Estudio de la línea Nelco-6000

Bilal Hammu Mohamed

[9]:
```
import math
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import numpy.linalg
import matplotlib.pyplot as plt
import scipy.integrate as integrate
import scipy.special as special
from sympy import integrate
from sympy import Symbol
#Importamos las librerías que necesitaremos a posteriori.
```

El objetivo de este programa es calcular las perdidas en una microstrip, ya sea por efecto skin como pérdidas dieléctricas en una línea microstrip. Se trasmitirá una señal cuadrada por lo que será necesario conservar un cierto número de armónicos para no generar una distorsión de la señal.

La impedancia característica de la línea es: $Z_o(f) = \sqrt{\frac{R(f)+j2*\pi*f*L}{G(f)+j2*\pi*f*C}}$

Asumiendo que la onda se propaga en el eje Z, la ecuación que rige dicho voltaje es la siguiente:

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

Dónde tenemos una parte real y otra imaginaria, la parte imaginaria, al no alterar el módulo de la señal genera un cambio en la fase, mientras que la parte real, como decae con la longitud representa la atenuación que esta sufre.

La línea a caracterizar será una NELCO-6000, en la cuál definiremos los siguientes parámetros:

$\mu_0 = 4 * \pi * 10^-7 kg * m * coul^{-2}$ $\epsilon_0 = 8.84542 * 10^-12 \frac{farad}{m}$
$c = \frac{1}{\sqrt{(Epsilon0*mu0)}}$
$\rho = 6.585 * 10^7 \Omega * in = 0.167 * 10^7 \Omega * m$

[10]:
```
mu0=4*(math.pi)*10**(-7)
Epsilon0=8.4542*10**(-12)
c=1/math.sqrt(mu0*Epsilon0)
Lo=12*0.0254
Rho=6.585*0.0254*(10**(-7)) #Multiplicamos para pasar de pulgadas a metros.
```

En este caso la PCB que vamos a utilizar es la Nelco-6000 PC board.

Los parámetros de esta son los siguientes:

$$w := .012in \rightarrow w = 304.8\mu m$$
$$t = .001in \rightarrow t = 25.4\mu m$$
$$h = .006in \rightarrow h = 152.4\mu m$$
$$L_o := 12in \rightarrow L_o = 0.3045m$$
$$\epsilon_r = 3.2$$
$$tan(\delta(f)) = 0.004 + 0.0002\frac{f}{GHz}$$

[11]:
```
#Definiremos las variables que nombramos anteriormente
w=0.01*0.03048; #Multiplicamos para pasar a metros
t=25.4*10**(-6);
EpsilonR=3.2;
h=0.006*0.0254; #Multiplicamos para pasar a metros.
```

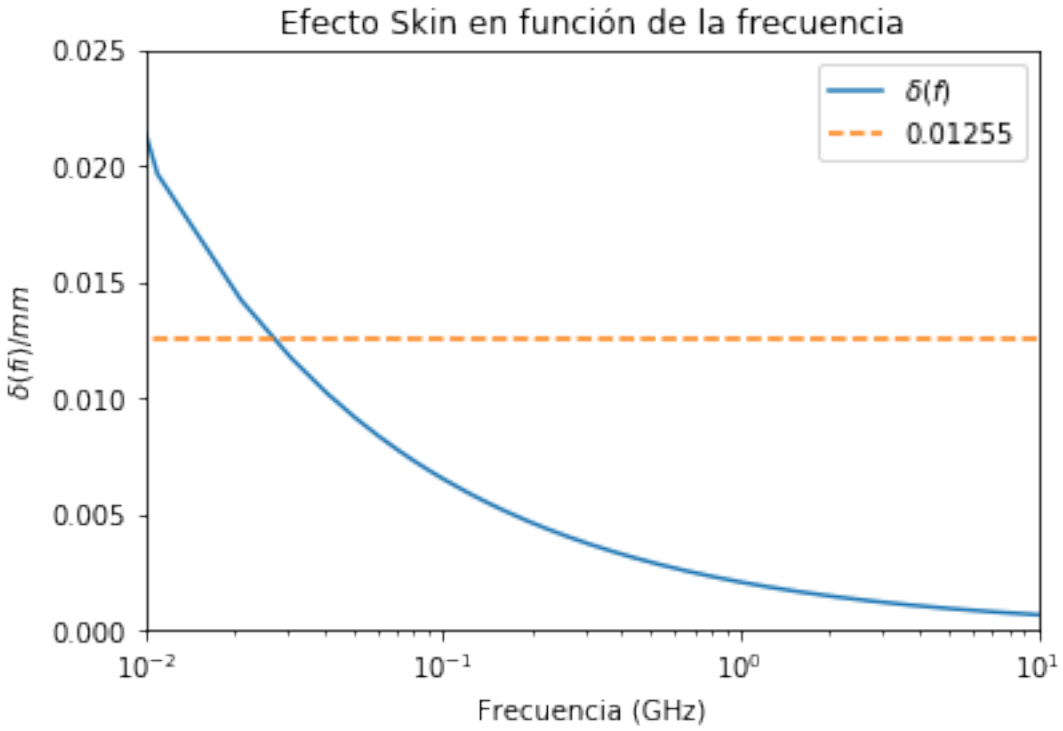
Al aumentar la frecuencia sobre un conductor se produce el conocido efecto skin o efecto pelicular que consiste en un aumento de la densidad de corriente en la zona superficial del cable debido a un incremento de campo magnético en la zona central del conductor, lo que dificulta el movimiento de los electrones en dicha zona.
Este efecto genera una reducción de la sección transversal utilizada y viene dado por dicha expresión:

$$\delta(f) = \sqrt{\frac{2\rho}{2\pi f \mu_0}}$$

A continuación pasamos a estudiar el efecto skin o efecto pelicular en este caso:

[12]:
```
#Representamos el efecto Skin en función de la frecuencia.
i=np.linspace(1,100000,10000) #Definimos el vector i que serán las muestras que␣
 ↪tomemos de la frecuencia
f=i*10**6  #Vector frecuencias
Delta=np.zeros(len(f)) #Será nuestro efecto skin, primeramente lo definimos como␣
 ↪ceros
                       #y luego en el bucle lo pondremos a funcionar
for k in range(len(f)):
    Delta[k]=math.sqrt(Rho/(math.pi*f[k]*mu0)) #Realizamos la operacion y vamos␣
 ↪introduciendo los resultados en el vector Delta.


#Mostramos los resultados en una gráfica
plt.plot(f/1e9, Delta*1e3,label='$\delta(f)$') #Multiplicamos por 10e9 para␣
 ↪pasar a GHz y por 10e3 para pasar a mm.
plt.plot(f/1e9, np.ones(len(f))*0.01255, linestyle='--', label='0.01255')
plt.legend()
```

```
matplotlib.pyplot.xscale('log') #Pone el eje x en escala logaritmica
plt.title("Efecto Skin en función de la frecuencia")
plt.xlim(0.01, 10)
plt.ylim(0, 0.025)
plt.xlabel('Frecuencia (GHz)')
plt.ylabel("${\delta(fi)}/{mm}$")
plt.show()
```



Según el criterio que tomemos el efecto skin tendrá importancia a partir de ciertas frecuencias, en el documento en el que nos basamos dicho efecto cobra efecto apartir de 25 Mhz, que es dónde dicho efecto alcanza el valor de 0.01255.

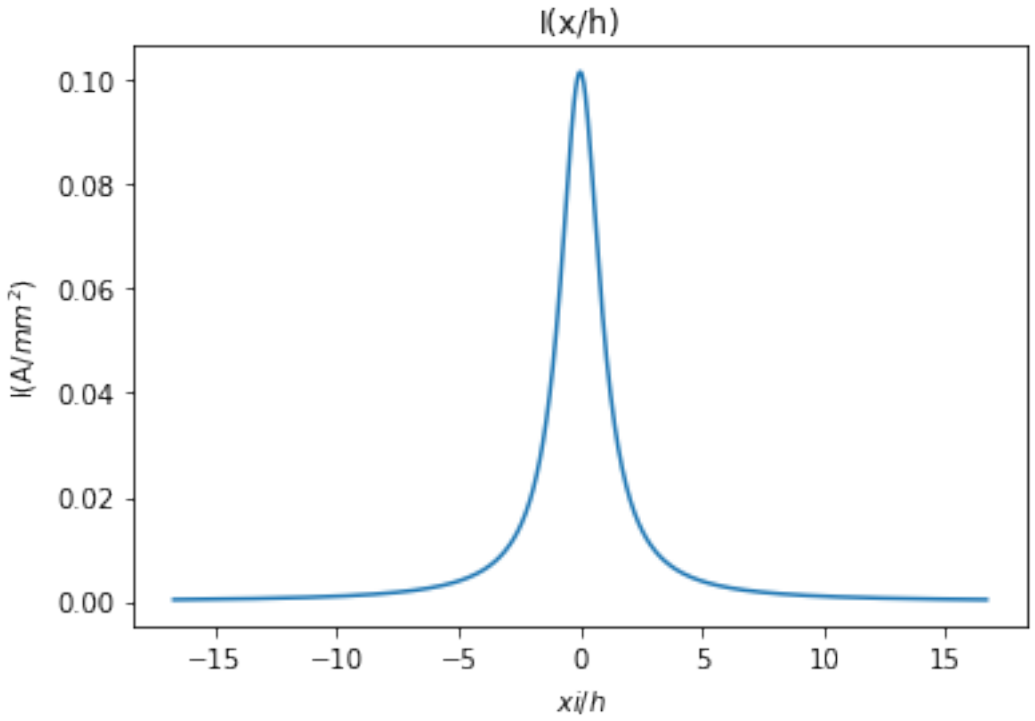
En dicha línea microstrip la corriente dependerá de la frecuencia, aquí la corriente I_o= 0.1 mA para 1 GHz, utilizando la siguiente expresión:

$$I(x) := if[f < 25MHz, \frac{Io}{(\pi * h * t[1 + \frac{x}{h}^2]}, \frac{Io}{(\pi * h * \delta(f) * [1 + \frac{x}{h}^2]}]$$

Dónde $i = 1..100$, $xi = \frac{(i-50)h}{3}$ Pasaremos a estudiar el comportamiento de la corriente según el valor de x:

[13]:
```
Skin_1GHz=math.sqrt(Rho/(math.pi*1e9*mu0)) #Definimos el valor del efecto skin a␣
↪1 GHz.

Io=0.1e-3; #Corriente a 1 GHz (la da el enunciado)
```

```
i=np.linspace(0,100,1000)
I=np.zeros(len(i))
xi=(i-50)*h/3 #Iremos variando el valor de x para estudiar la corriente para␣
 ↪distintos valores

for k in range(len(xi)):
    I[k]=Io/(math.pi*h*Skin_1GHz*(1+(xi[k]/h)**2)) #Aplicamos la expresión

#Monstramos en una gráfica los resultados obtenidos
plt.plot(xi/h,I*1e-6) #Multiplicamos por 1e-6 para pasar a Amperios/(mm^2)
plt.title("I(x/h)")
plt.ylabel("I(A/$mm^2$)")
plt.xlabel("$xi/h$")
plt.grid
plt.show()
```



Pasaremos a estudiar el valor límite de x para conseguir que la corriente sea el 5% de su valor original, para ello:

$$a = raiz[\delta(f) \int_{-ah}^{ah} I(x)dx - 0.95I_o, a]$$

[14]:
```
from sympy import *
```

```
x,a=symbols('x,a') #Insertamos a y x como simbolos
integral=integrate(Io/(math.pi*h*Skin_1GHz*(1+(x/h)**2)),(x,-a*h,a*h))␣
 ↪#Calculamos dicha integral
a=solveset(Skin_1GHz*integral-0.95*Io,a,domain=S.Reals) #Igualamos a cero y␣
 ↪resolvemos la ecuación
a
```

[14]: $\{12.7062047361747\}$

[15]:
```
Por tanto el valor de a deseado es de 12.706.
```

```
  File "<ipython-input-15-4779ff3aaaab>", line 1
    Por tanto el valor de a deseado es de 12.706.
             ^
SyntaxError: invalid syntax
```

Sobre la resistencia en DC la calculamos con la siguiente expresión:

$$R_{DC} := \frac{\rho}{wt} + \frac{\rho}{2aht}$$

[16]:
```
a=12.703
R_dc=Rho/(w*t)+Rho/(2*a*h*t)
R_dc
```

[16]: 2.33050573644641

Tras aplicar la expresión, el resultado es:

$$R_{DC} = 0.059\frac{\Omega}{in}$$

Y en metros, dicho resultado es:

$$R_{DC} = 2.33054\frac{\Omega}{m}$$

Al calcular la resistencia a alta frecuencia, lógicamente tenemos que tener en cuenta el efecto skin ya que este afectará a la superficie de conductor que realmente actúa como tal. Para esto aplicamos la siguiente expresión:

$$R_{skin}(f) := \frac{\sqrt{\rho\pi\mu_0 f}}{2(w+t)} + \frac{\sqrt{\rho\pi\mu_0 f}}{2ah}$$

[ ]:
```
R_skin_1GHz=math.sqrt(Rho*math.pi*mu0*1e9)/(2*(w+t))+math.sqrt(Rho*math.
 ↪pi*mu0*1e9)/(2*a*h)
```

Y tras aplicar los cálculos a la frecuencia de 10 GHz:

$$R_{skin}(f) := 14.403\frac{\Omega}{m}$$

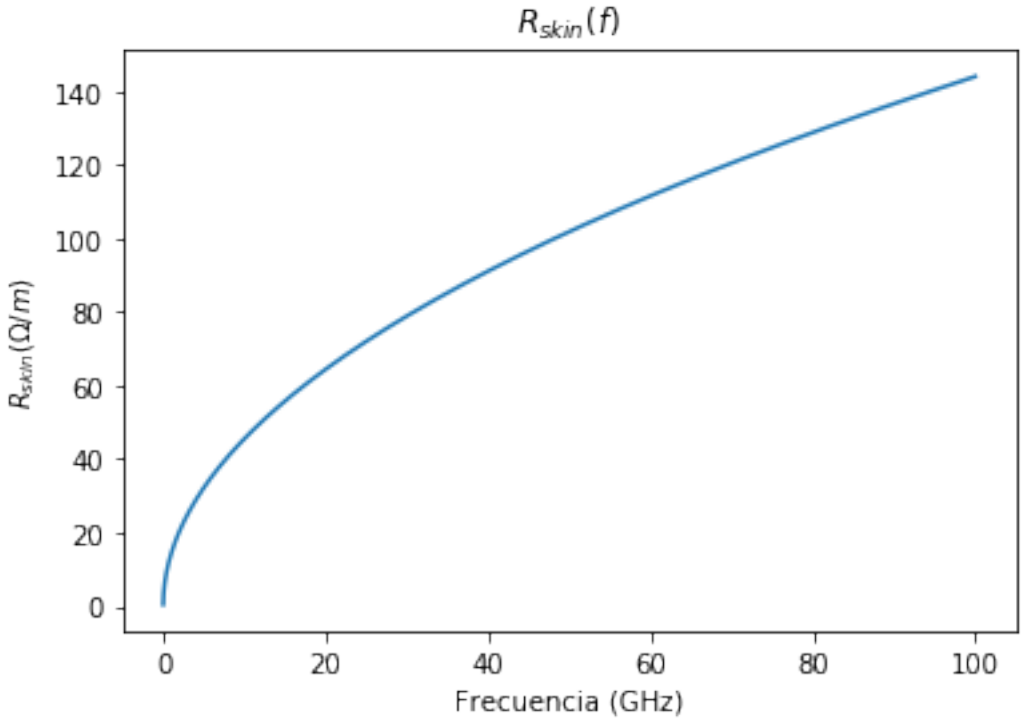
A continuación estudiamos la dependencia de la resistencia con la frecuencia:

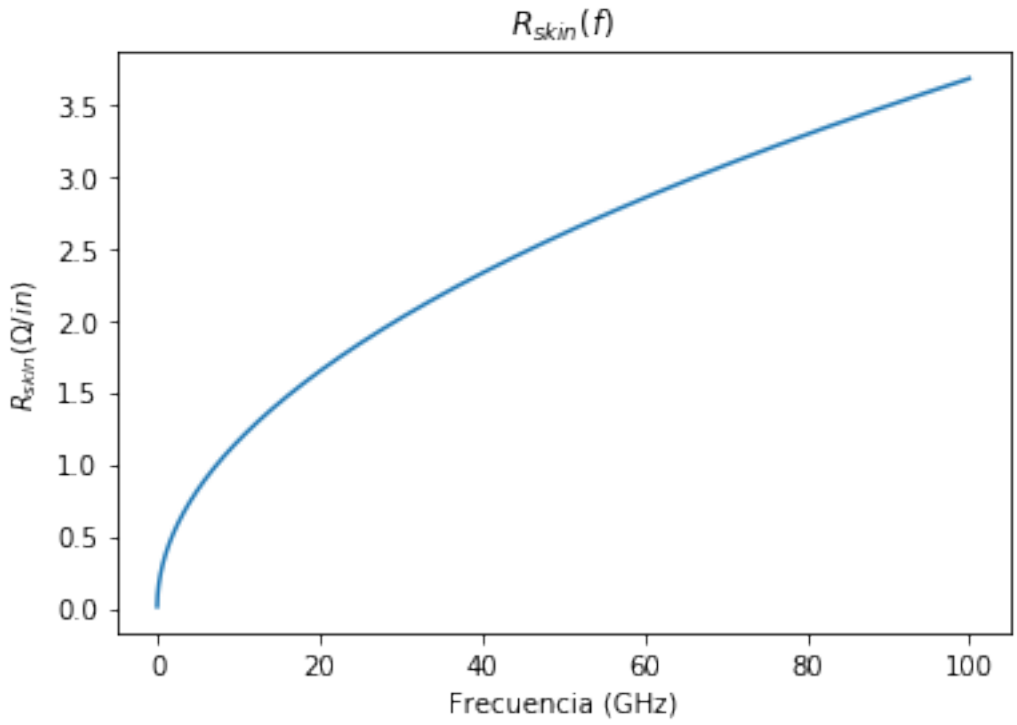
$$R(f) = R_{skin}(f) + R_{DC}$$

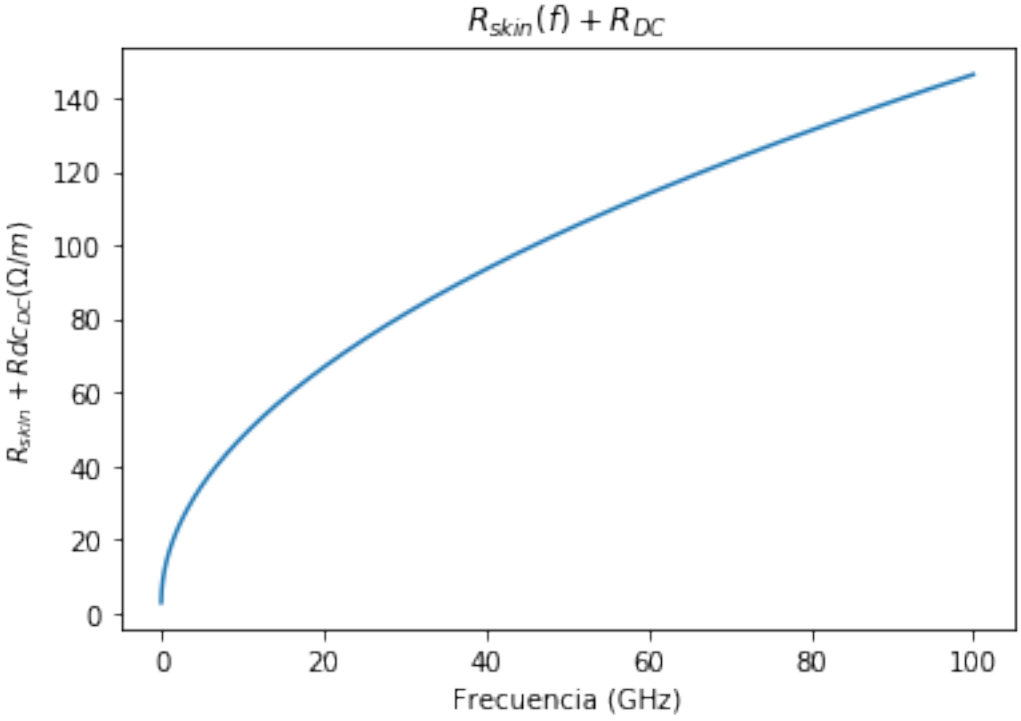
, estudiamos dicha dependencia:

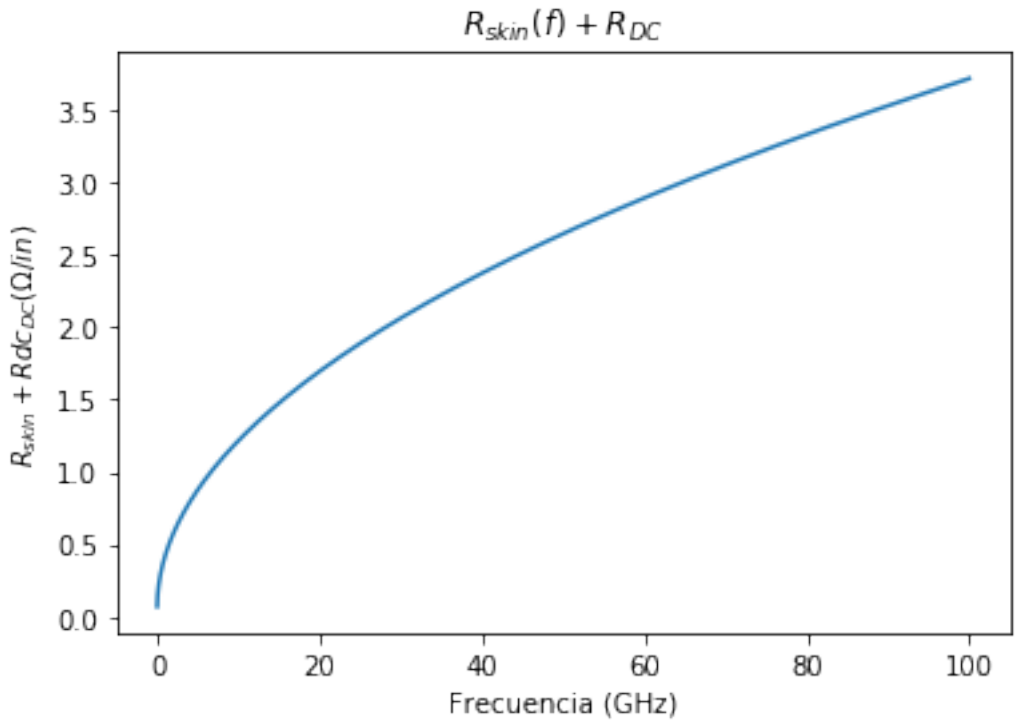
[17]:
```
R_skin_frecuencia=np.zeros(len(f)) #Definimos un vector para la Rskin
G=np.zeros(len(f))

for k in range(len(f)):
    R_skin_frecuencia[k]=math.sqrt(Rho*math.pi*mu0*f[k])/(2*(w+t))+math.
 ↪sqrt(Rho*math.pi*mu0*f[k])/(2*a*h)
    #Calculamos R_skin a distintas frecuencias
R=R_skin_frecuencia+R_dc #Aplicamos la expresión de la resistencia total

#Mostramos los datos en una gráfica
plt.plot(f*1e-9,(R_skin_frecuencia))
plt.title("$R_{skin}(f)$")
plt.ylabel("$R_{skin} (\Omega/m)$")
plt.xlabel("Frecuencia (GHz)")
plt.grid
plt.show()

#Mostramos los datos en pulgadas
plt.plot(f*1e-9,(R_skin_frecuencia)/39.17) #Dividimos entre 39.37 para pasar de␣
 ↪metros a pulgadas
plt.title("$R_{skin}(f)$")
plt.ylabel("$R_{skin} (\Omega/in)$")
plt.xlabel("Frecuencia (GHz)")
plt.grid
plt.show()

#Mostramos Rskin+R_dc
plt.plot(f*1e-9,R_skin_frecuencia+R_dc)
plt.title("$R_{skin}(f)+R_{DC}$")
plt.ylabel("$R_{skin}+Rdc_{DC} (\Omega/m)$")
plt.xlabel("Frecuencia (GHz)")
plt.grid
plt.show()

#Mostramos lo mismo que anteriormente pero esta vez en pulgadas.
plt.plot(f*1e-9,(R_skin_frecuencia+R_dc)/39.37) #Dividimos entre 39.37 para␣
 ↪pasar de metros a pulgadas.
plt.title("$R_{skin}(f)+R_{DC}$")
```

```
plt.ylabel("$R_{skin}+Rdc_{DC} (\Omega/in)$")
plt.xlabel("Frecuencia (GHz)")
plt.grid
plt.show()
```




$R_{skin}(f)$
$R_{skin}(\Omega/in)$
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
20
40
60
80
100
Frecuencia (GHz)


$R_{skin}(f) + R_{DC}$
$R_{skin} + Rdc_{DC}(\Omega/m)$
0
20
40
60
80
100
120
140
0
20
40
60
80
100
Frecuencia (GHz)



Vamos a estudiar el voltaje viajando a través de la línea de trasmisión, debido a que nos encontramos en una guía de onda aquí no son válidas las leyes de Kirchhoff. Utilizaremos las siguientes expresiones:
$V(f,z,t) := e^{Re(\Gamma(f))(\sqrt{\epsilon_r}\frac{z}{c}-t)} j e^{Im(\Gamma(f))(\sqrt{\epsilon_r}\frac{z}{c}-t)}$

$\epsilon(f) := \epsilon_r(1 + jtan\delta(f))$ Y la expresión para la impedancia característica viene dada por:
$Z_o := \frac{87\Omega}{\sqrt{\epsilon_r+1.41}} ln[\frac{5.98h}{(0.8w)+t}]$

Para el estudio de la inductancia y la capacidad de la línea de trasmisión se aplica las siguientes expresiones:
$L = \frac{\sqrt{\epsilon_r} Z_o}{c}$ $C = \frac{\sqrt{\epsilon_r}}{cZ_o}$

[18]:
```
#Calculamos la impedancia caracteristica de la guia, la capacidad caracteristica
↪y la inductancia caracteristica
Zo=87/(math.sqrt(EpsilonR+1.41))*ln(5.98*h/(0.8*w+t))
L=math.sqrt(EpsilonR)*Zo/c
C=math.sqrt(EpsilonR)/(c*Zo)
```

$$Z_o = 49.4\Omega$$

$$L = 7.488\frac{nH}{in} \rightarrow L = 288.07\frac{nH}{m}$$

$$C = 3.06 * 10^{-12}\frac{F}{in} \rightarrow C = 118.01 * 10^{-12}\frac{F}{m}$$

En cuanto a la relación entre la conductancia y la tangente de pérdidas viene dado por la siguiente expresión:

$$G(f) = \frac{2\pi f\sqrt{\epsilon_r}}{cZ_o} tan\delta(f)$$

Y como:

$$\Gamma(f) = \sqrt{(R(f) + j\omega L)(G(f) + j\omega C)}$$

Encontrando la parte real e imaginaria del coeficiente de reflexión podemos definir $\alpha(f)$ y $\beta(f)$, siendo la constante de atenuación y la constante de fase respectivamente. Dichos términos dependen de la frecuencia y vienen dados por la siguiente expresión:

$$\alpha(f) = \frac{R(f)}{2Z_o} + \frac{G(f)}{2} Z_o$$

$$\beta(f) = \omega\sqrt{LC}(1 - \frac{R(f)G(f)}{16\pi^2 f^2 LC})$$

$$V(f, z, t) := e^{\alpha(f)(\sqrt{\epsilon_r}\frac{z}{c} - t)} j e^{\beta(f)(\sqrt{\epsilon_r}\frac{z}{c} - t)}$$

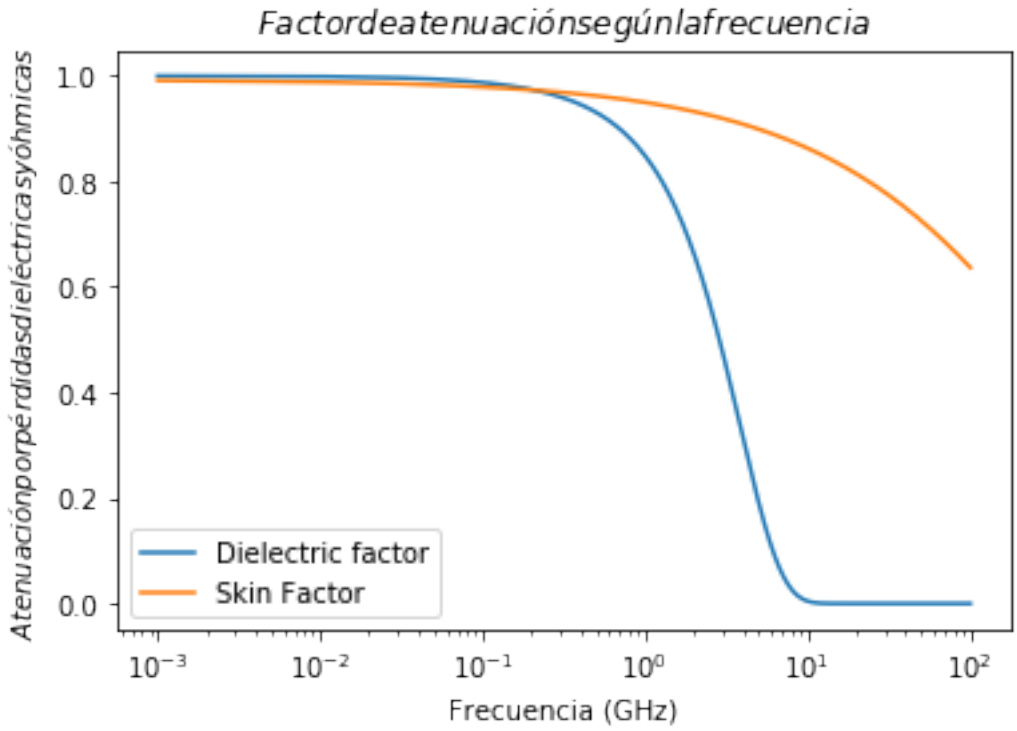
Además, en el factor de atenuación nos encontramos con una serie de pérdidas debidas a pérdidas óhmicas y otras a pérdidas dieléctricas, dichas pérdidas vienen dadas por las siguientes expresiones:

$$\alpha_{skin}(f) = \frac{R(f)}{2|Z_o|} \rightarrow skinfactor(f) = e^{-\alpha_{skin}(f)L_o}$$

$$\alpha_{dielectrico}(f) = G(f)\frac{Z_o}{2} \rightarrow dielectricfactor(f) = e^{-\alpha_{dielectric}(f)L_o}$$

[19]:
```
#Pasamos a calcular la dependencia de las pérdidas con la frecuencia
G=np.zeros(len(f)) #Conductancia
skin_factor=np.zeros(len(f)) # Pérdidas debidas al efecto skin
dielectric_factor=np.zeros(len(f)) #Pérdidas debidas al dieléctrico
alpha_skin=np.zeros(len(f)) #Cte de perdidas por efecto skin
alpha_dielectric=np.zeros(len(f)) #Cte de perdidas por el dielectrico

G=2*math.pi*f*math.sqrt(EpsilonR)*(0.022+0.0075*f*1e-9)/(c*Zo)
alpha_skin=R/(2*Zo)
alpha_dielectric=G*Zo/2
for m in range(len(f)):
    skin_factor[m]=math.e**(-alpha_skin[m]*Lo)
    dielectric_factor[m]=math.e**(-alpha_dielectric[m]*Lo)

#Representamos dichas variables
```

```
plt.plot(f*1e-9,dielectric_factor, label='Dielectric factor')
plt.plot(f*1e-9,skin_factor, label='Skin Factor')
plt.legend()
plt.show
plt.title("$Factor de atenuación según la frecuencia$")
plt.ylabel("$Atenuación por pérdidas dieléctricas y óhmicas$")
plt.xlabel("Frecuencia (GHz)")
matplotlib.pyplot.xscale('log') #Pone el eje x en escala logaritmica
plt.grid
plt.show()
```



Ahora vamos a calcular el factor de perdidas de la señal, podemos hablar de 3 factores de pérdidas distintos: factor de pérdidas por efecto skin, factor de pérdidas dieléctricas y factor de pérdidas total y se calculan con las siguientes expresiones:

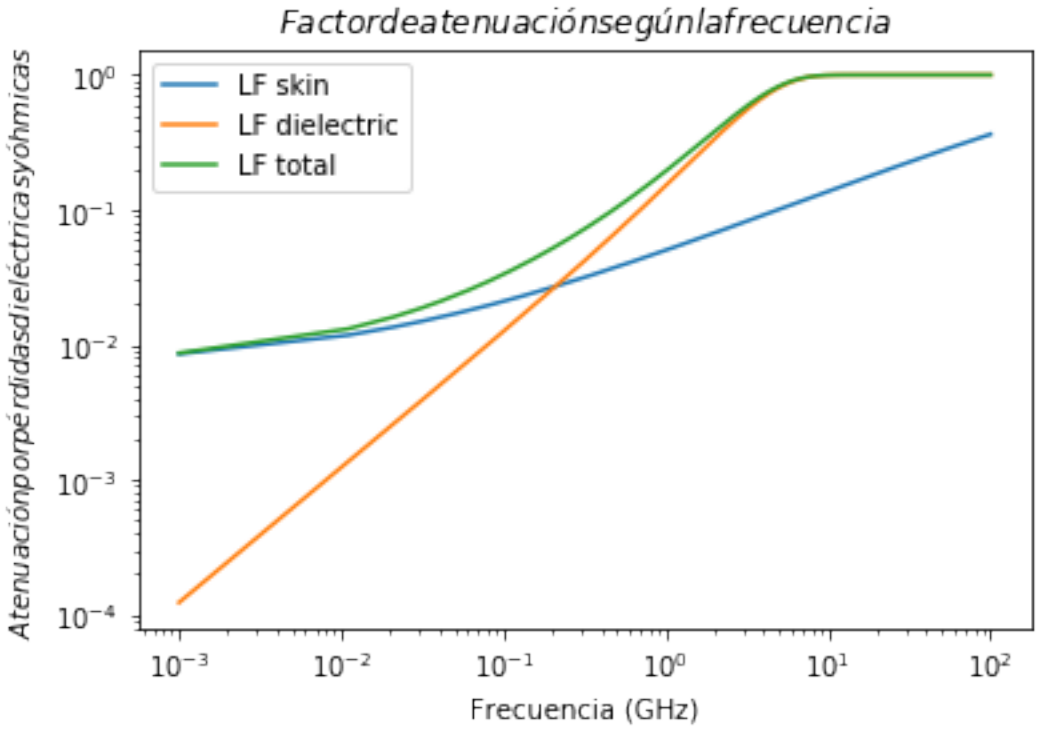
$$LossFactor_{skin}(f) = (1 - skin_factor(f))$$

$$LossFactor_{dielectric}(f) = (1 - dielectric_{factor}(f))$$

$$LossFactor_{total}(f) = (1 - dielectric_{factor}(f) * skin_{factor})$$

Pasamos a calcular los tres factores y a representarlos:

[20]:
```
LF_skin=1-skin_factor
LF_diel=1-dielectric_factor
LF_total=1-skin_factor*dielectric_factor

#Graficamos los factores de perdidas
plt.plot(f*1e-9,LF_skin, label='LF skin')
plt.plot(f*1e-9,LF_diel, label='LF dielectric')
plt.plot(f*1e-9,LF_total, label='LF total')
plt.legend()
plt.show
plt.title("$Factor de atenuación según la frecuencia$")
plt.ylabel("$Atenuación por pérdidas dieléctricas y óhmicas$")
plt.xlabel("Frecuencia (GHz)")
matplotlib.pyplot.xscale('log') #Pone el eje x en escala logaritmica
matplotlib.pyplot.yscale('log') #Pone el eje x en escala logaritmica
plt.grid
plt.show()
```



Una vez hecho este estudio previo, pasamos a estudiar como se comporta una señal cuadrada en dicha línea de trasmisión, para ello, primeramente definiremos una línea de trasmisión con una serie de armónicos:

$k = 1,3..11$
$fc = 10GHz$
$T_o = \frac{1}{f_c}$
$F_k = k2T_o$
$B_k = if(k < 1,0,2/k)$
$S_o(t) = \frac{2}{\pi}[\sum[B_k sin[2\pi F_k(t - \frac{L_o\sqrt{\epsilon_r}}{c}]]]$

Seguidamente añadimos las pérdidas óhmicas:

$B_{skin_k} = if(k < 1,0,\frac{2}{k} skinfactor(F_k))$

$S_{skin}(t) = \frac{2}{\pi}[\sum[B_{skin_k} sin[2\pi F_k(t - \frac{L_o\sqrt{\epsilon_r}}{c}]]]$

Seguidamente añadimos las pérdidas del dieléctrico:

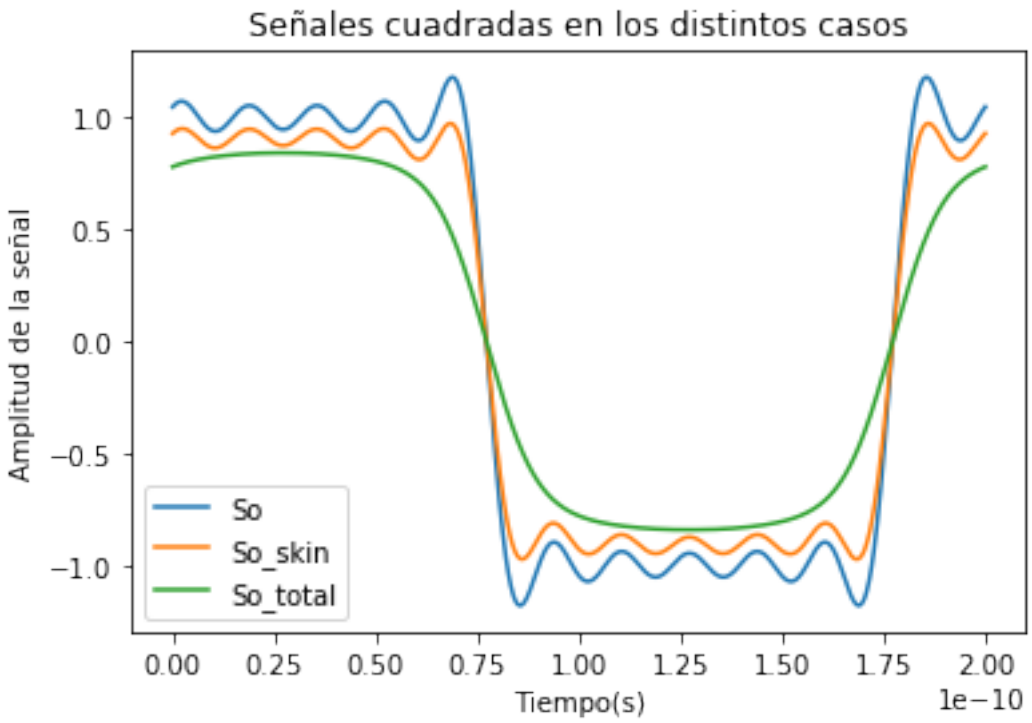
$B_{tot_k} = if(k < 1,0,\frac{2}{k} skinfactor(F_k) dielectricfactor(F_k))$

$S_{tot_k}(t) = \frac{2}{\pi}[\sum[B_{tot_k} sin[2\pi F_k(t - \frac{L_o\sqrt{\epsilon_r}}{c}]]]$

Para el tiempo tomaremos distintas muestras de este, siendo $j = 0,1...1000$ y $t_j = \frac{j}{500} * T_o + \frac{L_o\sqrt{\epsilon_R}}{c}$

[21]:
```
## Definimos ciertos parametros de la señal cuadradada
k=numpy.arange(1,12,2) #Crea un vector que va desde 1 hasta 11 de 2 en 2
j=numpy.arange(0,1001,1) #Crea un vector de 0 a 1000 de 1 en 1.
fc=10e9
To=1/fc
Fk=k/(2*To)
time=j*To/500


#Definimos el vector de tiempos y el seno:
So=np.zeros(len(j))
So_nuevo=np.zeros(len(time))
So_nuevo_skin=np.zeros(len(time))
So_nuevo_total=np.zeros(len(time))
So_skin=np.zeros(len(j))
So_total=np.zeros(len(j))
#Aplicamos la expresión de la constante de perdidas por efecto skin y el factor␣
 ↪de skin
Stotal=np.zeros(len(j))

for i in range(len(k)):

    #Parametros de perdidas por 'skin effect'
    R_freq=Rho/(w*t)+Rho/(2*a*h*t)+math.sqrt(Rho*math.pi*mu0*Fk[i])/
↪(2*(w+t))+math.sqrt(Rho*math.pi*mu0*Fk[i])/(2*a*h)
    alpha_skin=R_freq/(2*Zo)
    Skinfactor=exp(-alpha_skin*Lo)
```

```
    #Parametros de perdidas dielectricas
    G=2*math.pi*Fk[i]*sqrt(EpsilonR)*(0.004+0.0002*Fk[i]*1e-9)/(c*Zo)
    alpha_dielectric=G*Zo/2
    dielectric_factor=exp(-alpha_dielectric*Lo)

    for m in range(len(j)):
        So_nuevo[m]=(2/k[i])*sin(2*math.pi*Fk[i]*(m*To/500-Lo*math.
↪sqrt(EpsilonR)/c))*2/math.pi
        So_nuevo_skin[m]=(2*Skinfactor/k[i])*sin(2*math.pi*Fk[i]*(m*To/
↪500-Lo*math.sqrt(EpsilonR)/c))*2/math.pi
        So_nuevo_total[m]=(2*Skinfactor*dielectric_factor/k[i])*sin(2*math.
↪pi*Fk[i]*(m*To/500-Lo*math.sqrt(EpsilonR)/c))*2/math.pi

    So=So+So_nuevo
    So_skin=So_skin+So_nuevo_skin
    So_total=So_total+So_nuevo_total



plt.plot(time,So, label='So')
plt.plot(time,So_skin, label='So_skin')
plt.plot(time,So_total, label='So_total')
plt.title('Señales cuadradas en los distintos casos')
plt.xlabel("Tiempo(s)")
plt.ylabel("Amplitud de la señal")
plt.legend()
plt.show()
```

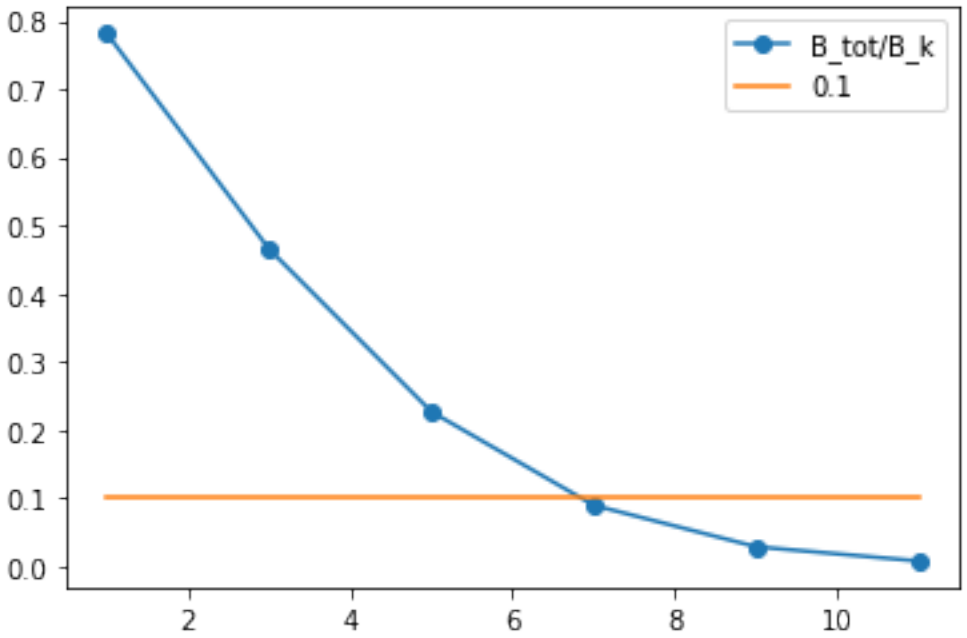


Ya tenemos una idea aproximada del comportamiento del sistema en cuanto a la señal cuadrada, podemos ver como al introducir pérdidas las oscilaciones desaparecen, debido a que dichas oscilaciones se deben a armónicos de alta frecuencia dónde las pérdidas se hacen mucho mayores.
Supongamos que queremos que dicha condición: $B_{tot}/B > 0.1$ Se cumpla hasta el quinto armónico, esto es equivalente a decir que queremos que el tiempo de subida del 0 al 90% sea $0.25T_o$ Para ello, graficamos B_{tot_k} y B_k), de forma que tienen que estar por encima de 0.1.

[22]:
```
#Será necesario redefinir ciertos parámetros ya utilizados anteriormente, pero␣
 ↪esta vez para las nuevas frecuencias a estudiar
R_freq=np.zeros(len(k))
alpha_skin=np.zeros(len(k))
Skinfactor=np.zeros(len(k))
for i in range(len(k)):
    R_freq[i]=Rho/(w*t)+Rho/(2*a*h*t)+math.sqrt(Rho*math.pi*mu0*Fk[i])/
 ↪(2*(w+t))+math.sqrt(Rho*math.pi*mu0*Fk[i])/(2*a*h)
    alpha_skin[i]=R_freq[i]/(2*Zo)
    Skinfactor[i]=exp(-alpha_skin[i]*Lo)

G=np.zeros(len(k))
alpha_dielectric=np.zeros(len(k))
dielectric_factor=np.zeros(len(k))
```

```
#Repetimos el mismo procedimiento para las pérdidas totales, en estas␣
 ↪incluiremos las pérdidas óhmicas y las dieléctricas:
for i in range(len(k)):
    G[i]=2*math.pi*Fk[i]*sqrt(EpsilonR)*(0.004+0.0002*Fk[i]*1e-9)/(c*Zo)
    alpha_dielectric[i]=G[i]*Zo/2
    dielectric_factor[i]=exp(-alpha_dielectric[i]*Lo)

Bk=2/k
Btot=2*dielectric_factor*Skinfactor/k

plt.plot(k,Btot/Bk,'o-', label='B_tot/B_k')
plt.plot(k,ones(len(k),1)*0.1, label='0.1')
plt.legend()
```

[22]: <matplotlib.legend.Legend at 0x28974ec2d88>



Se ve como dicha condición se cumple hasta el tercer armónico, ya que está por encima. En caso de que no se cumpliese sería necesario varíar f_c o L_o para que se cumpliese.