EE5613_Lab6_Conductance

April 18, 2025

[30]:
```
import skrf as rf
import matplotlib.pyplot as plt
import numpy as np
import cmath as cm
import math
import sympy as sp
from sympy.solvers import solve
pi = math.pi

#Prints out numbers without "np.flat64" displaying
np.set_printoptions(legacy='1.25')
```

[31]:
```
# Constants
epsilon_0 = 8.854*10**-12 #permitivity of free space
mu_0 = 4*pi*10**-7 #permeability of free space
c = 2.998*10**8 #speed of light
```

[32]:
```
# Generic Equations

def freq_to_omega(freq) : #get angular frequency (rad/s)
    return 2*pi*freq

def Np_to_dB(Np) : # converts Nepers to Decibels.
    return Np*20/(np.log(10))

#mu_r, epsilon_r = relative properties of medium
#freq = frequency of propagation
def calc_wavenumber(mu_r, epsilon_r, freq) : # wavenumber, k
    return freq_to_omega(freq)*np.sqrt(mu_r*mu_0*epsilon_r*epsilon_0)

# k = wavenumber
def calc_SL_guide_wavelength(k) : # lambda_g
    return 2*pi/k

#frequency = frequency of propagation
#conductance = property of conductor, sigma
def calc_surface_resistance(frequency,conductance) :
    return np.sqrt(freq_to_omega(frequency)*mu_0/(2*conductance))
```

0.0.1 Stripline Equations

[33]:
```
#Rs = surface resistance.
#b = dialectric thickness
#t = internal conductor thickness
#W = internal conductor width
def calc_SL_conductor_loss(Zo, epsilon_r, Rs, b, t, W) :
    if (np.sqrt(epsilon_r)*Zo) <= 120 :
        A = 1 + (2*W)/(b-t) + (1/pi)*(b+t)*(np.log((2*b-t)/t))/(b-t)
        print(A)
        return (2.7*(10**-3)*Rs*epsilon_r*Zo)/(30*pi*(b-t))*A
    else :
        B = 1 + (b)*(0.5 + (0.414*t/W) + (1/(2*pi))*np.log(4*pi*W/t))/(0.5*W +
 ↪0.7*t)
        return (0.16*Rs/(Zo*b))*B

#tandelta = dialectric property
#k = wavenumber
def calc_SL_dialectric_loss(k,tandelta) :
    return (k*tandelta/2)

#Zo = characteristic impedence
#epsilon_r = relative permitivity of dialectric
def calc_SL_width(Zo,epsilon_r, b) :
    x = (30*pi)/(np.sqrt(epsilon_r)*Zo) - 0.441
    print(x)
    if np.sqrt(epsilon_r)*Zo <= 120 :
        return x*b
    else :
        return (0.85 - np.sqrt(0.6-x))*b

#Zo = characteristic impedence
#epsilon_r = relative permitivity of dialectric
#mu_r = relative permeability of dialectric
#freq = frequency of line
#b = dialectric thickness
#t = thickness of internal conductor.
#tandelta = property of dialectric
#conductance = conductance of conductor.
def calc_SL_total_loss(Zo, epsilon_r, mu_r, freq, b, t, tandelta, conductance) :
    W = calc_SL_width(Zo,epsilon_r, b)
    print(f'W = {W}')
    k = calc_wavenumber(mu_r, epsilon_r, freq)
    Rs = calc_surface_resistance(freq,conductance)
    cond_loss = Np_to_dB(calc_SL_conductor_loss(Zo, epsilon_r, Rs, b, t, W))
    print(f'Conductor Loss = {cond_loss} dB/m')
    dialectric_loss = Np_to_dB(calc_SL_dialectric_loss(k, tandelta))
```

```
    print(f'Dialectric Loss = {dialectric_loss} dB/m')
    return cond_loss + dialectric_loss
```

0.0.2 Microstrip Equations

[34]:
```
#epsilon_r = relative permitivity of dialectric
#Zo = desired characteristic impedence of line
#d = substrate thickness
def calc_MS_width(epsilon_r, Zo,d) :
    A = (Zo/60)*(np.sqrt((epsilon_r+1)/2)) + ((epsilon_r-1)/(epsilon_r+1))*(0.
↪23 + (0.11)/epsilon_r)
    B = 377*pi/(2*Zo*np.sqrt(epsilon_r))
    # print(f'A = {A}')
    # print(f'B = {B}')
    W = (8*np.exp(A)/(np.exp(2*A)-2))*d
    if ((W/d) <= 2) & (W>0) :
        #print(f'W/d <= 2')
        return W
    else :
        W = ((2/pi)*(B - 1 - np.log(2*B-1) + ((epsilon_r-1)/(2*epsilon_r))*(np.
↪log(B-1) + 0.39 - (0.61/epsilon_r)) ))*d
        if (W/d >= 2) : return W
        else :
            print(f'calc_MS_width ERROR: Zo = {Zo} and epsilon_r = {epsilon_r}␣
↪invalid. W/d = {W/d}')

#epsilon_r = relative permitivity of dialectric
#W = width of copper microstrip (not groundplane)
#d = substrate thickness
def calc_MS_epsilon_effective(epsilon_r, W, d) :
    return ((epsilon_r+1)/(2) + ((epsilon_r-1)/(2))*(1/(np.sqrt(1 + 12*(d/W)))))

#epsilon_r = relative permitivity of dialectric
#tandelta = property of dialectric
#W = width of copper microstrip (not groundplane)
#d = substrate thickness
def calc_MS_dialectric_loss(epsilon_r,freq, tandelta, W, d) :
    k0 = (2*pi*freq/c)
    epsilon_e = calc_MS_epsilon_effective(epsilon_r, W, d)
    return ((k0*epsilon_r*(epsilon_e-1)*tandelta)/(2*np.
↪sqrt(epsilon_e)*(epsilon_r-1)))

#Rs = surface resistance of conductor
#Zo = characteristic impedence of line.
#W = width of copper microstrip (not groundplane)
def calc_MS_conductor_loss(Rs, Zo, W) :
    return (Rs/(Zo*W))
```

```
#Zo = characteristic impedence of line.
#epsilon_r = relative permitivity of dialectric
#freq = frequency of line.
#d = substrate thickness
#tandelta = property of dialectric
#conductance = conductance of conductor.
def calc_MS_total_loss(Zo, epsilon_r, freq, d, tandelta, conductance) :
    Rs = calc_surface_resistance(freq,conductance)
    W = calc_MS_width(epsilon_r, Zo, d)
    print(f'W = {W}')
    cond_loss = Np_to_dB(calc_MS_conductor_loss(Rs, Zo, W))
    dialectric_loss = Np_to_dB(calc_MS_dialectric_loss(epsilon_r, freq,
 ↪tandelta, W, d))
    print(f'Conductor Loss = {cond_loss} dB/m')
    print(f'Dialectric Loss = {dialectric_loss} dB/m')
    return cond_loss + dialectric_loss
```

0.0.3 EE 5613 Lab 6 Work

[45]:
```
def calc_loaded_Q(Qe,Q0):
    return 1/((1/Qe)+(1/Q0))

frequency = 2*10**9
conductance = 5.817*10**7
Zo = 50

epsilon_r_fr4 = 4.4
tandelta_fr4 = 0.03
W_fr4 = 3.07*10**-3 #3.07 mm
d_fr4 = 0.0014986 #59 mil

epsilon_r_dur = 2.2
tandelta_dur = 0.0005
W_dur = 4.89*10**-3 #4.89 mm
d_dur = 0.0015875 #59 mil

Rs = calc_surface_resistance(frequency,conductance)

dloss_fr4 = calc_MS_dialectric_loss(epsilon_r_fr4,frequency, tandelta_fr4,
 ↪W_fr4, d_fr4)
closs_fr4 = calc_MS_conductor_loss(Rs, Zo, W_fr4)
dloss_dur = calc_MS_dialectric_loss(epsilon_r_dur,frequency, tandelta_dur,
 ↪W_dur, d_dur)
closs_dur = calc_MS_conductor_loss(Rs, Zo, W_dur)

print(f'FR-4')
```

```python
print(f'dialectric loss = {dloss_fr4} Np/m')
print(f'conductor loss = {closs_fr4} Np/m')
print()
print(f'Duroid')
print(f'dialectric loss = {dloss_dur} Np/m')
print(f'conductor loss = {closs_dur} Np/m')

L_oneport_lambda4_fr4 = 2*20.4*10**-3
L_oneport_lambda2_fr4 = 2*40.9*10**-3
L_twoport_lambda4_fr4 = 2*20.53*10**-3 + 3.07*10**-3 + 20.55*2*10**-3
L_twoport_lambda2_fr4 = 2*40.9*10**-3 + + 3.07*10**-3 + 0.0508 #oneport+2000␣
 ↪mils

L_oneport_lambda4_dur = 2*27.47*10**-3
L_oneport_lambda2_dur = 2*54.94*10**-3
L_twoport_lambda4_dur = 2*27.26*10**-3 + 4.89*10**-3 + 24*2*10**-3
L_twoport_lambda2_dur = 2*54.94*10**-3 + 4.89*10**-3 + 0.0508 #oneport+2000 mils

fr4_oneport_lambda4_total_loss = (closs_fr4*L_oneport_lambda4_fr4 +␣
 ↪dloss_fr4*L_oneport_lambda4_fr4)
fr4_oneport_lambda2_total_loss = (closs_fr4*L_oneport_lambda2_fr4 +␣
 ↪dloss_fr4*L_oneport_lambda2_fr4)

fr4_twoport_lambda4_total_loss = (closs_fr4*L_twoport_lambda4_fr4 +␣
 ↪dloss_fr4*L_twoport_lambda4_fr4)
fr4_twoport_lambda4_q0 = pi/
 ↪(2*fr4_twoport_lambda4_total_loss*L_twoport_lambda4_fr4)
fr4_twoport_lambda2_total_loss = (closs_fr4*L_twoport_lambda2_fr4 +␣
 ↪dloss_fr4*L_twoport_lambda2_fr4)
fr4_twoport_lambda2_q0 = pi/
 ↪(2*fr4_twoport_lambda2_total_loss*L_twoport_lambda2_fr4)


dur_oneport_lambda4_total_loss = (closs_dur*L_oneport_lambda4_dur +␣
 ↪dloss_dur*L_oneport_lambda4_dur)
dur_oneport_lambda2_total_loss = (closs_dur*L_oneport_lambda2_dur +␣
 ↪dloss_dur*L_oneport_lambda2_dur)
dur_twoport_lambda4_total_loss = (closs_dur*L_twoport_lambda4_dur +␣
 ↪dloss_dur*L_twoport_lambda4_dur)
dur_twoport_lambda4_q0 = pi/
 ↪(2*dur_twoport_lambda4_total_loss*L_twoport_lambda4_dur)
dur_twoport_lambda2_total_loss = (closs_dur*L_twoport_lambda2_dur +␣
 ↪dloss_dur*L_twoport_lambda2_dur)
dur_twoport_lambda2_q0 = pi/
 ↪(2*dur_twoport_lambda2_total_loss*L_twoport_lambda2_dur)
```

```python
Qe_lambda4 = 1.57
Qe_lambda2 = 6.25*10**-11

fr4_twoport_lambda2_QL = calc_loaded_Q(Qe_lambda2,fr4_twoport_lambda2_q0)
fr4_twoport_lambda4_QL = calc_loaded_Q(Qe_lambda4,fr4_twoport_lambda4_q0)

dur_twoport_lambda2_QL = calc_loaded_Q(Qe_lambda2,dur_twoport_lambda2_q0)
dur_twoport_lambda4_QL = calc_loaded_Q(Qe_lambda4,dur_twoport_lambda4_q0)


print()

print('FR-4')
#dialectric
print(f'single-port lambda/4 dialectric loss = {Np_to_dB(dloss_fr4*L_oneport_lambda4_fr4)} dB')
print(f'single-port lambda/2 dialectric loss = {Np_to_dB(dloss_fr4*L_oneport_lambda2_fr4)} dB')
print(f'two-port lambda/4 dialectric loss = {Np_to_dB(dloss_fr4*L_twoport_lambda4_fr4)} dB')
print(f'two-port lambda/2 dialectric loss = {Np_to_dB(dloss_fr4*L_twoport_lambda2_fr4)} dB')

#conductors
print(f'single-port lambda/4 conductor loss = {Np_to_dB(closs_fr4*L_oneport_lambda4_fr4)} dB')
print(f'single-port lambda/2 conductor loss = {Np_to_dB(closs_fr4*L_oneport_lambda2_fr4)} dB')
print(f'two-port lambda/4 conductor loss = {Np_to_dB(closs_fr4*L_twoport_lambda4_fr4)} dB')
print(f'two-port lambda/2 conductor loss = {Np_to_dB(closs_fr4*L_twoport_lambda2_fr4)} dB')
print(f'single-port lambda/4 total loss = {Np_to_dB(fr4_oneport_lambda4_total_loss)} dB')
print(f'single-port lambda/2 total loss = {Np_to_dB(fr4_oneport_lambda2_total_loss)} dB')
print(f'two-port lambda/4 total loss = {Np_to_dB(fr4_twoport_lambda4_total_loss)} Np')
print(f'two-port lambda/2 total loss = {Np_to_dB(fr4_twoport_lambda2_total_loss)} Np')
print(f'two-port lambda/4 Q0 = {fr4_twoport_lambda4_q0}')
print(f'two-port lambda/2 Q0 = {fr4_twoport_lambda2_q0}')
print(f'two-port lambda/4_QL = {fr4_twoport_lambda4_QL}')
print(f'two-port lambda/2_QL = {fr4_twoport_lambda2_QL}')
print()
```

```python
print('Duroid')
#dialectric
print(f'single-port lambda/4 dialectric loss = 
 ↪{Np_to_dB(dloss_dur*L_oneport_lambda4_dur)} dB')
print(f'single-port lambda/2 dialectric loss = 
 ↪{Np_to_dB(dloss_dur*L_oneport_lambda2_dur)} dB')
print(f'two-port lambda/4 dialectric loss = 
 ↪{Np_to_dB(dloss_dur*L_twoport_lambda4_dur)} dB')
print(f'two-port lambda/2 dialectric loss = 
 ↪{Np_to_dB(dloss_dur*L_twoport_lambda2_dur)} dB')

#conductors
print(f'single-port lambda/4 conductor loss = 
 ↪{Np_to_dB(closs_dur*L_oneport_lambda4_dur)} dB')
print(f'single-port lambda/2 conductor loss = 
 ↪{Np_to_dB(closs_dur*L_oneport_lambda2_dur)} dB')
print(f'two-port lambda/4 conductor loss = 
 ↪{Np_to_dB(closs_dur*L_twoport_lambda4_dur)} dB')
print(f'two-port lambda/2 conductor loss = 
 ↪{Np_to_dB(closs_dur*L_twoport_lambda2_dur)} dB')
print(f'single-port lambda/4 total loss = 
 ↪{Np_to_dB(dur_oneport_lambda4_total_loss)} dB')
print(f'single-port lambda/2 total loss = 
 ↪{Np_to_dB(dur_oneport_lambda2_total_loss)} dB')
print(f'two-port lambda/4 total loss = 
 ↪{Np_to_dB(dur_twoport_lambda4_total_loss)} Np')
print(f'two-port lambda/2 total loss = 
 ↪{Np_to_dB(dur_twoport_lambda2_total_loss)} Np')
print(f'two-port lambda/4 Q0 = {dur_twoport_lambda4_q0}')
print(f'two-port lambda/2 Q0 = {dur_twoport_lambda2_q0}')
print(f'two-port lambda/4_QL = {dur_twoport_lambda4_QL}')
print(f'two-port lambda/2_QL = {dur_twoport_lambda2_QL}')
```

```
FR-4
dialectric loss = 1.0444537492772428 Np/m
conductor loss = 0.07589915738377478 Np/m

Duroid
dialectric loss = 0.012235069394021735 Np/m
conductor loss = 0.04765039124093837 Np/m

FR-4
single-port lambda/4 dialectric loss = 0.3701380079300435 dB
single-port lambda/2 dialectric loss = 0.7420904178597441 dB
two-port lambda/4 dialectric loss = 0.773207412153863 dB
two-port lambda/2 dialectric loss = 1.2307995964673775 dB
single-port lambda/4 conductor loss = 0.026897469550029942 dB
```

```
single-port lambda/2 conductor loss = 0.053926789440991406 dB
two-port lambda/4 conductor loss = 0.05618802278796697 dB
two-port lambda/2 conductor loss = 0.08944067877089615 dB
single-port lambda/4 total loss = 0.3970354774800734 dB
single-port lambda/2 total loss = 0.7960172073007354 dB
two-port lambda/4 total loss = 0.82939543494183 Np
two-port lambda/2 total loss = 1.3202402752382738 Np
two-port lambda/4 Q0 = 193.0101085185949
two-port lambda/2 Q0 = 76.17235482940701
two-port lambda/4_QL = 1.557332209757893
two-port lambda/2_QL = 6.24999999999487e-11

Duroid
single-port lambda/4 dielectric loss = 0.005838609088131469 dB
single-port lambda/2 dielectric loss = 0.011677218176262938 dB
two-port lambda/4 dielectric loss = 0.011414725193960706 dB
two-port lambda/2 dielectric loss = 0.017595531611247314 dB
single-port lambda/4 conductor loss = 0.022738899011745865 dB
single-port lambda/2 conductor loss = 0.04547779802349173 dB
two-port lambda/4 conductor loss = 0.04445549950585409 dB
two-port lambda/2 conductor loss = 0.06852711156488467 dB
single-port lambda/4 total loss = 0.028577508099877337 dB
single-port lambda/2 total loss = 0.057155016199754674 dB
two-port lambda/4 total loss = 0.0558702246998148 Np
two-port lambda/2 total loss = 0.08612264317613197 Np
two-port lambda/4 Q0 = 2273.573765787842
two-port lambda/2 Q0 = 956.8308845081689
two-port lambda/4_QL = 1.5689165959368963
two-port lambda/2_QL = 6.249999999999591e-11
```