CERN-ACC-NOTE-2023-0021

Analytic Computation of a Ferro-Electric Fast Reactive Tuner

Ilan Ben-Zvi*[1], Alick Macpherson[2], Alex Castilla[3] and Nicholas Shipman[2]

1) Physics and Astronomy Department, Stony Brook University, NY USA
2) CERN, CH-1211 Geneva, Switzerland
3) Jefferson Laboratory, Newport New VA USA

*Ilan.Ben-Zvi@StonyBrook.edu

ABSTRACT

An analytic solution using the Maple® application [1] of a Ferroelectric Fast Reactive Tuner (FE-FRT) is presented. The equations used are reproduced for convenience of the reader from a detailed analytical model of the tuner [2,3]. The program covers several configurations, allowing control of the frequency of superconducting and normal-conducting cavities in a variety of applications and frequencies. A Maple-text input file is provided to allow the interested reader to run the application.

I. INTRODUCTION

The design of a high-power Ferroelectric Fast Reactive Tuner (FE-FRT) is complex, time-consuming process. Using the Maple® based program speeds up of the process of designing the circuit and allows one to develop an intuitive understanding of the tuner. The solutions provided by this code are approximate but allow one to save time running mesh solvers and avoid local minima. The design concepts, the theoretical basis and its confirmation through numerical simulations have been presented [2,3] and will not be repeated here.

II THE TUNER CONIGURATIONS

II-1 THE EQUIVALENT CIRCUIT DIAGRAM

The ferroelectric tuner may be configured in several ways, depending on the application. A few options are shown in Figure 1. All versions have common features, such as a coupler port of the cavity to be tuned, connected through a certain length of a transmission line to the tuner elements. The basic element is a string of capacitors connected in series (four capacitors shown in Figure 1 as an arbitrary example). The permittivity of the capacitors is modulated by a bias voltage, with bias connections introduced between the series capacitors. This series connection of capacitors, which we introduced in [2], offers several advantages over a single capacitor configuration, in that it has a larger power handling capacity (due to the larger surface area) and a higher reactance (due to the series connection) to match the impedance of the coupler port. It also allows for a lower voltage bias power supply since bias-wise the wafers are in parallel.

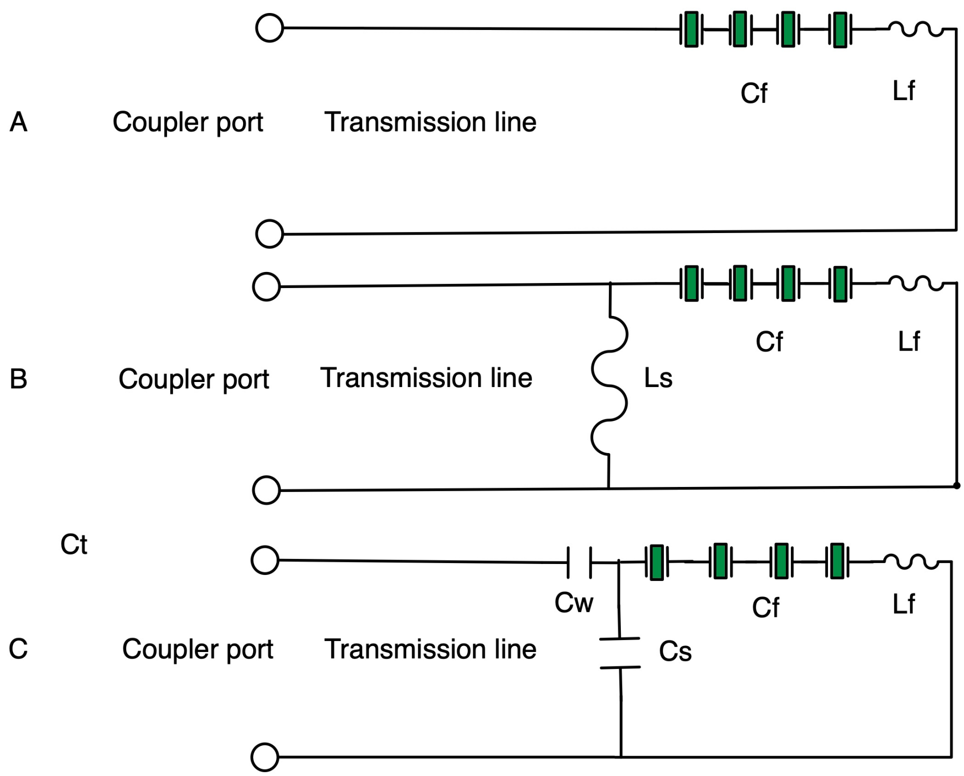

Figure 1, the circuit diagram for three potential configurations of a tuner using ferroelectric loaded capacitors. The number of ferroelectric capacitors connected in series is N_w. Configuration (A) is a proportional, resonant, or non-resonant circuit. Configuration (B) employs a resonant circuit for increased reactance. Configuration (C) shows a resonant circuit for high-frequency applications and in addition a capacitive coupling to the transmission line through a window.

Figure 1A depicts the simplest configuration, the string of capacitors connected directly to the cavity through a transmission line. The capacitor stack includes a self-inductance L_f which must be considered at high frequency. This configuration can be operated simply as a proportional tuner, for example in acoustic noise reduction in superconducting linac cavities.

Figure 1B, which we previously introduced [2], is suitable for either a proportional tuner or two states switching. Here the ferroelectric capacitor stack forms a resonator with the inductance L_s, and one arranges that the two states of the ferroelectric capacitor (no bias of full bias) produce reactances that are equal in magnitude but opposite in sign at the coupler port. In this configuration the resonant frequency of the tuner in the full bias is above the frequency of the tuned cavity (which we will denote as State 1), and the zero-bias mode of the tuner circuit lies below the frequency of the cavity. Note that as described in [3], the tuner can be reconfigured from a two-state to a proportional tuner by adding or subtracting a quarter wavelength to the transmission line. In a Smith Chart representation, the tuner then operates either around the open-end (two-state) or the shorted-end (proportional tuning) points of the Smith Chart.

In Figure 1C we introduce two additional capacitors (non-ferroelectric) to enhance the performance of the tuner.

C_s is useful at high frequencies, when the reactance of the inductor L_f exceeds the reactance of the capacitor C_f at the operating frequency of the cavity. C_w is introduced in series with the transmission line. It serves two functions: It can be used to control the reactance of the tuner to the desired impedance of the transmission line, and it serves as a window coupler, isolating the cavity vacuum from the tuner vacuum. This last feature is very valuable for protecting the vacuum of superconducting cavities.

II-2 THE IMPEDANCE OF THE TUNER

A simple proportional tuner can be constructed without the inductive element L_s, as in Figure 1A. The detailed model of this configuration can be derived as a simplification of the model which we describe in this publication. The PS tuner configuration described in [5] uses the two-state circuit, depicted in Figure 1B, comprising the capacitor stack and a "stub", inductive short of inductance L_s. As we demonstrated, such a tuner is capable of high-power reactive tuning.

The analytic model of either configuration may be easily implemented in a mathematical software package or programmed in general software such a Python. We use Maple® [1]. In this section we reproduce some of the material presented in [2,3] to derive the analytic expression of the impedance of the tuner as measured at the cavity port.

The tuner's impedance Z presented to the coupler port of the cavity is that of a transmission line with characteristic impedance Z_0, of length l and terminated by a load impedance Z_L is given by

1. $Z \equiv R + jX = Z_0 \frac{Z_L + Z_0 \tanh(\gamma l)}{Z_0 + Z_L \tanh(\gamma l)}$

Where γ is the complex propagation coefficient given by:

2. $\gamma = \alpha + j\beta$

3. $\beta = {}^{\omega}/_{c}$

For a coaxial transmission line of inner radius a, outer radius b and surface resistivity R_s:

4. $\alpha = \left(\frac{1}{a} + \frac{1}{b}\right) \frac{R_s}{2\eta \cdot \log({}^{b}/_{a})}$

Where $\eta = \sqrt{\mu_0/\epsilon_0} = 376.7\Omega$, the impedance of vacuum.

Note that the characteristic impedance Z_0 of a lossy line is a complex number

5. $Z_0 = \frac{\eta}{2\pi} \log\left(\frac{b}{a}\left(1 - j\frac{\alpha}{\beta}\right)\right)$

The load impedance of the tuner just before the transmission line, Z_L, is described by series and parallel circuits comprising the elements shown in Figure 1. We approximate the impedance of the inductors by transmission line sections to include resistive losses. The losses in the ferroelectric material are presented by the loss tangent $\tan(\delta)$, which we will approximate as δ. We neglect losses in the conventional capacitors.
Depending on the circuit configuration, we get the tuner's impedance at the coupling port Z:

6. (A) $$Z = Z_0 \frac{\frac{1}{j\omega C_f(1-j\delta)} + j\omega L_f + Z_0 \tanh(\gamma l)}{Z_0 + \left(\frac{1}{j\omega C_f(1-j\delta)} + j\omega L_f\right)\tanh(\gamma l)}$$

(B) $$Z = Z_0 \frac{\frac{j\omega L_S\left(\frac{1}{j\omega C_f(1-j\delta)} + j\omega L_f\right)}{j\omega L_S + \frac{1}{j\omega C_f(1-j\delta)} + j\omega L_f} + Z_0 \tanh(\gamma l)}{Z_0 + \frac{j\omega L_S\left(\frac{1}{j\omega C_f(1-j\delta)} + j\omega L_f\right)}{j\omega L_S + \frac{1}{j\omega C_f(1-j\delta)} + j\omega L_f}\tanh(\gamma l)}$$

(C) $$Z = Z_0 \frac{\frac{\frac{1}{j\omega C_S}\left(\frac{1}{j\omega C_f(1-j\delta)} + j\omega L_f\right)}{\frac{1}{j\omega C_S} + \frac{1}{j\omega C_f(1-j\delta)} + j\omega L_f} + \frac{1}{j\omega C_W} + Z_0 \tanh(\gamma l)}{Z_0 + \left(\frac{\frac{1}{j\omega C_S}\left(\frac{1}{j\omega C_f(1-j\delta)} + j\omega L_f\right)}{\frac{1}{j\omega C_S} + \frac{1}{j\omega C_f(1-j\delta)} + j\omega L_f} + \frac{1}{j\omega C_W}\right)\tanh(\gamma l)}$$

The Maple program applies equation (6) to get numerical results for the real and imaginary parts of Z. For the two states, where C_f take values of C_{f1} and C_{f2}, the impedance at the cavity's coupler port takes the values Z_1 and Z_2, respectively. A third state is also used in which Z is evaluated at a representative "mid-permittivity" of the ferroelectric is also used.

The impedance Z at the cavity's port can be adjusted by varying the circuit components such as the window capacity C_w, the length of the transmission line etc. The program then solves non-linear equations such as $\Im(Z_1) = -\Im(Z_2)$ etc.

The Figure of Merit, FoM, can be defined as

7. $$FoM = \frac{X_2 - X_1}{2\sqrt{R_1 R_2}}$$

Given a solution to these equations, we can evaluate the Figure of Merit using the real and imaginary components of Z_1 and Z_2 as well as other characteristics of the system.

II-3 COUPLING THE TUNER TO THE CAVITY

The impedance of the tuner at the port of the cavity must be translated into other variables of the problem, such as the tuning range. For that we have additional equations.

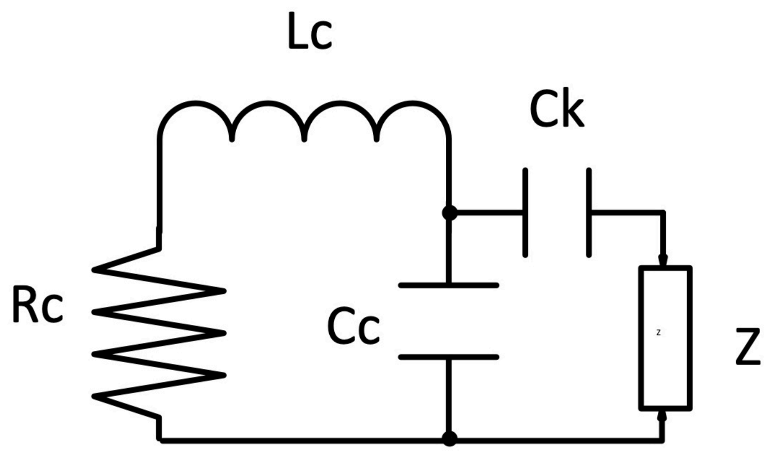

Figure 2. Equivalent circuit diagram of the cavity, showing the tuner's impedance Z coupled to the cavity circuit, represented by R_c-L_c-C_c through the coupling capacitor C_k.

As described in [2,3], the cavity is represented by the equivalent circuit diagram shown in Figure 2, comprising the inductance L_c, capacitance C_c and resistance R_c. The cavity is coupled to the external load Z through a port, represented here by the coupling capacitor C_k. The fundamental power coupling port is not shown. The natural (no tuner) resonant frequency of the cavity is ω_0, and the frequency of the tuned cavity (when the reactance Z is included) is ω.
Let us define a few additional functions. The external quality factor of the cavity's port is given by the cavity's shunt impedance (accelerator convention) ${R_{sh}}/{Q} = 1/\omega C_c$, the characteristic impedance Z_0 of the transmission line connected to the port and the coupling's transformer turn ratio k.

8. $Q_e = \frac{{R_{sh}}/{Q}}{Z_0 k^2}\,,\ Q \equiv \frac{\omega L_c}{R_c},\ \ \omega_0 \equiv \frac{1}{\sqrt{L_c C_c}},\ \ \Delta\omega \equiv \omega - \omega_0,\ \ \zeta \equiv \frac{\Delta\omega}{\omega},\ \ k \equiv \frac{C_k}{C_c}$

We assume that k is small, $k \ll 1$, but also that $\frac{1}{Qk} \ll 1$. The resistive part R of the tuner's impedance Z is transformed into the cavity's circuit with the square of the coupling strength:

9. $\Re(Z_c) \approx Rk^2$

The transformer ratio k of the port is related to Q_e

10. $Q_e \equiv \frac{\omega L_c}{\Re(Z_c)} = \frac{R_{sh}/Q}{Z_0 k^2}$

The reactive tuning of the cavity by the tuner circuit, neglecting dissipation by the real part of Z, is given by:

11. $\left(\frac{\omega}{\omega_0}\right)^2 \approx 1 - 2\zeta = \frac{X\omega C_c - k^{-1}}{X\omega C_c - 1 - k^{-1}}$

Leading to

12. $\zeta = \frac{-1}{2(X\omega C_c - 1 - k^{-1})}$

and

13. $\Delta\omega_{12} \equiv \Delta\omega_2 - \Delta\omega_1 = \frac{\omega}{2}\left[\frac{-1}{X_2\omega C_c - 1 - k^{-1}} - \frac{-1}{X_1\omega C_c - 1 - k^{-1}}\right]$

For a reasonably small coupling strength k, and given $\frac{X_{1,2}}{R_{sh}/Q} \ll k^{-1}$, we can neglect the reactance dependent terms in the denominator, we get a simple expression for the tuning of the cavity between the two end-states of the tuner:

14. $\Delta\omega_{12} = \frac{\omega}{2Q_e}\frac{(X_2 - X_1)}{Z_0}$

The connection between the frequency tune, stored energy, and reactive power is given by

15. $\Im(\Delta P_{12}) = 2U\Delta\omega_{12}$

When calculating the power dissipated in the ferroelectric material, we just set the surface resistivity of all conductors in the tuner to zero, then get $\Re(P_Z)$. Similarly, we can isolate the power dissipation in any component of the tuner.

IV. IMPLEMENTING THE ANALYTICAL MODELS IN MAPLE

The mechanical layout representing option C of Figure 1 is shown in Figure 3. Sapphire insulators, shown in blue, form the capacitors C_s and C_w. Note that by shorting C_w and replacing the capacitance C_s by an inductor L_s we obtain option B of Figure 1.

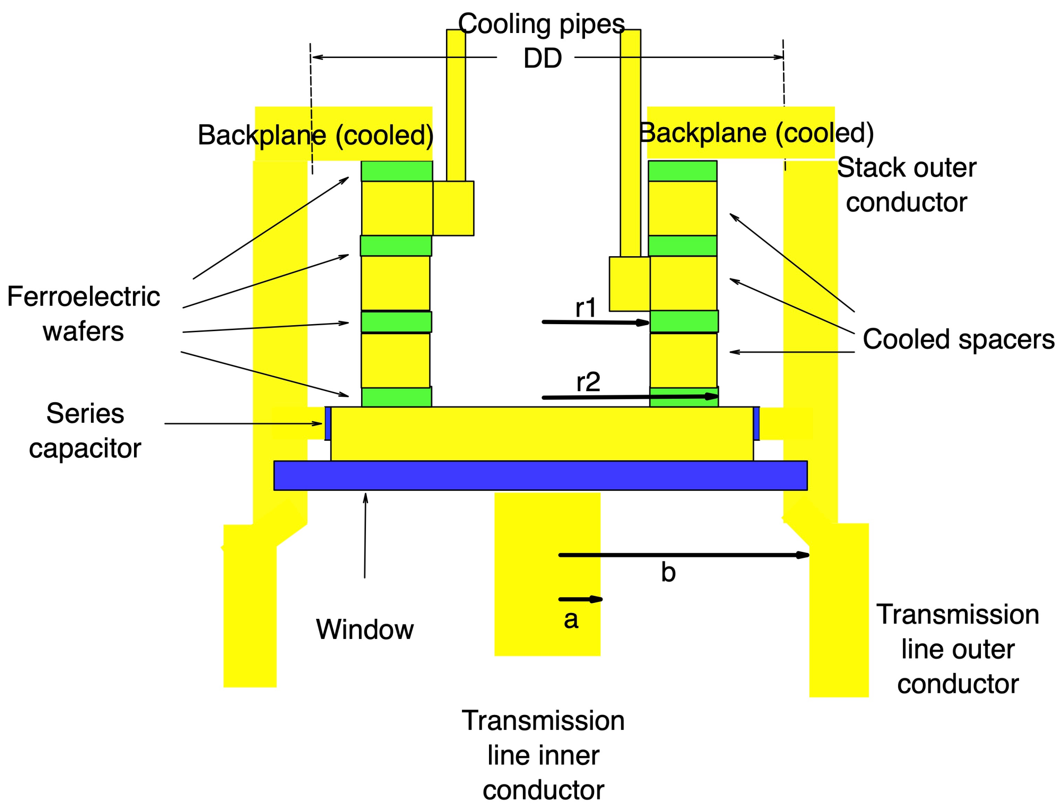

Figure 3. Schematic mechanical layout of the tuner representing configuration C of Figure 1, indicating most of the parameters used in the program. The ferroelectric wafers are shown in green, copper in yellow and sapphire in blue. The upper sapphire spacer is the dielectric of the series capacitor C_s and the lower one is for the window-capacitor C_w. A couple of cooling lines are show.

Determination of parameters

The configuration 1C (or 1D) is the most complex, so we will start with that. In this calculation we neglect lossy (real) terms.

Let $Z_w \equiv -\frac{1}{\omega C_w}$, $Z_t \equiv -\frac{1}{\omega C_t}$, $Z_s \equiv \omega L_s$, and Z_f represents the reactance of the ferroelectric capacitors stack, including the inductive elements of the spacers and transitions. The length of the spacers, $lspacer$, can be another variable of the system. Then we get the reactance of the tuner at the cavity terminal:

$$Z = Z_0 \frac{Z_w + Z_s - \frac{{Z_s}^2}{Z_f + Z_t + Z_s} + Z_0 \tan(\gamma L)}{Z_0 - \left[Z_w + Z_s - \frac{{Z_s}^2}{Z_f + Z_t + Z_s}\right] \tan(\gamma L)}$$

Let Z_{fi} , $i = 1..3$ be the values of Z_f at $\epsilon_i = \epsilon_1, \epsilon_2, \epsilon_3$, where $\epsilon_3 = \sqrt{\epsilon_1 \epsilon_2}$, then we can apply the desired values for Z by writing three equations with four parameters: Z_w, Z_s, $lspacer$, and L. Let us choose L as a free parameter, then we have three equations to solve for three unknowns. The best FoM is obtained when the end point reactances $Z(\epsilon_1)$ and $Z(\epsilon_2)$ have the same absolute value and opposite sign. Depending on the value of the Q_e of the tuner's port and the desired frequency tuning range, the absolute value of the end point reactances $Z(\epsilon_1)$ and $Z(\epsilon_2)$ is determined by Equation 13. For example, we can choose the following conditions:

$$\Im(Z(\epsilon_1)) = -Z_0$$
$$Z(\epsilon_3) = 0$$
$$\Im(Z(\epsilon_2)) = Z_0$$

The solution of these three equations produces a proportional tuner. There is a solution for various values of L, but not all solutions are acceptable and among the acceptable ones one can aim at an optimum depending on the engineering of the tuner system.

For Case 1B we have two variables, Z_s and L. Two variables are not sufficient to determine all three conditions, but we may use the length of the spacers in the ferroelectric capacitor stack to adjust the absolute size of the end reactances to the desired value, say Z_0 or some larger value if Q_e cannot provide the tuning range with the choice of $|Z(\epsilon_1)| = |Z(\epsilon_2)| = Z_0$.

V. SUMMARY

In summary, we presented an implementation of the analytic model of a fast reactive tuner using the Maple® application. The code is appended to this report and may be imported using Maple® Text. The code is annotated extensively to aid the user.

XII. BIBLIOGRAPHY

[1] Maple 2020.2, Maplesoft, a division of Waterloo Maple Inc., Waterloo, Ontario.
[2] Ilan Ben-Zvi, Alejandro Castilla, Alick Macpherson, and Nicholas Shipman, ArXiv:2109.06806v3, 2021

[3] Ilan Ben-Zvi, Graeme Burt, Alejandro Castilla, Alick Macpherson, and Nicholas Shipman, in preparation.

XIII. APPENDIX – MAPLE TEXT FILE

```
>
;
# FE-FRT tuning of a cavity
>
# This is a code developed in support on work in the references below. It is written in the Maple product language Maple 2020.2, Maplesoft, a division of Waterloo Maple Inc., Waterloo, Ontario.
>
# Fundamental constants:
> mu0 := 1.000000001*4*Pi*10^(-7);
> epsilon0 := 1.000000001/(36*Pi*10^9);
> eta := sqrt(mu0/epsilon0);
> cc := 1/sqrt(mu0*epsilon0);
# Calculations are carried out in accordance with four cases, corresponding to the circuit's schematic shown bellow [2]. Input parameters: These are set for four cases, A,B,C and D. The circuit diagram for cases A,B and C are shown in the figure below. Case D's diagram is identical to C, except for the configuration of the transmission line and window capacitor, where a short transmission line of length LA is placed ahead of the window capacitor.
# $$$$$$$$$$
> Case := D;
#
# TDD1 parameters
> if Case = D then
>    f := 4*10^8;
>    Nwafers := 4;
>    Ncool := 2*Nwafers;
>    Tw := 50;
>    dg := 0.0009;
>    r1 := 0.0165;
>    r2 := 0.0196;
>    DD := 0.085;
>    aline := 0.0196;
>    bline := 0.05;
>    L := 0.677;
>    LA := 0.01;
>    Ucav := 4.5;
>    RoverQ := 44;
>    Qe := 50000;
>    Xfactor := 1;
> end if;
```

$f := 400000000$

$Nwafers := 4$

Ncool := 2 Nwafers

Tw := 50

dg := 0.0009

r1 := 0.0165

r2 := 0.0196

DD := 0.085

aline := 0.0196

bline := 0.05

L := 0.677

LA := 0.01

Ucav := 4.5

RoverQ := 44

Qe := 50000

Xfactor := 1

;
Cf is the ferroelectric multi-wafer stack.
Lf is the parasitic inductance associated with the stack, mostly due to the spacers.
In Case B Ls is used to set the resonant frequency of the tuner circuit.
In Case C, the series capacitor Cs provides a tap in the resonant circuit to couple to the cavity to be tuned.
Cw is a window capacity. It is essentially a port connecting the tuner resonator to the outside world, in this case to the transmission line leading to the cavity which we tune. It is call "window capacitor" since its two electrodes are on opposing sides of the dielectric indow. The window provides vacuum sealing of the superconducting cavity. In Case C it is used to adjust the reactance of the tuner at the cavity's port to a value near the characteristic impedance of the trqnsmission line.
Problem parameters: Here we specify the cavity to be tuned (frequency, stored energy, R/Q, external Q of the tuner's coupling port and the frequency tuning which we target for the device)
f is the frequency of operation.

Œ¥ is the loss tangent of ferroelecric at the given frequency. It is calculated as a function of wafer temperature.
Nwafers is the number of wafers in the stack.
Tw is the wafer temperature (degrees C)
>
TherCon is the Thermal conductivity of the ferroelectric.
Ncool is he number of cooled surfaces per wafer.
dg is the single wafer thickness.
r1 is the inner radius of the ring-shaped wafer (the hole size), r2 is the outer radius, such that the area is as if there were no hole.
DD is the diameter of the conductor surrounding the stack of wafers and spacers.
Lspacer is the length of an inter-wafer electrode, which we be determined by the program in case C (also D)
aline is the radius of the inner conductor of the transmission line.
bline is the radius of the outer conductor of the transmission line.
Ucav is the stored energy in the cavity.
RoverQ is the R/Q of the cavity, in the accelerator convention R/Q=sqrt(L/C).
L is the length of the transmission line past the window capacitor to the cavity port.
In the proportional mode the reactance goes smoothly through zero in mide range (at about œµrep).
Xfactor is the ratio of the state reactance to the characteristic mpedance of the transmission line,
Material propertiers:
Used ferroelectric material properties for (Ba,Sr)TiO4+Mg oxides (BST(M)), as given by Euclid Techlab Inc.
œµ2 is the unbiased ferroelectric permittivity at the wafer's temperature Tw (degrees C)

```
> epsilon2 := 0.0274*Tw^2 - 3.3608*Tw + 229.14;
                  epsilon2 := 129.6000

;
```

Kdc is the bias tuning coefficient, assumed under a bias electric field of 8 volts per micron:

```
> Kdc := 1.29 + 0.11*(epsilon2 - 100)/60;
                   Kdc := 1.344266667

;
```

œµ1 is the permittivity under full bias

```
> epsilon1 := epsilon2/Kdc;
                epsilon1 := 96.40944255

;
> delta := (5*10^(-7)*Tw^2 - 5*10^(-5)*Tw + 0.0022)*(f/(4*10^8))^0.63;
                delta := 0.000950000000

;
```

```
> TherCon := 7.02;
# Conductivity of copper
> CondCu := 5.96*10^7;
                                        7
                 CondCu := 5.960000000 10

;
> Ebias := 8*10^6;
                      Ebias := 8000000

;
# "Representative" dielectric constant. We adjust the frequency of the tuner to about the
cavity' frequency when the ferromagnetic permittivity is at œµrep. This is a center point of the
permittivity range.
> `&epsilon;rep` := sqrt(epsilon1*epsilon2);
                &epsilon;rep := 111.7795319

;
#
# Calculations of some basic variables
# Number of cooling surfaces in each capacitor. A single ferroelectric wafer has cooling on two
sides through the cooled electrode.
# For the most aggressive tuner (high reactive power) we expect to cool each wafer through the
spacers on each side. For low reactive power this is not necessary. However, the temperature
rise in the wafers is calculated with this maximal cooling.
> Ncool := 2*Nwafers;
                         Ncool := 8

;
# Required bias voltage (Volts):
> Vbias := dg*Ebias;
                     Vbias := 7200.0000

;
# Wafer area
> S := Pi*(-r1^2 + r2^2);
                    S := 0.0003515756339

;
# Angular frequency of the cavity:
> omega := 2*Pi*f;
                   omega := 800000000 Pi

;
```

```
# The size of the capacitor in the equivalent circuit of the cavity:
> Ccav := 1.0000001/(RoverQ*omega);
                                    -12
                 Ccav := 9.042895396 10

;
#
# Preliminary calculations:
>
;
# Surface resistance of copper at operating frequency
> Rsurf := sqrt(Pi*f*mu0/CondCu);
                  Rsurf := 0.005147385970

;
# Rcon is an array specifying which conductor elements (assumed to be copper) will contribute to RF losses. This feature allows one to isolate the power loss in any element of the device.
# The switches are valued either 1 (for inclusion) or 0 (for exclusion) of loss at given element.
# Order of switches: [Rbackplane, Rspacers, RouterStack,RinnerTL,RouterTL]
>
# For the efinitions of the various elements, consult the figure above:
# Rbackplane - surface resistivity of the shorting plane terminating the stack to the stack's outer conductor
# Rspacers - surface resistivity of the spacers between the ferroelectric wafers
# RouterStack - surface resistivity of the stack outer envelope
# RinnerTL - surface resistivity of the transmission line's inner conductor
# RouterTL- surface resistivity of the transmission line's outer conductor
# We tune the device first just with dielectric losses, thus
> Rcon := array([0, 0, 0, 0, 0]);
RconAssign(Typesetting:-mtable(Typesetting:-mtr(Typesetting:-mtd(

  0, rowalign = "", columnalign = "", groupalign = "",

  rowspan = "1", columnspan = "1"), Typesetting:-mtd(0,

  rowalign = "", columnalign = "", groupalign = "",

  rowspan = "1", columnspan = "1"), Typesetting:-mtd(0,

  rowalign = "", columnalign = "", groupalign = "",

  rowspan = "1", columnspan = "1"), Typesetting:-mtd(0,

  rowalign = "", columnalign = "", groupalign = "",
```

```
  rowspan = "1", columnspan = "1"), Typesetting:-mtd(0,

  rowalign = "", columnalign = "", groupalign = "",

  rowspan = "1", columnspan = "1"), rowalign = "",

  columnalign = "", groupalign = ""), foreground = "[0,0,0]",

  readonly = "false", align = "axis", rowalign = "baseline",

  columnalign = "center", groupalign = "{left}",

  alignmentscope = "true", columnwidth = "auto", width = "auto",

  rowspacing = "1.0ex", columnspacing = "0.8em",

  rowlines = "none", columnlines = "none", frame = "none",

  framespacing = "0.4em 0.5ex", equalrows = "false",

  equalcolumns = "false", displaystyle = "false", side = "right",

  minlabelspacing = "0.8em"))
```

;
Now set the various copper loss elements given the Rcon array. This block will be repeated later on with various loss elements.
Resistance of shorted backplane is Rbackplane
>
Attenuation constant of transmission line representation of the stack of wafers is Œ±stack
>
Attenuation constant of transmission line from window to cavity is Œ±TL
>
Propagation constant of transmission line in vacuum without the losses
From that, the complex characteristic impedance of the stack and of the transmission line are Z0stack and Z0
Complex propagation coefficient of the transmission line is Œ≥Œ≥, and of the stack Œ≥Œ≥stack.
>
Complex propagation coefficient of the transmission line representing the stack:
>
> Rbackplane := ln(bline/aline)*Rsurf/(2*Pi)*Rcon[1];

```
> `&alpha;stack` := Rsurf*(1/(2*r2)*Rcon[2] + Rcon[3]/DD)/(eta*ln(DD/(2*r2)));
> `&alpha;TL` := Rsurf*(Rcon[4]/aline + Rcon[5]/bline)/(2*eta*ln(bline/aline));
> Z0stack := eta*ln(DD/(2*r2))/(2*Pi)*(1 - `&alpha;stack`*I/beta);
> Z0 := eta*ln(bline/aline)/(2*Pi)*(1 - `&alpha;TL`*I/beta);
> `&gamma;&gamma;` := beta*I + `&alpha;TL`;
> `&gamma;&gamma;stack` := beta*I + `&alpha;stack`;
                  Rbackplane := 0.

                 &alpha;stack := 0.

                  &alpha;TL := 0.

             Z0stack := 46.43847058 - 0. I

               Z0 := 56.18960635 - 0. I

               &gamma;&gamma; := I beta

             &gamma;&gamma;stack := I beta

;
> beta := omega/cc;
                beta := 8.377580421

;
> Zastack := abs(Z0stack);
               Zastack := 46.43847058

;
> Za := abs(Z0);
                 Za := 56.18960635

;
# The capacitance of a single-wafer capacitor, with vacuum as dielectric:
> Csingle := epsilon0*S/dg;
                                       -12
            Csingle := 3.454012350 10

;
# The total capacitance of the stack as seen by the pulsed bias supply at average bias:
> `&epsilon;rep`*Csingle*Nwafers;
                                -9
                  1.544351535 10
```

```
;
# The coupler's transform turn number ratio is k:
> k := sqrt(RoverQ/(Za*Qe));
                  k := 0.003957430908

;
# Matching reactance for this coupler to satisfy frequency tune:
> Xtgt := `&Delta;&omega;`*Za*Qe/omega;
          Xtgt := 0.001117856700 &Delta;&omega;

# Definition of various functions and procedures for convenience:
>
# Here we compute the impedance of a single ferroelectric capacitor including the effects of the current flowing in the transition from the spacer to the wafer and back up the next spacer as a function of its relative permittivity œµ:
>
> RP := Rsurf/Pi;
> LP := dg*mu0/(2*Pi);
> CP := epsilon -> 2*Pi*epsilon0*epsilon*(1 - delta*I)/dg;
> `&gamma;P` := epsilon -> sqrt((RP + omega*LP*I)*omega*CP(epsilon)*I);
> ZP := epsilon -> sqrt(-I*(RP + omega*LP*I)/(omega*CP(epsilon)));
> N := 20;
                       N := 20

;
> `&epsilon;&epsilon;` := Vector([1 .. N]);
> Rout := Vector([1 .. N]);
> Xout := Vector([1 .. N]);
# Now we solve the Ricatti ODE for a number of vlues of the premittivity, to get the resistivity and reactance of the capacitor:
> for nn to N do
>    `&epsilon;n` := epsilon1 + (nn - 1)*(epsilon2 - epsilon1)/(N - 1);
>    `&epsilon;&epsilon;`(nn) := `&epsilon;n`;
>    ode := diff(Z(x), x)*ZP(`&epsilon;n`)/x + `&gamma;P`(`&epsilon;n`)*Z(x)^2 -
`&gamma;P`(`&epsilon;n`)*ZP(`&epsilon;n`)^2/x^2;
>    bics := Z(r1) = -I*10^4;
>    answer := dsolve({bics, ode}, numeric);
>    Rout(nn) := Re(rhs(answer(r2)[2]));
>    Xout(nn) := Im(rhs(answer(r2)[2]));
> end do;
> with(CurveFitting);
> Xspline := eps -> Spline(`&epsilon;&epsilon;`, Xout, eps, degree = 3);
Xspline := proc (eps) options operator, arrow; CurveFitting:-Spl\
```

```
  ine(`&epsilon;&epsilon;`, Xout, eps, degree = 3) end proc


;
> Rspline := eps -> Spline(`&epsilon;&epsilon;`, Rout, eps, degree = 3);
Rspline := proc (eps) options operator, arrow; CurveFitting:-Spl\

  ine(`&epsilon;&epsilon;`, Rout, eps, degree = 3) end proc


;
# The reactance and resistivity are approximated as Splines of 3rd degree and thus we have the impedance of the ferroelectric capacitor as a function of permittivity:
>
;
> ZC := eps -> Rspline(eps) + Xspline(eps)*I;
ZC := proc (eps) options operator, arrow; Rspline(eps)+I*Xspline\

  (eps) end proc


;
> if r1 = 0 then
>     ZC := epsilon -> -I/(omega*C*epsilon*(1 - delta*I));
> end if;
# The impedances of just the single capacitor in the two states:
>
;
> ZC1 := ZC(epsilon1);
       ZC1 := 0.00122183985522319 - 1.17082567056880 I

;
> ZC2 := ZC(epsilon2);
      ZC2 := 0.000931552457786097 - 0.864851266554209 I

;
# Function for the impedance presented by a transmission line, with characteristic impedance Zzero, of length "length" terminated by a load "Zload":
>
> Zline := (Zload, length, Zzero) -> Zzero*(Zload + Zzero*tanh(`&gamma;&gamma;`*length))/(Zzero + Zload*tanh(`&gamma;&gamma;`*length));
Zline := proc (Zload, length, Zzero) options operator, arrow;

   Zzero*(Zload+Zzero*tanh(`&gamma;&gamma;`*length))/(Zzero+Zloa\
```

```
  d*tanh(`&gamma;&gamma;`*length)) end proc

;
# Function for the impedance presented by the stack element of a transmission line, with characteristic impedance Zzero, of length "length" terminated by a load "Zload":
> Zstack := (Zload, length, Zzero) -> Zzero*(Zload +
Zzero*tanh(`&gamma;&gamma;stack`*length))/(Zzero +
Zload*tanh(`&gamma;&gamma;stack`*length));
Zstack := proc (Zload, length, Zzero) options operator, arrow;

   Zzero*(Zload+Zzero*tanh(`&gamma;&gamma;stack`*length))/(Zzero\

  +Zload*tanh(`&gamma;&gamma;stack`*length)) end proc

;
# Procedure for the figure of merit from given impedance:
> FoM := proc(impedance1, impedance2) local FigOfMerit; FigOfMerit :=
abs(1/2*(Im(impedance2) - Im(impedance1))/sqrt(Re(impedance1)*Re(impedance2))); end
proc;
FoM := proc (impedance1, impedance2) local FigOfMerit;

   FigOfMerit := abs((1/2)*(Im(impedance2)-Im(impedance1))/sqrt(\

  Re(impedance1)*Re(impedance2))) end proc

;
# Function to calcculate the impedance of the stack of ferroelectric capacitors and intermediate spaces. This includes the parasitic inductance of the stack. These various impedances are in series, so we add the impedances.
# The reactance due to the parasitic inductance of a spacer is calculated as if it were a section of transmission line of length Lspacer and characteristic impedance Zastack. The thickness of the ferroelectric capacitor is added to this transmission line.
# The impedance of the stack expressed as a function of the dielectric constant of the FE wafers:
>
> Zcap := (epsilon, lspacer) -> Zstack(Rbackplane, lspacer*(Nwafers - 1) + dg*Nwafers, Z0stack)
+ ZC(epsilon)*Nwafers;
   Zcap := proc (epsilon, lspacer) options operator, arrow;

      Zstack(Rbackplane, lspacer*(Nwafers-1)+dg*Nwafers,
```

Z0stack)+ZC(epsilon)*Nwafers end proc

;
The SFoM function (Static FoM) is the ratio of the imaginary part of the impedance (Imp) divided by the real part of the impedance (Imp). It is like the FoM, but it is a function of a particular state and not of two states. It signifies the ratio of the reactive power to the real power at the point where the impedance is Imp.
>
> SFoM := Imp -> Im(Imp)/Re(Imp);
SFoM := proc (Imp) options operator, arrow; Im(Imp)/Re(Imp) end

proc

;
The load impedance presented to the transmission line is the window's capacitance in series with a parallel circuit. The parallel circuit comprising the capacitor stack with the impedance Zt of the tuning capacitor in series, and that in parallel with the coupling reactance (Zcoup).
>
Zpar is the the function providing the impedance of this parallel circuit:
> if Case = A then
>   Zpar := (epsilon, lspacer, Zs) -> Zcap(epsilon, Lspacer);
> end if;
> if Case = B then
>   Zpar := (epsilon, lspacer, Zs) -> Zcap(epsilon, Lspacer)*Zs/(Zcap(epsilon, Lspacer) + Zs);
> end if;
> if Case = C or Case = D then
>   Zpar := (epsilon, lspacer, Zs) -> Zcap(epsilon, lspacer)*Zs/(Zcap(epsilon, lspacer) + Zs);
> end if;
Zpar := proc (epsilon, lspacer, Zs) options operator, arrow;

Zcap(epsilon, lspacer)*Zs/(Zcap(epsilon, lspacer)+Zs) end proc

;
Now we define the function Zfin to apply a transmission line of length L, in series with the window's impedance (if any), then in series with the parallel load defined above. This will be the final impedance seen by the cavity at its coupler's port, and it is a function of œμ, with various other parameters such as window capacitor, Zs, Nwafers, Lspacer, dg, etc.
> if Case = A then
>   Zfin := (epsilon, lspacer, Zs, Zw, L) -> Zline(Zpar(epsilon, lspacer, Zs), L, Z0);
> end if;

```
> if Case = B then
>     Zfin := (epsilon, lspacer, Zs, Zw, L) -> Zline(Zpar(epsilon, lspacer, Zs), L, Z0);
> end if;
> if Case = C then Zfin := (epsilon, lspacer, Zs, Zw, L) -> Zline(Zpar(epsilon, lspacer, Zs) + Zw, L,
Z0); end if;
> if Case = D then
>     Zfin := (epsilon, lspacer, Zs, Zw, L) -> Zline(Zline(Zpar(epsilon, lspacer, Zs), LA, Z0) + Zw, L,
Z0);
> end if;
 Zfin := proc (epsilon, lspacer, Zs, Zw, L) options operator,

    arrow; Zline(Zline(Zpar(epsilon, lspacer, Zs), LA, Z0)+Zw,

    L, Z0) end proc


;
# Final calculations:
>
> N := 40;
                    N := 40

;
> Zsvec := Vector([1 .. N]);
> Zsvec := Vector([1 .. N]);
> Zwvec := Vector([1 .. N]);
> Lspvec := Vector([1 .. N]);
> Lvec := Vector([1 .. N]);
> FoMvec := Vector([1 .. N]);
> invZwvec := Vector([1 .. N]);
> invZsvec := Vector([1 .. N]);
> Lsave := L;
> Lmin := L - 0.04;
> Lmax := L + 0.05;
> for nn to N do
>     Lvec(nn) := Lmin + (nn - 1)*(Lmax - Lmin)/N;
> end do;
                  Lsave := 0.677

                  Lmin := 0.637

                  Lmax := 0.727

;
```

```
> if Case = A then
>     eq3 := 0 = Im((Zcap(`&epsilon;rep`, Lspacer) + Z0*tanh(`&gamma;&gamma;stack`*LS))/(Z0 + Zcap(`&epsilon;rep`, Lspacer)*tanh(`&gamma;&gamma;stack`*LS)));
>     solution := solve(eq3, LS);
>     L := Re(solution);
> end if;
> if Case = B then
>     Zf1 := Im(Zcap(epsilon1, Lspacer));
>     Zf2 := Im(Zcap(epsilon2, Lspacer));
>     Zf3 := Im(Zcap(`&epsilon;rep`, Lspacer));
>     eq1 := -(ZS*Zf1/(Zf1 + ZS) + Za*tan(beta*LS))/(Za - ZS*Zf1*tan(beta*LS)/(Zf1 + ZS)) = (ZS*Zf2/(Zf2 + ZS) + Za*tan(beta*LS))/(Za - ZS*Zf2*tan(beta*LS)/(Zf2 + ZS));
>     eq3 := 0 = ZS*Zf3/(Zf3 + ZS) + Za*tan(beta*LS);
>     eqs := {eq1, eq3};
>     vars := {LS, ZS};
>     solution := solve(eqs, vars);
>     L := rhs(solution[2, 1]);
>     Zs := rhs(solution[2, 2])*I;
> end if;
> if Case = C then
>     ZC1 := Im(ZC(epsilon1))*Nwafers;
>     ZC2 := Im(ZC(epsilon2))*Nwafers;
>     ZC3 := Im(ZC(`&epsilon;rep`))*Nwafers;
>     for nn to N do
>         L := Lvec(nn);
>         th := tan(beta*L);
>         eq1 := Za*(ZW + ZS*(Zastack*tan(beta*lspacer*(Nwafers - 1)) + ZC1)/(ZS + Zastack*tan(beta*lspacer*(Nwafers - 1)) + ZC1) + Za*th)/(Za - (ZW + ZS*(Zastack*tan(beta*lspacer*(Nwafers - 1)) + ZC1)/(ZS + Zastack*tan(beta*lspacer*(Nwafers - 1)) + ZC1))*th) = -Za*Xfactor;
>         eq2 := Za*(ZW + ZS*(Zastack*tan(beta*lspacer*(Nwafers - 1)) + ZC2)/(ZS + Zastack*tan(beta*lspacer*(Nwafers - 1)) + ZC2) + Za*th)/(Za - (ZW + ZS*(Zastack*tan(beta*lspacer*(Nwafers - 1)) + ZC2)/(ZS + Zastack*tan(beta*lspacer*(Nwafers - 1)) + ZC2))*th) = Za*Xfactor;
>         eq3 := 0 = ZW + ZS*(Zastack*tan(beta*lspacer*(Nwafers - 1)) + ZC3)/(ZS + Zastack*tan(beta*lspacer*(Nwafers - 1)) + ZC3) + Za*th;
>         eqs := {eq1, eq2, eq3};
>         vars := {ZS, ZW, lspacer};
>         solution := solve(eqs, vars);
>         Zsvec(nn) := rhs(solution[2, 1]);
>         invZsvec(nn) := 1/Zsvec(nn);
>         Zwvec(nn) := rhs(solution[2, 2]);
>         invZwvec(nn) := 1/Zwvec(nn);
>         Lspvec(nn) := Re(rhs(solution[2, 3]));
```

```
>    end do;
>    L := Lsave;
>    th := tan(beta*L);
>    eq1 := Za*(ZW + ZS*(Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC1)/(ZS +
Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC1) + Za*th)/(Za - (ZW +
ZS*(Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC1)/(ZS +
Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC1))*th) = -Za*Xfactor;
>    eq2 := Za*(ZW + ZS*(Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC2)/(ZS +
Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC2) + Za*th)/(Za - (ZW +
ZS*(Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC2)/(ZS +
Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC2))*th) = Za*Xfactor;
>    eq3 := 0 = ZW + ZS*(Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC3)/(ZS +
Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC3) + Za*th;
>    eqs := {eq1, eq2, eq3};
>    vars := {ZS, ZW, lspacer};
>    solution := solve(eqs, vars);
>    Zs := rhs(solution[2, 1])*I;
>    Zw := rhs(solution[2, 2])*I;
>    Lspacer := Re(rhs(solution[2, 3]));
> end if;
> if Case = D then
>    ZC1 := Im(ZC(epsilon1))*Nwafers;
>    ZC2 := Im(ZC(epsilon2))*Nwafers;
>    ZC3 := Im(ZC(`&epsilon;rep`))*Nwafers;
>    for nn to N do
>       L := Lvec(nn);
>       th := tan(beta*L);
>       thA := tan(beta*LA);
>       eq1 := Za*(Za*(ZS*(Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC1)/(ZS +
Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC1) + Za*thA)/(Za -
ZS*(Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC1)*thA/(ZS +
Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC1)) + ZW + Za*th)/(Za -
(Za*(ZS*(Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC1)/(ZS +
Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC1) + Za*thA)/(Za -
ZS*(Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC1)*thA/(ZS +
Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC1)) + ZW)*th) = -Za*Xfactor;
>       eq2 := Za*(Za*(ZS*(Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC2)/(ZS +
Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC2) + Za*thA)/(Za -
ZS*(Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC2)*thA/(ZS +
Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC2)) + ZW + Za*th)/(Za -
(Za*(ZS*(Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC2)/(ZS +
Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC2) + Za*thA)/(Za -
ZS*(Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC2)*thA/(ZS +
Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC2)) + ZW)*th) = Za*Xfactor;
```

```
>       eq3 := 0 = (Za*ZS*(Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC3)/(ZS +
Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC3) + Za*thA)/(Za -
ZS*(Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC3)*thA/(ZS +
Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC3)) + ZW + Za*th;
>       eqs := {eq1, eq2, eq3};
>       vars := {ZS, ZW, lspacer};
>       solution := solve(eqs, vars);
>       StubCap := -10^12/(omega*rhs(solution[4, 1]));
>       if 20 < StubCap then
>          Nsolution := 4;
>       else
>          Nsolution := 3;
>       end if;
>       Zsvec(nn) := rhs(solution[Nsolution, 1]);
>       invZsvec(nn) := 1/Zsvec(nn);
>       Zwvec(nn) := rhs(solution[Nsolution, 2]);
>       invZwvec(nn) := 1/Zwvec(nn);
>       Lspvec(nn) := Re(rhs(solution[Nsolution, 3]));
>    end do;
>    L := Lsave;
>    th := tan(beta*L);
>    eq1 := Za*(Za*(ZS*(Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC1)/(ZS +
Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC1) + Za*thA)/(Za -
ZS*(Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC1)*thA/(ZS +
Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC1)) + ZW + Za*th)/(Za -
(Za*(ZS*(Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC1)/(ZS +
Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC1) + Za*thA)/(Za -
ZS*(Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC1)*thA/(ZS +
Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC1)) + ZW)*th) = -Za*Xfactor;
>    eq2 := Za*(Za*(ZS*(Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC2)/(ZS +
Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC2) + Za*thA)/(Za -
ZS*(Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC2)*thA/(ZS +
Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC2)) + ZW + Za*th)/(Za -
(Za*(ZS*(Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC2)/(ZS +
Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC2) + Za*thA)/(Za -
ZS*(Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC2)*thA/(ZS +
Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC2)) + ZW)*th) = Za*Xfactor;
>    eq3 := 0 = (Za*ZS*(Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC3)/(ZS +
Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC3) + Za*thA)/(Za -
ZS*(Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC3)*thA/(ZS +
Zastack*tan(beta*(lspacer*(Nwafers - 1) + dg*Nwafers)) + ZC3)) + ZW + Za*th;
>    eqs := {eq1, eq2, eq3};
>    vars := {ZS, ZW, lspacer};
>    solution := solve(eqs, vars);
```

```
>     StubCap := -10^12/(omega*rhs(solution[4, 1]));
>     if 20 < StubCap then
>        Nsolution := 4;
>     else
>        Nsolution := 3;
>     end if;
>     Zs := rhs(solution[Nsolution, 1])*I;
>     Zw := rhs(solution[Nsolution, 2])*I;
>     Lspacer := Re(rhs(solution[Nsolution, 3]));
> end if;
# Series capacitance vs transmission line length
> if Case = C or Case = D then
>     dataplot(Lvec, -10^12*invZsvec/omega);
> end if;

# Window capacitance vs transmission line length
> if Case = C or Case = D then
>     dataplot(Lvec, -10^12*invZwvec/omega);
> end if;

# Spacer length (mm) vs transmission line length (m)
> if Case = C or Case = D then
>     dataplot(Lvec, 1000*Lspvec);
> end if;

> `&Delta;&omega;1` := omega/(2*Qe)*Im(Zfin(epsilon1, Lspacer, Zs, Zw, L))/Za;
          &Delta;&omega;1 := -25132.7597743504

;
> `&Delta;&omega;2` := omega/(2*Qe)*Im(Zfin(epsilon2, Lspacer, Zs, Zw, L))/Za;
          &Delta;&omega;2 := 25132.7297469017

;
> `&Delta;f1` := `&Delta;&omega;1`/(2*Pi);
            &Delta;f1 := -4000.00295110673

;
> `&Delta;f2` := `&Delta;&omega;2`/(2*Pi);
            &Delta;f2 := 3999.99817208983

;
> dPreactive := 2*Ucav*(`&Delta;&omega;1` - `&Delta;&omega;2`);
                                  5
          dPreactive := -4.52389405691269 10
```

```
;
# The figure of merit with no copper losses:
> FoM(Zfin(epsilon1, Lspacer, Zs, Zw, L), Zfin(epsilon2, Lspacer, Zs, Zw, L));
                  143.398419272870

;
> dPreactive1 := 2*Ucav*`&Delta;&omega;1`;
                                     5
          dPreactive1 := -2.26194837969154 10

;
> dPreactive2 := 2*Ucav*`&Delta;&omega;2`;
                                    5
          dPreactive2 := 2.26194567722115 10
>
;
# The power absorbed in the ferroelectric (under Rcon being (00000))
> PFabs1 := dPreactive1/SFoM(Zfin(epsilon1, Lspacer, Zs, Zw, L));
              PFabs1 := 1763.33143996436

;
> PFabs2 := dPreactive2/SFoM(Zfin(epsilon2, Lspacer, Zs, Zw, L));
              PFabs2 := 1411.04927955178

;
# The average temperature rise in the wafer, assuming cooling on both sides and only dielectric
loss (Rcon must be all zero) for this to be correct:
> Twafer1 := PFabs1*dg/(6*S*TherCon*Ncool);
             Twafer1 := 13.3961296618815

;
> Twafer2 := PFabs2*dg/(6*S*TherCon*Ncool);
             Twafer2 := 10.7198219686721

;
# An approximate estimate of the peak RF voltage across one FE wafer. To get the correct
number, we set copper resistivity to zero everywhere.
> VRFwafer1 := sqrt(2*abs(PFabs1)*abs(Zcap(epsilon1, Lspacer))/(Nwafers*delta));
            VRFwafer1 := 2813.11611841379

;
> VRFwafer2 := sqrt(2*abs(PFabs2)*abs(Zcap(epsilon2, Lspacer))/(Nwafers*delta));
            VRFwafer2 := 2691.01430820130
```

```
;
# Now lets check the losses in the inner conductor of the transmission line:
> Rcon := array([0, 0, 0, 1, 0]);
RconAssign(Typesetting:-mtable(Typesetting:-mtr(Typesetting:-mtd(

  0, rowalign = "", columnalign = "", groupalign = "",

  rowspan = "1", columnspan = "1"), Typesetting:-mtd(0,

  rowalign = "", columnalign = "", groupalign = "",

  rowspan = "1", columnspan = "1"), Typesetting:-mtd(0,

  rowalign = "", columnalign = "", groupalign = "",

  rowspan = "1", columnspan = "1"), Typesetting:-mtd(1,

  rowalign = "", columnalign = "", groupalign = "",

  rowspan = "1", columnspan = "1"), Typesetting:-mtd(0,

  rowalign = "", columnalign = "", groupalign = "",

  rowspan = "1", columnspan = "1"), rowalign = "",

  columnalign = "", groupalign = ""), foreground = "[0,0,0]",

  readonly = "false", align = "axis", rowalign = "baseline",

  columnalign = "center", groupalign = "{left}",

  alignmentscope = "true", columnwidth = "auto", width = "auto",

  rowspacing = "1.0ex", columnspacing = "0.8em",

  rowlines = "none", columnlines = "none", frame = "none",

  framespacing = "0.4em 0.5ex", equalrows = "false",

  equalcolumns = "false", displaystyle = "false", side = "right",

  minlabelspacing = "0.8em"))
```

```
;
> Rbackplane := ln(bline/aline)*Rsurf/(2*Pi)*Rcon[1];
> `&alpha;stack` := Rsurf*(1/(2*r2)*Rcon[2] + Rcon[3]/DD)/(eta*ln(DD/(2*r2)));
> `&alpha;TL` := Rsurf*(Rcon[4]/aline + Rcon[5]/bline)/(2*eta*ln(bline/aline));
> Z0stack := eta*ln(DD/(2*r2))/(2*Pi)*(1 - `&alpha;stack`*I/beta);
> Z0 := eta*ln(bline/aline)/(2*Pi)*(1 - `&alpha;TL`*I/beta);
> `&gamma;&gamma;` := beta*I + `&alpha;TL`;
> `&gamma;&gamma;stack` := beta*I + `&alpha;stack`;
                    Rbackplane := 0.

                   &alpha;stack := 0.

               &alpha;TL := 0.0003719330824

              Z0stack := 46.43847058 - 0. I

            Z0 := 56.18960635 - 0.002494607326 I

      &gamma;&gamma; := 0.0003719330824 + 8.377580421 I

         &gamma;&gamma;stack := 0. + 8.377580421 I

;
> PTLIabs1 := dPreactive1/SFoM(Zfin(epsilon1, Lspacer, Zs, Zw, L)) - PFabs1;
               PTLIabs1 := 110.865796318810

;
> PTLIabs2 := dPreactive2/SFoM(Zfin(epsilon2, Lspacer, Zs, Zw, L)) - PFabs2;
               PTLIabs2 := 120.705311556602

;
# Now turn on losses in all copper elements:
> Rcon := array([1, 1, 1, 1, 1]);
RconAssign(Typesetting:-mtable(Typesetting:-mtr(Typesetting:-mtd(

  1, rowalign = "", columnalign = "", groupalign = "",

  rowspan = "1", columnspan = "1"), Typesetting:-mtd(1,

  rowalign = "", columnalign = "", groupalign = "",

  rowspan = "1", columnspan = "1"), Typesetting:-mtd(1,
```

```
rowalign = "", columnalign = "", groupalign = "",

rowspan = "1", columnspan = "1"), Typesetting:-mtd(1,

rowalign = "", columnalign = "", groupalign = "",

rowspan = "1", columnspan = "1"), Typesetting:-mtd(1,

rowalign = "", columnalign = "", groupalign = "",

rowspan = "1", columnspan = "1"), rowalign = "",

columnalign = "", groupalign = ""), foreground = "[0,0,0]",

readonly = "false", align = "axis", rowalign = "baseline",

columnalign = "center", groupalign = "{left}",

alignmentscope = "true", columnwidth = "auto", width = "auto",

rowspacing = "1.0ex", columnspacing = "0.8em",

rowlines = "none", columnlines = "none", frame = "none",

framespacing = "0.4em 0.5ex", equalrows = "false",

equalcolumns = "false", displaystyle = "false", side = "right",

minlabelspacing = "0.8em"))


;
> Rbackplane := ln(bline/aline)*Rsurf/(2*Pi)*Rcon[1];
> `&alpha;stack` := Rsurf*(1/(2*r2)*Rcon[2] + Rcon[3]/DD)/(eta*ln(DD/(2*r2)));
> `&alpha;TL` := Rsurf*(Rcon[4]/aline + Rcon[5]/bline)/(2*eta*ln(bline/aline));
> Z0stack := eta*ln(DD/(2*r2))/(2*Pi)*(1 - `&alpha;stack`*I/beta);
> Z0 := eta*ln(bline/aline)/(2*Pi)*(1 - `&alpha;TL`*I/beta);
> `&gamma;&gamma;` := beta*I + `&alpha;TL`;
> `&gamma;&gamma;stack` := beta*I + `&alpha;stack`;
              Rbackplane := 0.0007672053190

             &alpha;stack := 0.0006575754049

                &alpha;TL := 0.0005177308510
```

```
         Z0stack := 46.43847058 - 0.003645061529 I

           Z0 := 56.18960635 - 0.003472493399 I

      &gamma;&gamma; := 0.0005177308510 + 8.377580421 I

    &gamma;&gamma;stack := 0.0006575754049 + 8.377580421 I

;
> if Case = B then
>    Lstub := arctan(Im(Zs)/Za)/beta;
>    Zs := Z0*tanh(`&gamma;&gamma;`*Lstub);
> end if;
# Harmonic power loss Ph calculation
>
> Ep := 174000;
> delta2 := 2*delta;
> `&gamma;&gamma;2` := sqrt(2)*`&alpha;TL` + 2*I*beta;
> I0 := VRFwafer1/abs(ZC1);
                    Ep := 174000

               delta2 := 0.001900000000

                                (1/2)
  &gamma;&gamma;2 := 0.0005177308510 2      + 16.75516084 I

               I0 := 600.669294568668

;
> VRFstack := I0*abs(Zcap(epsilon1, Lspacer));
             VRFstack := 5121.89306537061

;
> Vh := evalf(VRFstack^2/(2*dg*Nwafers*Ep*epsilon1));
               Vh := 217.199938353148

;
# Zhc is the impedance of the capacitors at the harmonic frequency
> Zhc := Nwafers/(2*I*omega*Csingle*(1 - delta2*I)*epsilon1);
          Zhc := 0.004540448049 - 2.389709501 I

;
# Zhp is the impedance of the parasitic inductance of the stack at the harmonic frequency
```

```
>
> Zhp := evalf(Za*tanh(`&gamma;&gamma;2`*Nwafers*Lspacer));
          Zhp := 0.002585869740 + 43.48625671 I

;
> if Case = B then
>    Zhs := evalf(Za*tanh(`&gamma;&gamma;2`*Lstub));
> end if;
# Zhtl is the impedance of the open transmission line at the harmonic frequency
> Zhtl := evalf(Za/tanh(`&gamma;&gamma;2`*L));
          Zhtl := 0.03151022950 + 20.36253575 I

;
> if Case = A then
>    Zh := Zhc + Zhp + Zhtl;
> end if;
>
;
> if Case = B then
>    Zh := Zhc + Zhp + Zhs*Zhtl/(Zhs + Zhtl);
> end if;
> if Case = C or Case = D then
>    Zh := evalf(Zhc + Zhp + Zs/2*(Zhtl + Zw/2)/(Zs/2 + Zhtl + Zw/2));
> end if;
           Zh := 0.008953427907 + 37.24136918 I

;
> Ph := evalf(abs(Vh)^2*Re(Zh)/(2*abs(Zh)^2));
               Ph := 0.152274589189034

;
#
# Reflection coefficient and Smith Chart
>
> `&Gamma;func` := eps -> (Zfin(eps, Lspacer, Zs, Zw, L) - Z0)/(Zfin(eps, Lspacer, Zs, Zw, L) + Z0);
 &Gamma;func := proc (eps) options operator, arrow; (Zfin(eps,

   Lspacer, Zs, Zw, L)-Z0)/(Zfin(eps, Lspacer, Zs, Zw, L)+Z0)

   end proc


;
```

```
> plot([[Re(`&Gamma;func`(xx)), Im(`&Gamma;func`(xx)), xx = epsilon1 .. epsilon2], [cos(tt),
sin(tt), tt = 0 .. 2*Pi]], color = ["Red", "Blue"], style = [point, line]);

# Grand summary of all parameters and results:
> Cpar := Nwafers*Csingle*epsilon2;
                                    -9
              Cpar := 1.790560002 10

;
> Tw;
                      50

;
> epsilon1;
                  96.40944255

;
> epsilon2;
                   129.6000

;
> `&epsilon;rep`;
                  111.7795319

;
> 1.00001*delta;
                 0.0009500095000

;
> Qe;
                    50000

;
> Zfin1 := Zfin(epsilon1, Lspacer, Zs, Zw, L);
     Zfin1 := 0.741492918982460 - 56.1891933024507 I

;
> Zfin2 := Zfin(epsilon2, Lspacer, Zs, Zw, L);
     Zfin2 := 0.670529705597002 + 56.1889963612585 I

;
> Za;
                  56.18960635
```

```
;
# The reactive power in the two states:
> dPreactive1;
                                      5
                  -2.26194837969154 10

;
> dPreactive2;
                                      5
                   2.26194567722115 10

;
# "State" Figire of Merit and the total absorbed power in the tuner in each state:
> SFoM1 := SFoM(Zfin(epsilon1, Lspacer, Zs, Zw, L));
                SFoM1 := -75.7784624289580

;
> SFoM2 := SFoM(Zfin(epsilon2, Lspacer, Zs, Zw, L));
                SFoM2 := 83.7979225860412

;
> Pabs1 := dPreactive1/SFoM1;
                Pabs1 := 2984.94889865588

;
> Pabs2 := dPreactive2/SFoM2;
                Pabs2 := 2699.28610091575

;
# The power absorbed just in the ferroelectric material:
> PFabs1;
                    1763.33143996436

;
> PFabs2;
                    1411.04927955178

;
# The average temperature rise in the wafer, due to just the dielectric loss.
> Twafer1;
                    13.3961296618815

;
> Twafer2;
```

```
                 10.7198219686721

;
# Bias voltage across a single wafer:
> Vbias;
                   7200.0000

;
# An  estimate of the peak RF voltage across one FE wafer.
> VRFwafer1;
                 2813.11611841379

;
> VRFwafer2;
                 2691.01430820130

;
# The reactive power (difference between the two states) sent from the cavity to the tuner:
> dPreactive;
                                 5
               -4.52389405691269 10

;
# Power loss in the inner conductor of the transmission line:
>
;
> PTLIabs1;
                 110.865796318810

;
> PTLIabs2;
                 120.705311556602

;
> Qfrt1 := abs(Ucav*omega*SFoM1/dPreactive1);
                                    6
            Qfrt1 := 3.78892032459677 10

;
> Qfrt2 := abs(Ucav*omega*SFoM2/dPreactive2);
                                    6
            Qfrt2 := 4.18989804236131 10

;
```

```
# Capacitor FoM (loss just in FE material) :
> FoM(ZC(epsilon1), ZC(epsilon2));
                  143.398398215237

;
# Stack FoM (loss in stack, including spacers and FE material):
> FoM(Zcap(epsilon1, Lspacer), Zcap(epsilon2, Lspacer));
                  84.5033362838723

;
> r1;
                      0.0165

;
> r2;
                      0.0196

;
> DD;
                      0.085

;
> plotlimit := 10*abs(`&Delta;f1`);
>
;
# Plot of the frequency tuning of the cavity as a function of the permittivity of the ferroelectric.
#
#
> plot(max(min(f/(2*Qe*Za)*Im(Zfin(xx, Lspacer, Zs, Zw, L)), plotlimit), -plotlimit), xx = epsilon1
.. epsilon2);

>
;
> `&Delta;f` := abs((`&Delta;&omega;1` - `&Delta;&omega;2`)/(2*Pi));
              &Delta;f := 8000.00112319656

;
# Total final Figure of Merit:
> FoMtot := FoM(Zfin(epsilon1, Lspacer, Zs, Zw, L), Zfin(epsilon2, Lspacer, Zs, Zw, L));
               FoMtot := 79.6873749619759

;
> L;
                      0.677
```

```
;
> Lspacer;
                    0.009827344357

;
> if Case = B then
>    Lstub;
> end if;
> if Case = C or Case = D then
>    Cs := Re(10^12/(omega*Zs*I));
>    Cw := 10^12/(omega*Zw*I);
> end if;
                  Cs := 39.17797902

                  Cw := 5.469636211

;
```

Assuming that the capacitor Cs has a length œÄ*DD and width bb, what is the gap gg that will yield the above value of Cs? (Fringe fields included)

```
> bb := 0.012;
                     bb := 0.012

;
> fsolve(38.73 = 10^12*epsilon0*(Pi*DD*(1.15*bb/gg + 1.4*10^0.222) + 4.12*gg*10^0.728), gg);
                   0.0009865754923
```