So')
mp.plot((tj-(L0_in*np.sqrt(er)/c_in))/T0,So_skin, label='Sskin')
mp.plot((tj-(L0_in*np.sqrt(er)/c_in))/T0,So_tot, label='Stot')
```

[100]: [<matplotlib.lines.Line2D at 0x151fd40d50>]



[101]:
```
#mp.plot(Bk_tot/Bk,'o-')
mp.plot(k,Bk_tot)
```

[101]: [<matplotlib.lines.Line2D at 0x151fe15490>]



[ ]:

[ ]:

[ ]:

[ ]:

[ ]: