Equivalent Circuits

Lee Farrugia

10th November 2022

1 Abstract

yes

2 Introduction & Theoretical Background

The use of open-ended coaxial probes to measure the complex permittivity is used as it is a non invasive and non destructive to the materials being checked. However, those are not the only advantages this method has, but also due to the way it is utilised it allow for bandwidth measurement and easy sampling (Liao et al., 2011). This method can be used to measure the microwave complex permittivity of dielectric materials including lossy dielectrics. As the relationship between the reflection and transmission is quite complex, the equivalent circuit method is used to simplify the relationship (Stuchly et al., 1982). This is quite useful when dealing with lossy dielectrics and biological tissues, as the biological tissues are not only made up from one type of material but, multiple types of tissues in one sample. Another example of the use of this method is explored in Zajíček et al., 2006, where they discuss the relationship of the the complex permittivity and the transmission of healthy tissues and tumour tissues. They further discuss how this method could be used to image people with non-ionising radiation.

3 Materials & Methods

3.1 Language and Packages

Python 3.10.8, Numpy, Pandas, Matplotlib.pyplot .

3.2 Methodology

1. The data provided for both methanol and sodium chloride was imported into the program and each sheet was given its own `pandas` data frame.
2. The average for the real and imaginary parts at each frequency was calculated for each standard material, i.e. air, short, di-ionised water.
3. The averages calculated in the previous were used to calculate the ρ of each material. These were then used to calculate the Δs found the equation.
4. The equation used was:

$$\varepsilon_m = -\frac{\Delta_{m2}\Delta_{13}}{\Delta_{m1}\Delta_{32}}\varepsilon_3 - \frac{\Delta_{m3}\Delta_{21}}{\Delta_{m1}\Delta_{32}}\varepsilon_1 \tag{1}$$

5. The average for the real part and imaginary part of the data for the complex permittivity was calculated.
6. The obtained values for the complex permittivity and the averages of the data given were plotted against each other.
7. Specific points of the data were printed in order to be compared to each other.

4 Results & Discussion

In Marsland and Evans, 1987 the error correction was described as follows:

- Calculate **E**.
- Use the calculated **E** to correct subsequent error that arise from the measurements.

However, as **E** is not necessary one can substitute the three known admittance values (of air, short and deionised water) along with their respective reflective-coefficient thus, cancelling out the **E** terms.

As can be seen in Marsland and Evans, 1987 the admittance of the probe can be modelled in two ways:

$$y(\omega, \varepsilon_r) = G_0 Z_0 \varepsilon_r^{\frac{5}{2}} + j\omega Z_0(\varepsilon_r C_0 + C_p), \tag{2}$$

$$y(\omega, \varepsilon_r) = \omega Z_0(\varepsilon_r C_0 + C_f)\,, \tag{3}$$

called the admittance model 1 and admittance model 2 respectively, where ω is the angular frequency, ε_r is the relative permittivity of the material in which the probe is embedded/immersed in, G_0 is the free-space radiation conductance, $\varepsilon_r C_0$ is the capacitance which represents the fringing field in the external dielectric material, C_p is the capacitance which represents the fringing field in the Teflon dielectric of the cable, which usually is $C_0 >> C_p$. However, only the admittance model 1 is considered in Marsland and Evans, 1987. Applying a simple linear transformation to equation 2, we obtain:

$$y'(\omega, \varepsilon_r) = \left(\frac{1}{j\omega C_0 Z_0}\right) \cdot y(\omega, \varepsilon_r) - \left(\frac{C_f}{C_0}\right), \tag{4}$$

and using equation 2:

$$y'(\omega, \varepsilon_r) = \varepsilon_r + G_n \varepsilon_r^{\frac{5}{2}}, \tag{5}$$

where $G_n = \frac{G_0}{j\omega C_0}$. This satisfies the linear transformation due to the cross-ratio invariance, which can be further expanded to:

$$\frac{(\rho_m - \rho_1)(\rho_3 - \rho_1)}{(\rho_m - \rho_2)(\rho_1 - \rho_3)} = \frac{(y'_m - y'_1)(y'_3 - y'_2)}{(y'_m - y'_2)(y'_1 - y'_2)}\,. \tag{6}$$

Expanding equation 6 to its 5$^{\text{th}}$ order polynomial, we obtain:

$$G_n \varepsilon_m^{\frac{5}{2}} + \varepsilon_m + \left[\frac{\Delta_{m1}\Delta_{32} y'_3 y'_2 + \Delta_{m2}\Delta_{13} y'_1 y'_3 + \Delta_{m3}\Delta_{21} y'_2 y'_1}{\Delta_{m1}\Delta 32 y'_1 + \Delta_{m2}\Delta 12 y'_2 + \Delta_{m3}\Delta 21 y'_3}\right] = 0\,. \tag{7}$$

By applying the short-circuit simplification and setting the termination 1 as a short circuit we obtain:

$$G_n \varepsilon_m^{\frac{5}{2}} + \varepsilon_m + \left[\frac{\Delta_{m2}\Delta_{13}}{\Delta_{m1}\Delta_{32}} y'_3 + \frac{\Delta_{m3}\Delta_{21}}{\Delta_{m1}\Delta_{32}} y'_2\right] = 0 \tag{8}$$

as $y'_1 \to \infty$. By omitting the G_n in equation 8 and in $y'(\omega, \varepsilon_r) = \varepsilon_r + G_n \varepsilon_r^{\frac{5}{2}}$ we obtain:

$$\varepsilon_m = -\frac{\Delta_{m2}\Delta_{13}}{\Delta_{m1}\Delta_{32}}\varepsilon_3 - \frac{\Delta_{m3}\Delta_{21}}{\Delta_{m1}\Delta_{32}}\varepsilon_2\,. \tag{9}$$

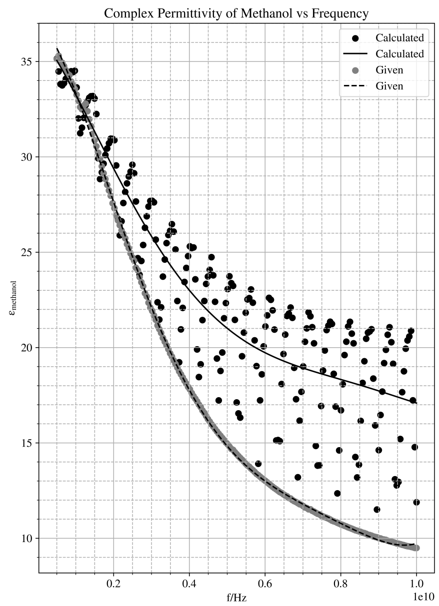

Figure 1: A graph of the complex permittivity at each frequency for Methanol

Frequency/Hz $\times 10^8$	$\varepsilon_{\text{calculated}}$	$\varepsilon_{\text{given}}$
5.00	$35.15 + 3.22j$	$35.13 - 5.07j$
40.15	$25.31 + 6.19j$	$17.81 - 13.83j$
87.65	$20.85 + 0.76j$	$10.17 - 8.99j$
100.00	$11.89 + 1.19j$	$9.49 - 8.14j$

Table 1: A table of four different data points for methanol

Frequency/Hz $\times 10^8$	$\varepsilon_{\text{calculated}}$	$\varepsilon_{\text{given}}$
5.00	$82.94 + 41.57j$	$86.41 - 48.50j$
40.15	$80.60 + 4.30j$	$82.00 - 21.69j$
87.65	$80.93 + 1.69j$	$71.63 - 32.25j$
100.00	$81.91 + 2.70j$	$68.40 - 34.31j$

Table 2: A table of four different data points for NaCl

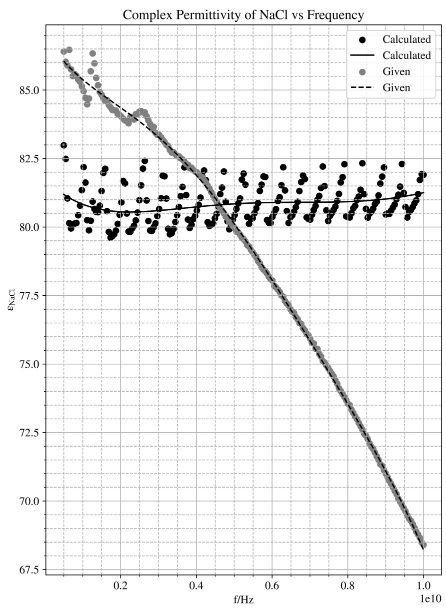

Figure 2: A graph of the complex permittivity at each frequency for Sodium Chloride

5 References

Liao, K.-m., Wu, Y.-q., Qian, C., & Du, G.-p. (2011). An accurate equivalent circuit method of open ended coaxial probe for measuring the permittivity of materials. In *Electrical power systems and computers* (pp. 779–784). Springer.

Marsland, T., & Evans, S. (1987). Dielectric measurements with an open-ended coaxial probe. *IEE Proceedings H (Microwaves, Antennas and Propagation)*, *134*(4), 341–349.

Stuchly, M. A., Brady, M. M., Stuchly, S. S., & Gajda, G. (1982). Equivalent circuit of an open-ended coaxial line in a lossy dielectric. *IEEE Transactions on instrumentation and Measurement*, (2), 116–119.

Zajíček, R., Vrba, J., & Novotnỳ, K. (2006). Evaluation of a reflection method on an open-ended coaxial line and its use in dielectric measurements. *Acta Polytechnica*, *46*(5).

6 Appendix

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# importing data to different dataframes for each sheet
data_air = pd.read_excel('Data.xlsx', 0)
data_short = pd.read_excel('Data.xlsx', 1)
data_water = pd.read_excel('Data.xlsx', 2)
data_methanol = pd.read_excel('Data.xlsx', 3).dropna()
data_NaCl = pd.read_excel('Data.xlsx', 4).dropna()
e_methanol = pd.read_excel('Data.xlsx', 5).dropna()
e_NaCl = pd.read_excel('Data.xlsx', 6).dropna()

# defining the frequency variable
frequency = np.array(data_air['%freq[Hz]'])

# finding the average of the real part of air
air_real_avg = data_air[['Real (S11) R1', 'Real (S11) R2', 'Real
↪  (S11) R3', 'Real (S11) R4' , 'Real (S11) R5', 'Real (S11)
↪  R6']].mean(axis=1)
data_air['Real Avg'] = air_real_avg

# finding the average of the imaginary part of air
air_imaginary_avg = data_air[['Imag (S11) R1', 'Imag (S11) R2',
↪  'Imag (S11) R3', 'Imag (S11) R4' , 'Imag (S11) R5', 'Imag (S11)
↪  R6']].mean(axis=1)
data_air['Imag Avg'] = air_imaginary_avg

# finding the average of the real part of water
```

```
water_real_avg = data_water[['Real (S11) R1', 'Real (S11) R2', 'Real
↪  (S11) R3', 'Real (S11) R4' , 'Real (S11) R5', 'Real (S11)
↪  R6']].mean(axis=1)
data_water['Real Avg'] = water_real_avg

# finding the average of the imaginary part of water
water_imaginary_avg = data_water[['Imag (S11) R1', 'Imag (S11) R2',
↪  'Imag (S11) R3', 'Imag (S11) R4' , 'Imag (S11) R5', 'Imag (S11)
↪  R6']].mean(axis=1)
data_water['Imag Avg'] = water_imaginary_avg

# finding the average of the real part of methanol
methanol_real_avg = data_methanol[['Real (S11) R1', 'Real (S11) R2',
↪  'Real (S11) R3', 'Real (S11) R4' , 'Real (S11) R5', 'Real (S11)
↪  R6']].mean(axis=1)
data_methanol['Real Avg'] = methanol_real_avg

# finding the average of the imaginary part of methanol
methanol_imaginary_avg = data_methanol[['Imag (S11) R1', 'Imag (S11)
↪  R2', 'Imag (S11) R3', 'Imag (S11) R4' , 'Imag (S11) R5', 'Imag
↪  (S11) R6']].mean(axis=1)
data_methanol['Imag Avg'] = methanol_imaginary_avg

# finding the average of the real part of NaCl
NaCl_real_avg = data_NaCl[['Real (S11) R1', 'Real (S11) R2', 'Real
↪  (S11) R3', 'Real (S11) R4' , 'Real (S11) R5', 'Real (S11)
↪  R6']].mean(axis=1)
data_NaCl['Real Avg'] = NaCl_real_avg

# finding the average of the imaginary part of NaCl
NaCl_imaginary_avg = data_NaCl[['Imag (S11) R1', 'Imag (S11) R2',
↪  'Imag (S11) R3', 'Imag (S11) R4' , 'Imag (S11) R5', 'Imag (S11)
↪  R6']].mean(axis=1)
data_NaCl['Imag Avg'] = NaCl_imaginary_avg

# finding the average of the given real permittivity of methanol
e_methanol_real_avg =
↪  e_methanol[["e'_R1","e'_R2","e'_R3"]].mean(axis=1)
e_methanol['Real Avg'] = e_methanol_real_avg

# finding the average of the given imaginary permittivity of
↪  methanol
e_methanol_imaginary_avg =
↪  e_methanol[["e''_R1","e''_R2","e''_R3"]].mean(axis=1)
e_methanol['Imag Avg'] = e_methanol_imaginary_avg

# finding the average of the given real permittivity of NaCl
e_NaCl_real_avg = e_NaCl[["e'_R1","e'_R2","e'_R3"]].mean(axis=1)
```

```
e_NaCl['Real Avg'] = e_NaCl_real_avg

# finding the average of the given imaginary permittivity of NaCl
e_NaCl_imaginary_avg =
↪ e_NaCl[["e''_R1","e''_R2","e''_R3"]].mean(axis=1)
e_NaCl['Imag Avg'] = e_NaCl_imaginary_avg

# defining the different perimittivity as numpy array
short_r_avg = data_short['re:Trc1_S11'].to_numpy()
short_i_avg = data_short['im:Trc1_S11'].to_numpy()

air_r_avg = data_air['Real Avg'].to_numpy()
air_i_avg = data_air['Imag Avg'].to_numpy()

water_r_avg = data_water['Real Avg'].to_numpy()
water_i_avg = data_water['Imag Avg'].to_numpy()

methanol_r_avg = data_methanol['Real Avg'].to_numpy()
methanol_i_avg = data_methanol['Imag Avg'].to_numpy()

NaCl_r_avg = data_NaCl['Real Avg'].to_numpy()
NaCl_i_avg = data_NaCl['Imag Avg'].to_numpy()

e_methanol_r_avg = e_methanol['Real Avg'].to_numpy()
e_methanol_i_avg = e_methanol['Imag Avg'].to_numpy()

e_NaCl_r_avg = e_NaCl['Real Avg'].to_numpy()
e_NaCl_i_avg = e_NaCl['Imag Avg'].to_numpy()

# creating the complex numbers from the averages
air_complex = air_r_avg - (1j * air_i_avg)
short_complex = short_r_avg - (1j * short_i_avg)
water_complex = water_r_avg - (1j * water_i_avg)
methanol_complex = methanol_r_avg - (1j * methanol_i_avg)
NaCl_complex = NaCl_r_avg - (1j * NaCl_i_avg)
e_methanol_c = e_methanol_r_avg - (1j * e_methanol_i_avg)
e_NaCl_c = e_NaCl_r_avg - (1j * e_NaCl_i_avg)

# calculating the different deltas required in the equation
delta_13 = np.subtract(short_complex, water_complex)
delta_21 = np.subtract(air_complex, short_complex)
delta_23 = np.subtract(air_complex, water_complex)
delta_32 = np.subtract(water_complex, air_complex)
delta_m1_methanol = np.subtract(methanol_complex, short_complex)
delta_m2_methanol = np.subtract(methanol_complex, air_complex)
delta_m3_methanol = np.subtract(methanol_complex, water_complex)
delta_m1_NaCl = np.subtract(NaCl_complex, short_complex)
delta_m2_NaCl = np.subtract(NaCl_complex, air_complex)
```

```
delta_m3_NaCl = np.subtract(NaCl_complex, water_complex)

# using the equation and values to calculate the complex
↪ permittivity
e_methanol = -1 * (((delta_m2_methanol * delta_13) /
↪ (delta_m1_methanol * delta_32)) * 80.5) - (((delta_m3_methanol *
↪ delta_21) / (delta_m1_methanol * delta_32)) * 1)
e_NaCl = -1 * (((delta_m2_NaCl * delta_13) / (delta_m1_NaCl *
↪ delta_32)) * 80.5) - (((delta_m3_NaCl * delta_21) /
↪ (delta_m1_NaCl * delta_32)) * 1)

# taking the real parts only
methanol_real = np.real(e_methanol)
e_methanol_real = np.real(e_methanol_c)
NaCl_real = np.real(e_NaCl)
e_NaCl_real = np.real(e_NaCl_c)

# polyfit calculated methanol
coefficients, cov = np.polyfit(frequency, methanol_real, 5, cov=True)
poly_function = np.poly1d(coefficients)
trendline_c_m = poly_function(frequency)

# polyfit given methanol
coefficients, cov = np.polyfit(frequency, e_methanol_real, 5,
↪ cov=True)
poly_function = np.poly1d(coefficients)
trendline_g_m = poly_function(frequency)

# polyfit calculated NaCl
coefficients, cov = np.polyfit(frequency, NaCl_real, 5, cov=True)
poly_function = np.poly1d(coefficients)
trendline_c_n = poly_function(frequency)

# polyfit given NaCl
coefficients, cov = np.polyfit(frequency, e_NaCl_real, 5, cov=True)
poly_function = np.poly1d(coefficients)
trendline_g_n = poly_function(frequency)

# value comparisons
positions = (0, 74, 174, 200)
for i in positions:
    print(f'The calculated permittivity at data point {i+1} for
    ↪ methanol is:{methanol_complex[i]:.2f}, while the given
    ↪ value is: {e_methanol_c[i]:.2f}')
    print(f'The calculated permittivity at data point {i+1} for
    ↪ NaCl is:{NaCl_complex[i]:.2f}, while the given value is:
    ↪ {e_NaCl_c[i]:.2f}')
```

```
# plotting the data for methanol
plt.figure(figsize=(7.5, 10.5))
plt.rcParams['font.family'] = 'STIXGeneral'
plt.rcParams['mathtext.fontset'] = 'stix'
plt.rcParams['font.size'] = 12
plt.rcParams['font.weight'] = 'normal'
plt.minorticks_on()
plt.grid(visible=True, which='major', linestyle='-')
plt.grid(visible=True, which='minor', linestyle='--')

plt.scatter(frequency, methanol_real, color='k', label='Calculated')
plt.plot(frequency, trendline_c_m, color = 'k', label='Calculated')
plt.scatter(frequency, e_methanol_real, color='grey', label='Given')
plt.plot(frequency, trendline_g_m, '--',  color = 'k', label =
↪  'Given')
plt.xlabel('f/Hz')
plt.ylabel(r'$\mathrm{\epsilon_{methanol}}$')
plt.title('Complex Permittivity of Methanol vs Frequency')
plt.legend()
plt.savefig('Plot1.pdf', dpi=800)
plt.show()
plt.close()

# plotting the data of NaCl
plt.figure(figsize=(7.5, 10.5))
plt.rcParams['font.family'] = 'STIXGeneral'
plt.rcParams['mathtext.fontset'] = 'stix'
plt.rcParams['font.size'] = 12
plt.rcParams['font.weight'] = 'normal'
plt.minorticks_on()
plt.grid(visible=True, which='major', linestyle='-')
plt.grid(visible=True, which='minor', linestyle='--')

plt.scatter(frequency, NaCl_real, color='k', label='Calculated')
plt.plot(frequency, trendline_c_n, color = 'k', label='Calculated')
plt.scatter(frequency, e_NaCl_real, color='grey', label='Given')
plt.plot(frequency, trendline_g_n, '--',  color = 'k', label =
↪  'Given')
plt.xlabel('f/Hz')
plt.ylabel(r'$\mathrm{\epsilon_{NaCl}}$')
plt.title('Complex Permittivity of NaCl vs Frequency')
plt.legend()
plt.savefig('Plot2.pdf', dpi=800)
plt.show()
```