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.

2. **Specification of Tasks**

i) Add resistance to the ABCD KID model you have already made

ii) 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)

iii) For your lowest temperature KID, note F0 and the I and Q values at F0. Lets call this F0 F0_base

iv) add the I and Q values of each KID at the frequency F0_base to your Q vs I sweep plots as a symbol.

v) Make another plot of F0 vs temperature. Calculate F0 of each resonance as from the frequency corresponding to the minimum in S21 magnitude.

vi) 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

vii) calculate di/df and dQ by df at F0 for the lowest temperature sweep these will be single values

viii) 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

3. **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 $dF0$ 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 F_0. The formula is given as follows:

$$\partial F_0 = \frac{\partial I(t)\frac{\partial I}{\partial F} + \partial Q(t)\frac{\partial Q}{\partial F}}{\left(\frac{\partial I}{\partial F}\right)^2 + \left(\frac{\partial Q}{\partial F}\right)^2}$$

where $\partial I(t)$ and $\partial Q(t)$ 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. $\left(\frac{\partial I}{\partial F}\right)$ and $\left(\frac{\partial Q}{\partial F}\right)$ 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.

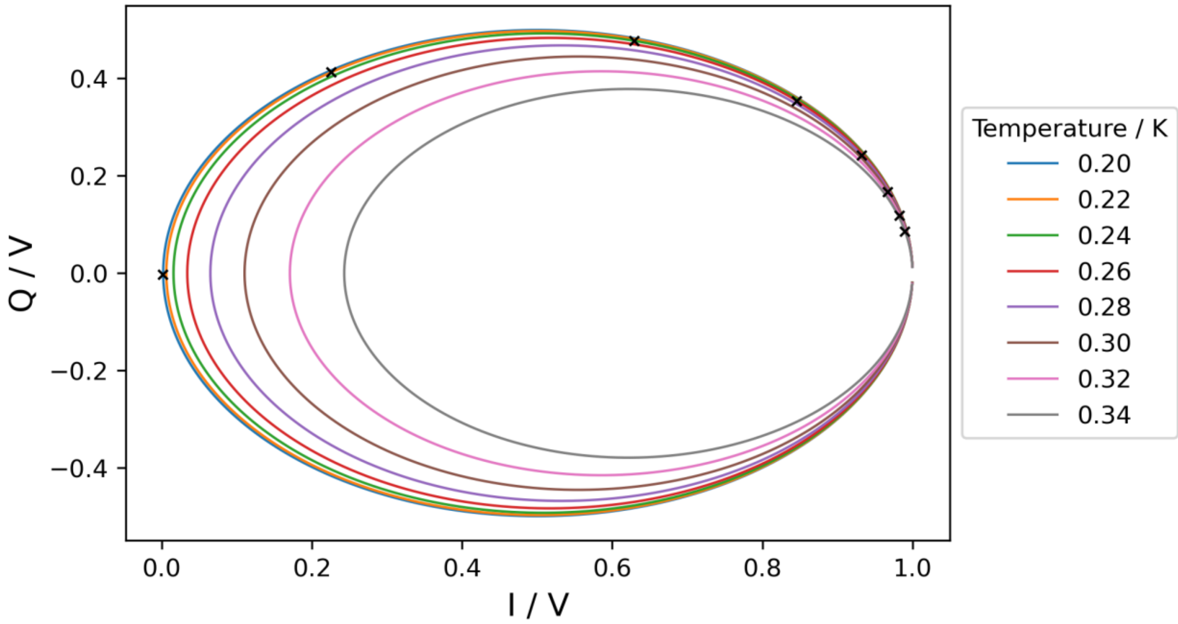
Moving on to the task, we first plot the I and Q values as given in section 4, the cross represents the I and Q value at the tone frequency, we can clearly observe that the I and Q values have changed.

Following this, we can find $\left(\frac{\partial I}{\partial F}\right)$ and $\left(\frac{\partial Q}{\partial F}\right)$ 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 $\partial I(t)$ and $\partial Q(t)$ 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).

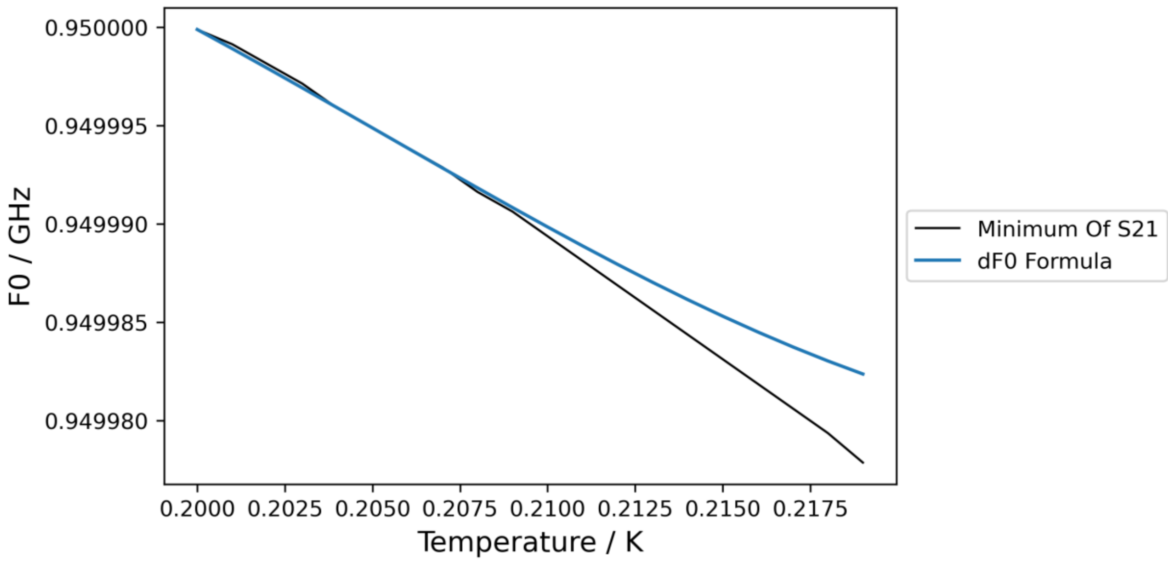
Then, we can use these values and plug them into the dF0 formula to obtain a dF0. We can subtract dF0 for each temperature from F0 and plot F0 against temperature. We can also find F0 for the minimums of S21 for varying temperatures and plot this along with dF0 formula. This is shown in section 5. The graph shows that the ∂F_0 formula holds remarkably well up till 0.21K. As such, the ∂F_0 formula is appropriate for low temperature variations, therefore the maximum change in F_0 for the approach to still be valid is $\pm\ 10000Hz$.

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, $dF0/dP$. This will be explored when we come to responsivity measurements in semester 2.

4. Plot of I vs Q



5. Plot of F0 against Temperature for dF0 formula and Minimum of S21



6. 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()
```