Analytic Performance Estimator

Summary:

This program summarizes most of my work on the small KID's project. In here the most of the quasi analytical model that i use to predict/estimate performance of cetain set of parameters.

Table of Contents

Setup

```
%% Setup
%clc
close all
%clear
tic
%clear all %clear classes % Increasingly strong statements about clearing
%everything
%-Useful Physics constants
phyconst.c = 299792458 ;% [m/s] - Speed of light!

% Parameters
Z0 = 79.605;%[Ohm] --> value from CPW simulation sonnet. varies only slightly as function of fr
Z0_PEC =65.3696 ;%%[Ohm] --> value from CPW simulation sonnet. varies only slightly as function
epsilon_eff = 10.6008;%[-] --> value from CPW simulation sonnet. varies only slightly as functi
epsilon_eff_PEC = 7.1484;%[-] --> value from CPW simulation sonnet. varies only slightly as fun
epsilon0 = 8.854187E-12;% [C/m]
length_M = 0.001;%[m]
d = 250E-9;%[m]

%Data path
addpath('.\Sonnet_data')
filename_begin = 'PPCV0_9_9A';
filename_end = '.csv';
filename_CPW_SC = 'AlHybridV0_0_3_SC.csv';
filename_CPW_PEC = 'AlHybridV0_0_3_PEC.csv';
```

Loading data and making objects

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sed mi mauris. Duis scelerisque, mauris a ultricies luctus, felis enim molestie ligula, ultricies sagittis magna sapien quis magna. Curabitur velit enim, porttitor sed odio vel, vehicula hendrerit purus. Morbi ultricies ipsum quis nisi porttitor, in sollicitudin tortor commodo. Nulla facilisi. Donec sed laoreet lacus. Integer ornare semper metus tristique aliquet. Nunc tristique mi ex, et rutrum mauris lacinia ac.

```
%% Loading Data and making objects.

A_filename_umsquared = [12710,10056,8320,7094,6184,5480,4920,4464,4086,3766,3493,3257,3051,2869
A_ppc = A_filename_umsquared.*((1E-6)^2);

%5Target Toy model data validation
F0_5Target = [5.86083 5.5316 5.33138 4.72955 4.01304 3.10839].*10^9;
PPC_To_be_squared = [47.0213 50 52.0384 59.1016 70.1727 91.214].*10^(-6);
A_ppc_5Target = PPC_To_be_squared.*PPC_To_be_squared;
% Jochem Code data
F0_Jochem = [2310621857.66901,2594151197.17164,2848091619.57189,3080193995.21574,3294570175.759
A_ppc;



iterator = 1:length(A_ppc);
%Constructs the PPC theory  and PPC sonnet objects.
PPCt = repmat(PPC_theory(),1,length(iterator));% convoluded way of preallocating an array of th
PPCs = repmat(PPC_sonnet_oneport(),1,length(iterator));% convoluded way of preallocating an arr
for i=iterator
    %Constructor arg: PPC_theory(W (Width PPC(m)),H (Lenght PPC(m)),d (thickness dielectric(m))
    PPCt(i) = PPC_theory(sqrt(A_ppc(i)),sqrt(A_ppc(i)),250E-9,10);
    total_filename = filename_begin+string(A_filename_umsquared(i))+filename_end;
    % PPC_sonnet_oneport(Width(m),N Length(m),d thickness(m),data_dir)
    PPCs(i) = PPC_sonnet_oneport(sqrt(A_ppc(i)),sqrt(A_ppc(i)),250E-9,total_filename,9);
end
% Constructs the CPW objects.
CPWtSC = CPW_theory(length_M,Z0,epsilon_eff);
CPWtPEC = CPW_theory(length_M,Z0_PEC,epsilon_eff_PEC);
CPWsSC = CPW_sonnet_oneport(2E-6,2E-6,length_M,filename_CPW_SC,9);
CPWsPEC = CPW_sonnet_oneport(2E-6,2E-6,length_M,filename_CPW_PEC,9);
```

Analysis: Finding intersectes Im[Z_{cpw}] = -Im[Z_{PPC}]

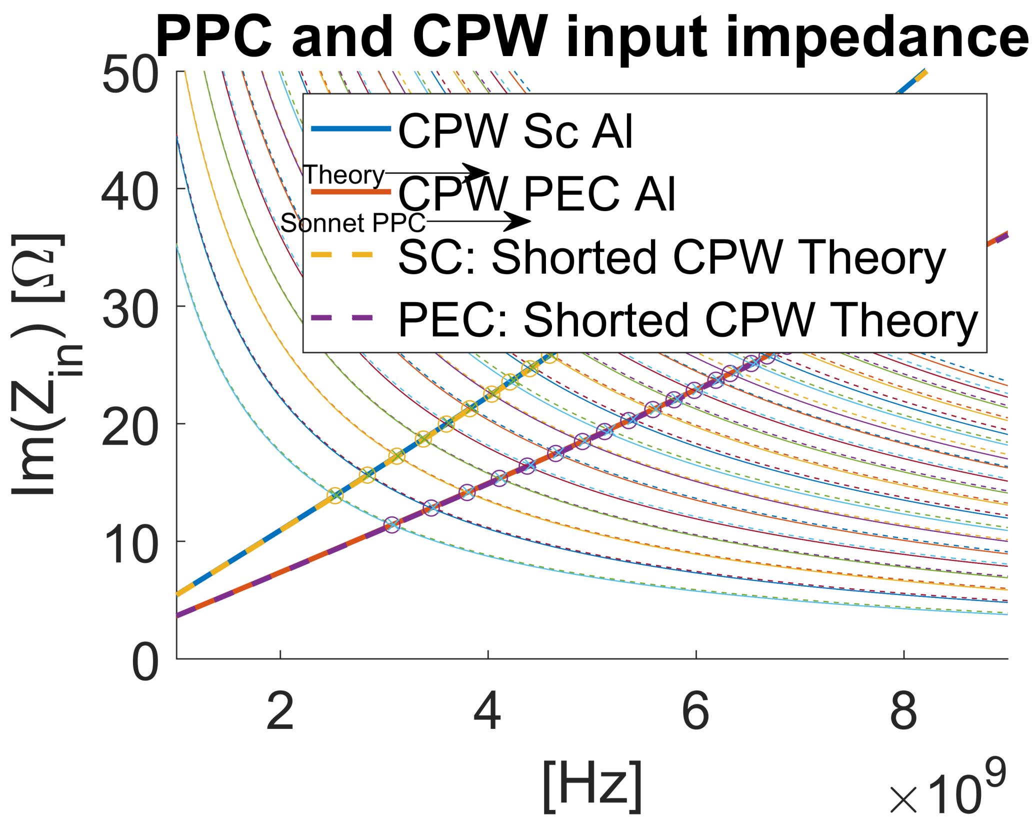
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sed mi mauris. Duis scelerisque, mauris a ultricies luctus, felis enim molestie ligula, ultricies sagittis magna sapien quis magna. Curabitur velit enim, porttitor sed odio vel, vehicula hendrerit purus. Morbi ultricies ipsum quis nisi porttitor, in sollicitudin tortor commodo. Nulla facilisi. Donec sed laoreet lacus. Integer ornare semper metus tristique aliquet. Nunc tristique mi ex, et rutrum mauris lacinia ac.

```
%% Analyse data. Finding the intersects between CPW and PPC curves.
%Disp graph for the input impedance
freq = PPCs.get_freq();
f1 = figure;
hold on
```

```
hcpwsc = sweetplot(CPWsSC.get_freq(),imag(CPWsSC.get_Zin));
hcpwpec = sweetplot(CPWsPEC.get_freq(),imag(CPWsPEC.get_Zin));
hcpwt = plot(freq,imag(CPWtSC.get_Zin(freq)),'--','linewidth',2);
hcpwt_PEC = plot(freq,imag(CPWtPEC.get_Zin(freq)),'--','linewidth',2);
plot_iterator = iterator;
for i=plot_iterator
    ht(i) = plot(freq,-imag(PPCt(i).get_Zin(freq)),'--');
    hts(i) = plot(freq,-imag(PPCs(i).get_Zin()));
    [x_intersect_sc(i),y_intersect_sc(i)] = find_intersect_2lines(PPCs(i).get_freq(),CPWsSC.get
    [x_intersect_pec(i),y_intersect_pec(i)] = find_intersect_2lines(PPCs(i).get_freq(),CPWsPEC.
    [x_intersect_sc_t(i),y_intersect_sc_t(i)] = find_intersect_2lines(freq,freq,-imag(PPCt(i).g
    [x_intersect_pec_t(i),y_intersect_pec_t(i)] = find_intersect_2lines(freq,freq,-imag(PPCt(i
    %normalizedtextarrow(gca(),[8.9E9 -imag(PPCs(i).get_Zin(4000))],[(freq(4000)) -imag(PPCs(i)
    set(gca,'FontSize',20);
end
hisc = plot(x_intersect_sc(plot_iterator),y_intersect_sc(plot_iterator),'o');
hipec = plot(x_intersect_pec(plot_iterator),y_intersect_pec(plot_iterator),'o');
hitsc = plot(x_intersect_sc_t(plot_iterator),y_intersect_sc_t(plot_iterator),'x');
hitpec = plot(x_intersect_pec_t(plot_iterator),y_intersect_pec_t(plot_iterator),'x');



xlabel('[Hz]')
ylabel('Im(Z_{in}) [\Omega]')
title('PPC and CPW input impedance')
xlim([1E9 9E9]);
ylim([0 50]);
normalizedtextarrow(gca(),[(freq(3000)-1E9) -imag(PPCt(15).get_Zin(freq(3000)))],[(freq(3000))
normalizedtextarrow(gca(),[(freq(3400)-1E9) -imag(PPCs(15).get_Zin(3400))],[(freq(3400)) -imag(
legend([hcpwsc hcpwpec hcpwt hcpwt_PEC],{'CPW Sc Al','CPW PEC Al','SC: Shorted CPW Theory','PEC
hold off
```



Calculation of the deltaF_0

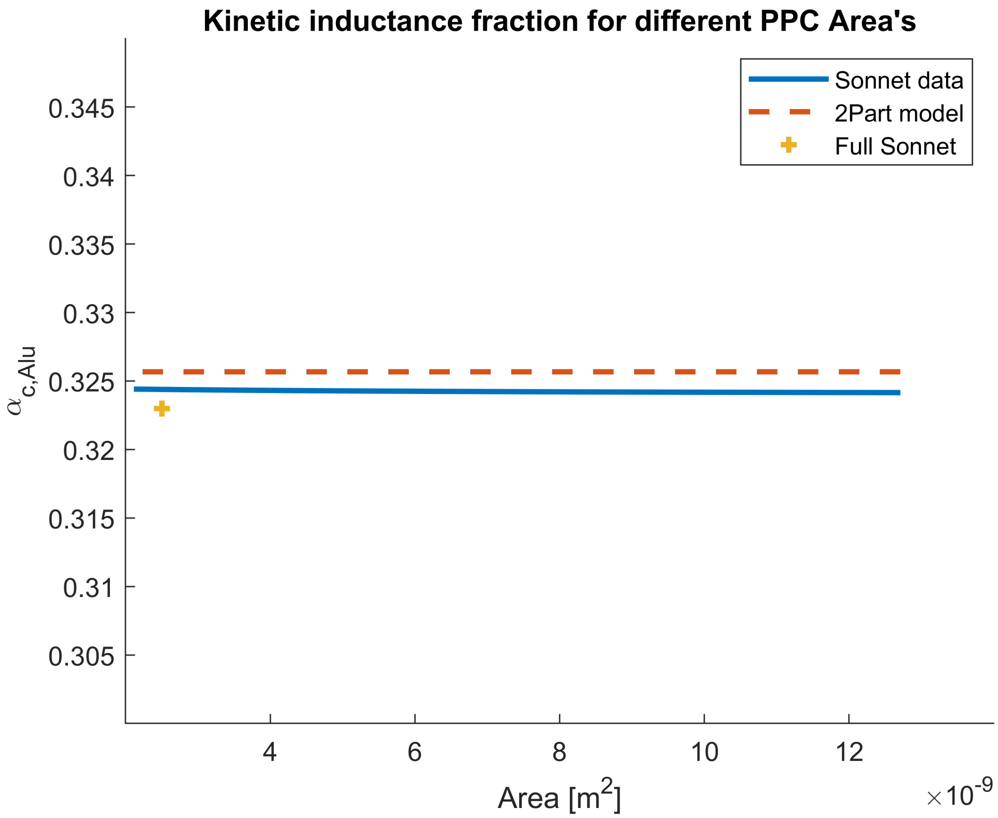
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sed mi mauris. Duis scelerisque, mauris a ultricies luctus, felis enim molestie ligula, ultricies sagittis magna sapien quis magna. Curabitur velit enim, porttitor sed odio vel, vehicula hendrerit purus. Morbi ultricies ipsum quis nisi porttitor, in sollicitudin tortor commodo. Nulla facilisi. Donec sed laoreet lacus. Integer ornare semper metus tristique aliquet. Nunc tristique mi ex, et rutrum mauris lacinia ac.

```
%% Calculation of the deltaF_0
iterator_partial = iterator;
alpha_c_alu =  kindfrac(x_intersect_sc,x_intersect_pec);
alpha_c_alu_t =  kindfrac(x_intersect_sc_t,x_intersect_pec_t);
%Now we add 1 point of data for the Toy_Model Origin2500 and Origin2500PEC
% In order to get 1 data Point.
alpha_c_alu_Origin2500 = kindfrac(5.5316,6.72285);



f2 = figure;
hold on
sizes_W = arrayfun( @(x) x.get_W, PPCs  );
sizes_H = arrayfun( @(x) x.get_H, PPCs );
sizes_A = sizes_W(iterator_partial).*sizes_H(iterator_partial);
sizes_W_t = arrayfun( @(x) x.get_W, PPCt  );
sizes_H_t = arrayfun( @(x) x.get_H, PPCt );
sizes_A_t = sizes_W_t(iterator_partial).*sizes_H_t(iterator_partial);
```

```
plot(sizes_A,alpha_c_alu(iterator_partial),'LineWidth',2);
plot(sizes_A_t,alpha_c_alu_t(iterator_partial),'--','LineWidth',2);
plot(2500E-12,alpha_c_alu_Origin2500,'+','LineWidth',2);
title("Kinetic inductance fraction for different PPC Area's")
ylim([0.3 0.35])
legend('Sonnet data','2Part model','Full Sonnet')
xlabel('Area [m^{2}]')
ylabel('\alpha_{c,Alu}')
hold off
```



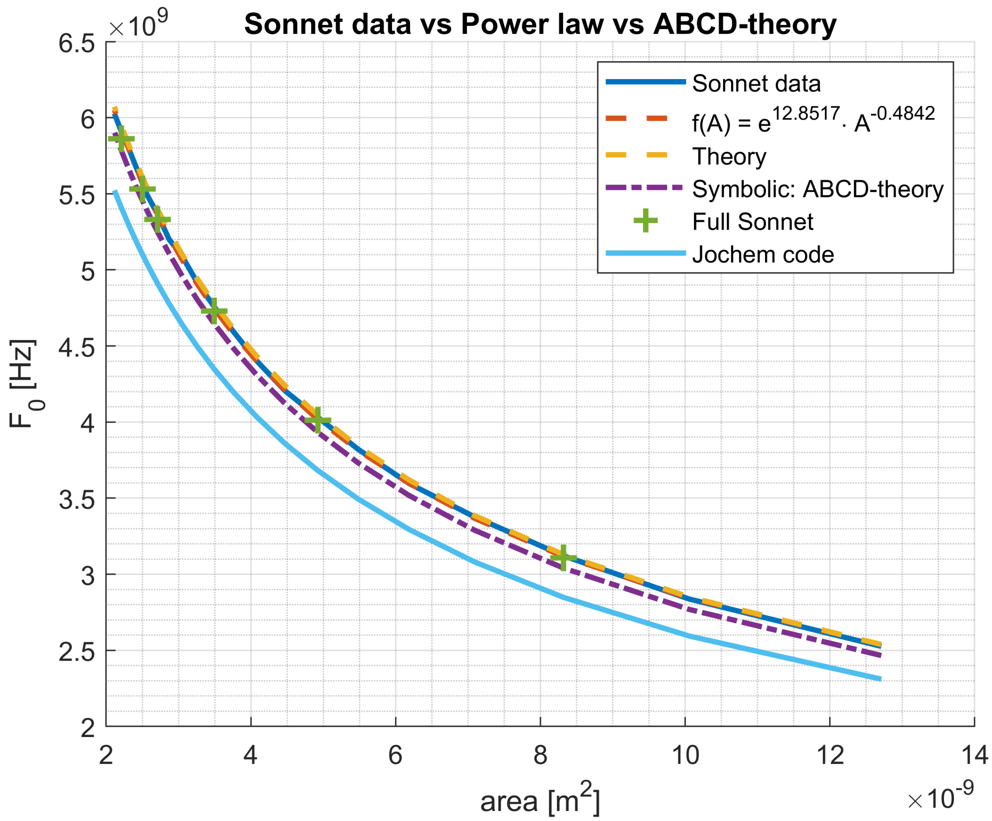

```
%ylim([0 0.2])
```

Fitting in Log-log space

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sed mi mauris. Duis scelerisque, mauris a ultricies luctus, felis enim molestie ligula, ultricies sagittis magna sapien quis magna. Curabitur velit enim, porttitor sed odio vel, vehicula hendrerit purus. Morbi ultricies ipsum quis nisi porttitor, in sollicitudin tortor commodo. Nulla facilisi. Donec sed laoreet lacus. Integer ornare semper metus tristique aliquet. Nunc tristique mi ex, et rutrum mauris lacinia ac.

```
%% fitting F0=C*A^k by fitting in log log space.
sizes_A = sizes_W(plot_iterator).*sizes_H(plot_iterator);
[f5,hpower, a , b , str_formula] = powerlaw_fit(sizes_A,x_intersect_sc(plot_iterator));
figure(f5)
conv_iter = 1:10;% this makes sure the theory is run 10 times to make it converge to the actual
f_start =1E9;
f = f_start;
```

```
%F0_out =  zeros(length(iterator));
F0 = zeros(length(conv_iter));
for j=iterator
    for i=conv_iter
    F0(i) = F0_theory(f,0.001,A_ppc(j),Z0,epsilon_eff,d);%F0 = F0_theory(freq[Hz],lenght_CPW[m
    err = (F0(i)-f)./F0(i);
    f = (F0(i)+f)/2;% in this piece of the code  we make a new guess using the formula averaged
    end
F0_out(j) = F0(length(conv_iter));
end
hold on
plot(sizes_A,x_intersect_sc_t,'--','linewidth',2);
plot(A_ppc,F0_out,'-.','linewidth',2)
h5Target = plot(A_ppc_5Target,F0_5Target,'+');
h5Target.MarkerSize = 10;
h5Target.LineWidth = 2;
hJochem = plot(A_ppc,F0_Jochem);
hJochem.MarkerSize = 10;
hJochem.LineWidth = 2;
legend('Sonnet data',str_formula,'Theory','Symbolic: ABCD-theory','Full Sonnet','Jochem code');
title('Sonnet data vs Power law vs ABCD-theory');
grid on
grid minor
hold off
```



Relation between Qc and A_coupler

In here i am now attempting to find the relation between the coupling strength expressed in the Qc of the resonator. And the overlap area beween the coupler and the readout line.

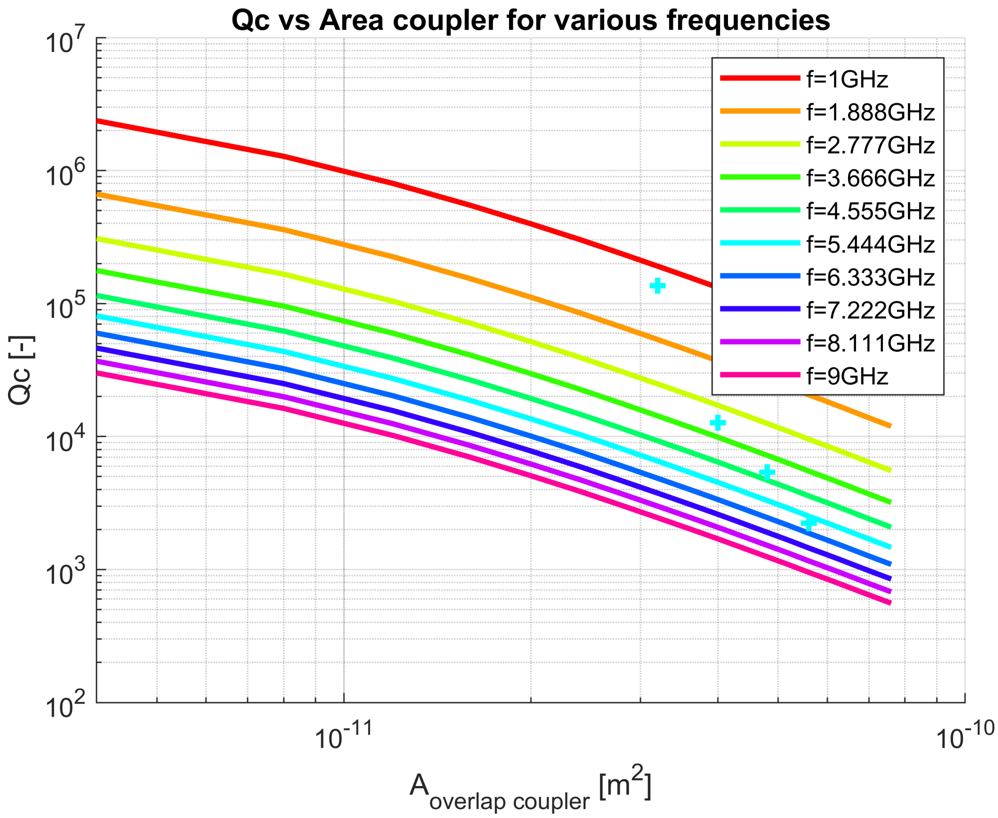
I already suspect that a large ovelap corresponds to a Low Qc. and a large overlap corresponds to a High Qc. But the relationship exact at this moment remains unknown.

Work in progress!

```
%% Relation between Qc and A_coupler
%g(A_coupler) =  Q_coupler
% UNIT_Coupler_sonnet_class();
%Making objects for the Coupler data
filename_coupler = 'Coupler_FullDielectricV0_5_0Oeff';
iterator_coupler = 1:20;
%CAUTION: i have added the x=2.554 correction factor that has been
%calcuated from the discussion on mail with Alejandro Akira.
for i=iterator_coupler
    Couplers(i) = Coupler_sonnet(4E-6,(i-1)*10^(-6),250E-9,2.55,filename_coupler+string(i)+file
end
% Number of header lines 15
%Plotting the coupler data for 1 freq.
figure
ax = axes('XScale', 'log', 'YScale', 'log');
Coupler_Freq_steps = 10;
ax.ColorOrder = hsv(Coupler_Freq_steps);
hold on
%Data from lorenzian fit
Lor_Area = [3.2e-11 4e-11 4.8e-11 5.5999e-11];
Lor_Qc = [136370.7 12717.8 5386.6 2229.08];



for i=1:Coupler_Freq_steps
freq_index = floor(subsref(linspace(1,8001,Coupler_Freq_steps),struct('type','()','subs',{{i}}
Area_Couplers = arrayfun( @(x) x.get_Area, Couplers  );
Qc_Couplers = arrayfun( @(x) x.get_Qc(freq_index), Couplers );
plot(Area_Couplers,Qc_Couplers,'LineWidth',2);
LegendData{i}='f='+string(Couplers(1).get_freq(freq_index)/10^9)+'GHz' ;
end
plot(Lor_Area, Lor_Qc,'+','Linewidth',2,'Color','c')
hold off
grid on
xlabel('A_{overlap coupler} [m^{2}]')
ylabel('Qc [-]')
legend(LegendData)
title('Qc vs Area coupler for various frequencies')
```



Make parameters for 16 CKIDs

Over here I calculate the area for 16 equally spaced CKIDs. Already talked to Jochem and probably we want to make 3 groups of 4 KIDs that are in quite a small piece of spectrum.

```
%% Make parameters for 16 CKIDs

% Linspace between 4-8 GHz of 16 CKIDs
addpath('.\Exportdata')
F0_16CKID = linspace(4,8,16);
%Use struct to refactor code over here.
Area_16CKID = f_inv_fit(a,b,F0_16CKID,'native')';

CKID_DataPackage= [F0_16CKID' Area_16CKID ];
writematrix(CKID_DataPackage,'Exportdata/Datapackage.csv','Delimiter','tab')
```

Get responsivity

```
%% Calc. parameters responsivity. Get responsivity.
% In here i want to calculate the parameters to calculate the responsivity.

%Parameters necessary for getNqp_Tbath function.
Alu.Teff = 0.120; % [K]
Alu.Tc          = 1.125;    % [K] (Source: Jochem code)
```

```
Alu.eta_pb      = 0.4;      % (Source: Jochem code)pair breaking eficiency 0.4 for medium thin
Alu.tau0        = 438e-9;   % (Source: Jochem code)Kaplan single particle tau_0. 438e-9 is the
Alu.length = 0.001; %[m]
Alu.width = 2E-6; %[m]
Alu.height = 30E-9; %[m]
Alu.V = Alu.length*Alu.width*Alu.height ; %[m^3]
KID.Tbath = 0.120 ; % [K]
```

Mazin naive formula

i found a formula for 6 GHz resonator in his dissertation. So my responsivity has to be in the ballpark range of this formula.

$$\frac{d\theta}{dN_{qp}} = 1.63 \cdot 10^{-7} \frac{\alpha_k Q}{V}$$

(Source:)

```
% Start-----Compare Mazin


% End ------Compare Mazin
```

```
sigma_n_Al = 3.77E7; % [Siemens/m] Source: https://en.wikipedia.org/wiki/Electrical_resistivity
beta = calcbeta(5.44E9,epsilon_eff,phyconst.c);
[Alu.nqp, Alu.tauqp, Alu.Nqp, KID.NEPGR] = getNqp_Tbath(Alu.Tc, Alu.eta_pb, Alu.V, KID.Tbath, A
```

```
Unrecognized function or variable 'getNqp_Tbath'.
```

```
[Alu.sigma1,Alu.sigma2,Alu.ds1dn,Alu.ds2dn]                        = Sigmas(2*pi*KID.Fres,
[KID.dthetadN,KID.dAdN,KID.dxdN,Alu.abssigma,KID.Qi,KID.Q,Alu.Beta]     = getresponsivity2(Alu.
                                                                      %[dthetadN,dRdN,dxdN,
sigma = 0;%To be filled in....
```

Appendix

Additional programs that can come in handy(You don't need to run these.)

Optional: Symbolic Matrix Multiplier(SMM)

Warning: it is quite slow!! can take like a few minutes to run.

This is an additional program that can calculate the ABCD matrices for the CKID. There is however an error in there because the obtained S21 parameters don't start at 0dB which makes no sense.

Do you want to run this section? *Check the checkbox*

```
RunSMM = false
```

```
RunSMM = logical
   0
```

```
if RunSMM

syms theta
syms Z
syms Z0
syms freq
syms C_PPC
syms C_c

COUPLER = [1 -1j*(1/(2*pi*freq*C_c));0 1];
PPC = [1   0 ; 1j*2*pi*freq*C_PPC 1];
CPW = [cos(theta) 1j*Z0*sin(theta) ; 1j*(1/Z0)*sin(theta) cos(theta) ];
SHORT = [1 0;1/Z  1 ];

SHORTED_CPW = CPW*SHORT;
ZIN_SHORTED_CPW = SHORTED_CPW(1,1)/SHORTED_CPW(2,1);
LIM_SHORTED_CPW = limit(ZIN_SHORTED_CPW,Z,0);
RES = PPC*CPW*SHORT ;
TOT = COUPLER*PPC*CPW*SHORT;
ZIN_RES = RES(1,1)/RES(2,1);
LIM_RES = limit(ZIN_RES,Z,0);
ZIN_TOT = TOT(1,1)/TOT(2,1);
LIM_TOT = limit(ZIN_TOT,Z,0);
disp(LIM_TOT);

%Parameters:
Z0 = 76.28;
C_PPC = 0.918E-12;%C_PPC = 2E-12;%C_PPC = 0.918E-12;% Capacitance value for the PPC.
C_c = 5.66E-15; %Capacitance value for the coupler.
l = 0.001;
c = 2.98E8;
epsilon_eff = 10.6009;
theta = 2*pi*((sqrt(epsilon_eff)*freq)/c)*l;

subput_RES = subs(LIM_RES);
subput_TOT = subs(LIM_TOT);
freq = linspace(5.5E9,5.6E9,10000);
Output_RES = subs(subput_RES);
Output_TOT = subs(subput_TOT);
figure
hold on

plot(freq,imag(double(Output_RES)));
plot(freq,imag(double(Output_TOT)));
%xlim([5E9 6E9])
ylim([-10000 10000])
title('Input impedance calculated with ABCD formalism');
legend('PPC + CPW + Short','coupler + PPC + CPW + Short')
xlabel('freq [Hz]')
```

```
ylabel('Im(Z_{11}) [\Omega])')
hold off


%% Including the beginning and the end of the readout line

syms Z_s
syms Z0_rl
syms Z_ref
syms theta_rl % for now we assume symmetry.
ZKID = [1 0;1/Z_s 1];%Compact kid modeled as shunt impedance
TL = [cos(theta_rl) 1j*Z0_rl*sin(theta_rl) ; 1j*(1/Z0_rl)*sin(theta_rl) cos(theta_rl) ];
SYSTEM = TL*ZKID*TL;
%Now we want to put in the info we obtained about Z_in_kid
double_output_tot = double(Output_TOT);
Z0_rl = 50;
Z_ref = 50;
theta_rl = 2*pi*((sqrt(epsilon_eff)*freq)/c)*l;
%clear freq
%syms freq
theta = 2*pi*((sqrt(epsilon_eff)*freq)/c)*l;
Z_s = LIM_TOT;
%freq = linspace(1E9,9E9,800);

SOUT = subs(SYSTEM);
S21_sym = 2/(SOUT(1,1) +(SOUT(1,2)/Z_ref)+ SOUT(2,1)*Z_ref + SOUT(2,2));% I want this to be a f
%% Plotting
figure
subplot(2,1,1)
plot(freq,20*log(abs(double(subs(S21_sym)))))
xlabel('freq [Hz]')
ylabel('|S_{21}| [dB]')
title('Analytic calculation of S_{21}');
legend('Mag','phase');
subplot(2,1,2)
plot(freq,angle(double(subs(S21_sym))),'--');
xlabel('freq [Hz]')
ylabel('arg(S_{21}) [rad]')
else
disp('RunSMM = false So not running this part')
end
```

```
RunSMM = false So not running this part
```