**Week 7: 17/11/2021 – Wednesday**

1. **Outline of Meeting**

The meeting was carried out on Zoom. The meeting was dedicated to explaining how we move forward with the model accomplished from the previous week. The project now moves onto reading out the measurement from the detector. This was accomplished by using the dF0 formula. Further details will be explained in the Outline of theory and methodology section. It essentially allows us to relate the detector output of S21 and the signal to measure.

1. **Specification of Tasks**
2. Add resistance to the ABCD KID model you have already made
3. Take your model, and make Q vs I plots for the resonator at each temperature (I= real part of S21, Q= imaginary part of S21)
4. For your lowest temperature KID, note F0 and the I and Q values at F0. Lets call this F0 F0\_base
5. add the I and Q values of each KID at the frequency F0\_base to your Q vs I sweep plots as a symbol.
6. Make another plot of F0 vs temperature. Calculate F0 of each resonance as from the frequency corresponding to  the minimum in S21 magnitude.
7. Using the “Magic formula” equation equation 6 from the understanding KID readout document, plot the F0 calculated from this formula. Do this by looking at I and Q at F0 base and calculating df0 from the formula. The dI/dF and dQ/dF data should be calculated from the lowest temperature sweep. This will tell use the range over which this formula is valid
8. calculate di/df and dQ by df at F0 for the lowest temperature sweep these will be single values
9. on subsequent S21 data (taken at higher temperatures, plug in the vales for dI this is the change in I In the new temperature from the I at F0\_base. Do the same for Q and this should give you a change in F0
10. **Outline of Theory and Methodology**

A KID typically measures the signal and outputs a I and Q value in units of Volts. This may not be entirely intuitive, as a voltage from an electronic circuit does not give much information in terms of its response to the detection.

The solution to this is therefore known as the formula. This formula allows us to convert I and Q values from the Sweep data (Data across the frequency domain) into a change in tone frequency . The formula is given as follows:

where and are the changes in I and Q values from the time stream data against the I and Q from the Sweep data, for a given time. In the case of the model, it is the change in I and Q for a given temperature compared to the base temperature at the tone frequency. and are the numerical derivates of the minimum of the sweep data (For the model, the base temperature). This allows us to calculate a value for the change in the tone frequency.

Moving on to the task, we first plot the I and Q values as given in section 4, the cross represents the I and Q value at the tone frequency, we can clearly observe that the I and Q values have changed.

Following this, we can find and by finding the numerical derivate of the minimum of S21 at the base temperature. Then, we can find the change in I and Q values and for temperatures above the base temperature by subtracting the I and Q values for the respective temperature from the I and Q of the base temperature, at the tone-frequency. (e.g. if we set tone at 0.95 GHz, find the I and Q values at 0.95GHz for both the base and new temperature and subtract each other).

Then, we can use these values and plug them into the dF0 formula to obtain a dF0. We can subtract dF0 for each temperature from F0 and plot F0 against temperature. We can also find F0 for the minimums of S21 for varying temperatures and plot this along with dF0 formula. This is shown in section 5. The graph shows that the formula holds remarkably well up till 0.21K. As such, the formula is appropriate for low temperature variations, therefore the maximum change in for the approach to still be valid is .

Important to note, the dF0 formula is quite important for characterising the response of the detector. This is because the dF0 allows the conversion of I and Q values from the output of the KID into a change in F0. We can then characterize the response as a change in F0 against a change in frequency, . This will be explored when we come to responsivity measurements in semester 2.

1. **Plot of I vs Q**Diagram

   Description automatically generated with medium confidence
2. **Plot of F0 against Temperature for dF0 formula and Minimum of S21**

Chart, line chart

Description automatically generated

1. Python Code for Task

#imports

import numpy as np

import matplotlib.pyplot as plt

import scipy.constants as const

from scipy.special import iv as I0

from scipy.special import kv as K0

#Define Global Variables

L\_geo = 55.6e-9

Z0 = 50.0

F0\_base = 0.95e9 #At lowest Temp

squares= 27223

c\_couple = 1.5e-14

TC = 1.5

Delta\_0 = (3.5\*const.Boltzmann\*TC)/2

sigma\_n = 6.0e7 # Normal stae conductvity if superconducting film

Thick = 20e-9 # Thickness of superconducting fil

w = 2 \* np.pi \* F0\_base

me = const.m\_e

miu\_0 = 4\*np.pi\*10\*\*-7

pi = np.pi

#Main code

def main():

#Define temperature range with step 0.01K

step = 0.02

temp = np.arange(0.20, 0.35, step)

#Find sigma1 and sigma 2 and Lint

sigma1, sigma2 = find\_sigma1\_sigma2(sigma\_n ,Thick, TC, Delta\_0, w, temp)

Lint = find\_Lint\_square(Thick, w, sigma2) \* squares

#Find lk

Lk = find\_lk(Thick, w, sigma2)

#Find Res

sigma12Ratio = sigma1/sigma2

Res = Lk\*w\*sigma12Ratio \*squares

#IDC for Lowest Temp (0.2K)

Ltot\_lowest = Lint[0] + L\_geo

IDC = find\_IDC(w, Ltot\_lowest, c\_couple)

#Find S21

Sweep\_points = 20000

BW = 5e6

I\_raw = np.zeros((Sweep\_points, len(temp)), dtype="float")

Q\_raw = np.copy(I\_raw)

Phase = np.copy(Q\_raw)

S21\_Volt = np.copy(I\_raw)

for i in range(0, len(Lint)):

Sweep, S21\_Volt[:,i], Phase[:,i], I\_raw[:,i], Q\_raw[:,i],\_,\_,\_,\_,\_ = Capacitive\_Res\_Sim(F0\_base, c\_couple, Z0, L\_geo, Lint[i], Res[i], BW, Sweep\_points, IDC)

plt.plot(Sweep/1e9, S21\_Volt[:,i], label=str("{:.2f}".format(temp[i])))

#Graph labels and title

plt.legend(loc='center left', bbox\_to\_anchor=(1, 0.5), fancybox=True, title="Temperature / K")

plt.xlabel('Frequency / GHz', fontsize=13)

plt.ylabel('S21 Amplitude / V', fontsize=13);

plt.title("S21 Amplitude For Varying Temperatures")

plt.xlim(0.9490, 0.9505)

plt.locator\_params(nbins=6)

plt.savefig("S21 Plot with Resistance")

plt.rcParams['figure.dpi'] = 300

plt.figure()

#Q vs I plots

for i in range(0, len(Lint)):

plt.plot(I\_raw[:,i], Q\_raw[:,i], linewidth=1,label=str("{:.2f}".format(temp[i])))

#Minimum S21 at lowest temp

S21\_Base = min(S21\_Volt[:,0])

I\_Base = np.zeros(len(temp), dtype="float")

Q\_Base = np.copy(I\_Base)

#Obtain F0\_base and I and Q values for Lowest Temp

for i in range(0, len(S21\_Volt[:,0])):

if S21\_Base == S21\_Volt[i,0]:

F0\_Base = Sweep[i]

#Plot I and Q values at F0\_Base

for i in range(0, len(temp)):

for j in range(0, len(Sweep)):

if F0\_Base == Sweep[j]:

I\_Base[i] = I\_raw[j,i]

Q\_Base[i] = Q\_raw[j,i]

plt.plot(I\_Base[i], Q\_Base[i], markersize=4, marker="x", color='black')

#labels

plt.legend(loc='center left', bbox\_to\_anchor=(1, 0.5), fancybox=True, title="Temperature / K")

plt.xlabel('I / V', fontsize=13)

plt.ylabel('Q / V', fontsize=13);

#plt.title("Q vs I Plot for Varying Temperature")

plt.savefig("Q vs I plot for varying temp")

plt.figure()

#Finding F0 for the different Temperatures

F0 = np.zeros(len(temp))

for i in range(0, len(temp)):

S21\_min = min(S21\_Volt[:,i])

for j in range(0, len(Sweep)):

if S21\_min == S21\_Volt[j,i]:

F0[i] = Sweep[j]

#Plotting F0 vs Temp

plt.plot(temp, F0/1e9, color='k', linewidth="1", label="Minimum Of S21")

plt.xlabel('Temperature / K', fontsize=13)

plt.ylabel('F0 / GHz', fontsize=13);

plt.rcParams['figure.dpi'] = 300

#plt.title("F0 vs Temperature")

#Finding dI/dF and dQ/dF for lowest temperature

#Using numerical derivatives

step = abs((Sweep[0]-Sweep[-1])/Sweep\_points)

for i in range(0, len(Sweep)):

if Sweep[i] == F0\_Base:

didf = (I\_raw[i+1,0] - I\_raw[i-1,0])/(2\*step)

dqdf = (Q\_raw[i+1,0] - Q\_raw[i-1,0])/(2\*step)

#Use Magic Formula

di = np.zeros(len(temp))

dq = np.copy(di)

di = abs(I\_Base - I\_Base[0])

dq = abs(Q\_Base - Q\_Base[0])

dF0 = Magic\_Formula(di, dq, didf, dqdf)

#Find F0 for different temp

F0\_Magic = F0\_Base - abs(dF0)

plt.plot(temp, F0\_Magic/1e9, label="dF0 Formula")

plt.legend(loc='center left', bbox\_to\_anchor=(1, 0.5), fancybox=True)

plt.ticklabel\_format(useOffset=False)

plt.rcParams['figure.dpi'] = 1000

plt.xlim(0.20, 0.22)

plt.ylim(0.949980, 0.95)

plt.savefig("Magic Formula plot")

#KID Simulating Function

def Capacitive\_Res\_Sim(F0, C\_couple, Z0, L\_geo, L\_int, Res, Sweep\_BW, Sweep\_points, Capacitance):

""" Help file here"""

j=complex(0,1)

Cc=C\_couple

F\_min=F0-(Sweep\_BW/2.0)

F\_max=F0+(Sweep\_BW/2.0)

Sweep=np.linspace(F\_min, F\_max, Sweep\_points)

W=Sweep\*2.0\*pi

W0=2.0\*pi\*F0

L=L\_geo+L\_int

C=Capacitance

Zres= 1.0/((1./((j\*W\*L)+Res))+(j\*W\*C)) # Impedance of resonator section

Zc=1.0/(j\*W\*Cc) #impedance of coupler

ZT=Zres+Zc

YT=1.0/ZT

S21 = 2.0/(2.0+(YT\*Z0))

I\_raw=S21.real

Q\_raw=S21.imag

shift=((1.0-min(I\_raw))/2.0)+min(I\_raw)

I\_cent=I\_raw-shift

Q\_cent=Q\_raw

Phase=Atan(abs(Q\_cent/I\_cent))

QU=(W0\*L)/Res

QL=(C\*2)/(W0\*(Cc\*\*2)\*Z0)

S21\_Volt=abs(S21)

I\_offset=shift

return (Sweep, S21\_Volt, Phase, I\_raw, Q\_raw, I\_cent, Q\_cent, QU, QL, I\_offset)

#Function to find sigma1 and sigma2

def find\_sigma1\_sigma2(sigma\_n ,Thick, TC, Delta\_0, w, T):

#An interpolation formula for delta\_T

delta\_T = Delta\_0\*np.tanh(1.74\*np.sqrt((TC/T)-1))

#Define constants to simplify eqn

multiplying\_constant = delta\_T/(const.hbar \* w)

e\_const\_1 = - Delta\_0/(const.Boltzmann\*T)

e\_const\_2 = (const.hbar\*w)/(2\*const.Boltzmann\*T)

#Parts of the sigma1 Ratio

A = 2\*multiplying\_constant

B = np.exp(e\_const\_1)

C = K0(0, e\_const\_2)

D = 2\*(np.sinh(e\_const\_2))

#Find Sigma 1 and Sigma 2

sigma1Ratio = A \* B \* C \* D

sigma2Ratio = np.pi\*multiplying\_constant\*(1 - (2\*np.exp(e\_const\_1)\*np.exp(-e\_const\_2)\*I0(0,e\_const\_2)))

sigma2 = sigma2Ratio \* sigma\_n

sigma1 = sigma1Ratio \* sigma\_n

return sigma1, sigma2

def find\_lk(Thick, w, sigma2):

#Depth

lower\_fraction = miu\_0\*sigma2\*w

Lambda\_T\_MB = (1/lower\_fraction)\*\*0.5

fraction = Thick/(2\*Lambda\_T\_MB)

#Terms for lk

A = (miu\_0\*Lambda\_T\_MB)/4

B = coth(fraction)

C = fraction\*(csch(fraction))\*\*2

#R vs T

lk = A\*(B+C)

return lk

def find\_Lint\_square(Thick, w, sigma2):

#Depth

lower\_fraction = miu\_0\*sigma2\*w

Lambda\_T\_MB = (1/lower\_fraction)\*\*0.5

#Internal Inductance

fraction = Thick/(2\*Lambda\_T\_MB)

L\_int = (miu\_0\*Lambda\_T\_MB/2)\*coth(fraction)

return L\_int

#Define coth and csch

def coth(x):

return np.cosh(x)/np.sinh(x)

def csch(x):

return 1/np.sinh(x)

def Atan(x):

return np.arctan(x)

#Find IDC function

def find\_IDC(w0, Ltot, Cc):

IDC = 1/((w0\*\*2)\*Ltot) - Cc

return IDC

def Magic\_Formula(di, dq, didf, dqdf):

return (di\*didf + dq\*dqdf)/(didf\*\*2 + dqdf\*\*2)

main()