Analysis and Optimization of an RF Baseband Transimpedance Amplifier with Low-Pass Filter Structure and Positive Feedback

Cisko Loos
University of Twente

June 30, 2017

Abstract **-** The goal of this assignment is to find ways to optimize the selectivity of a double-sided TIA filter for use in multiband transceivers. Its circuit features passive R and C components for attenuation and capacitive positive feedback to achieve high linearity. The analysis of the TIA filter is only in baseband and assumes ideal mixing. An equivalent circuit is derived, from which a system of equations is found. A general way of solving a system of equations with higher order solutions using MATLAB is found, and used to find expressions for the transfer function and input impedance of the TIA filter. These expressions are found to be correct by simulating their frequency response in Simulink and comparing them to circuit simulations in LTspice. Approximating the transfer function to a standard second order form proved to be appropriate in the relevant frequency range. System properties like natural frequency and quality factor are extracted from this approximation. Next, more insight is gained in the influence of some capacitors on the transfer function of the TIA filter by stepping their parameter values in simulation. These values are then tuned to achieve an f_{-3dB} point at 9.7MHz, a maximum of +1.3dB overshoot and a roll-off of -44dB between 10MHz and 100MHz, providing slightly more optimal selectivity while keeping the overshoot within acceptable range. This report provides a MATLAB implementation that has been proven to significantly ease the derivation of large equations. It can be used to automate much of the design and optimization process of higher order systems.

Contents

1 Introduction

Multiband transceivers have become a standard for mobile phones and other communication devices. They enable devices to communicate simultaneously over multiple frequency bands. The biggest challenge in multiband transceivers is to be able to receive a low power signal while simultaneously transmitting with high power at another frequency close to that of the weak received signal. The received signal is first frequency mixed to bring it to baseband, after which it is filtered and amplified. To filter out the unwanted transmitted signal, the filter needs to have a steep roll-off and high linearity, especially because in many RF applications RX and TX signals are very close to each other. Steep, higher order filtering can be accomplished by cascading N-path filter stages [2] [3], but here are often multiple active g_m cells involved, which all contribute to noise and nonlinearity. Resistive positive feedback has also been used in a receiver front-end [4] which eases input impedance matching at low noise. However, for better selectivity at high linearity and low noise, a more promising solution has recently been proposed [1]. It uses switches for very high linearity mixing and a Transimpedance Amplifier (TIA) with capacitive positive feedback for amplification and filtering. It provides a steep roll-off and high linearity while keeping a low noise figure (NF) [1].

The goal of this work is to explore ways to further optimize the selectivity of the proposed TIA filter [1]. Its electrical circuit schematic is shown in Figure 1. The TIA comprises of amplification g_mR_o with feedback filtering through R_f and C_f. R_{a1} and R_{a2} provide some attenuation, and together with C_{a1} and C_{a2} form the passive cross-coupling circuit for the capacitive positive feedback and higher order filtering.

The rest of this report is structured as follows. Section 2 contains the derivation of the expressions for the Transfer Function (TF) and Input Impedance (Z_{in}) of the circuit in Figure 1. Assumptions made for this derivation are also stated there, as well as an analysis on the influence of certain components. In Section 3, the found TF's are validated by simulation. In Section 4, the TF is approximated to a standard form, from which system properties like natural frequency and quality factor are extracted. Also, the influences of the analyzed components are discussed and tuned to provide optimal roll-off between 10MHz and 100MHz.

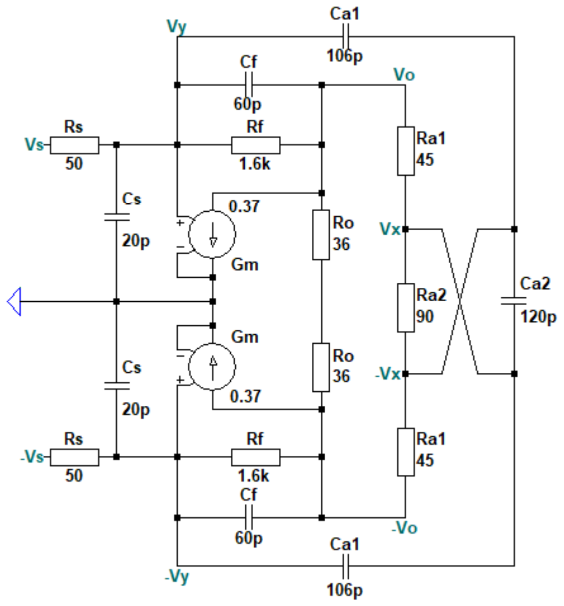


Figure 1: Schematic circuit model of the complete BB TIA with low-pass filter structure and positive feedback [1].

2 Circuit analysis

This section is built up as follows. First, the circuit shown in Figure 1 is simplified to an equivalent circuit to ease the derivation of a system of node equations. Then, symbolic analysis is used to derive the expressions of the TF and Z_{in}. Finally, the influences of some components are analyzed by stepping parameter values.

2.1 Equivalent circuit derivation

For easier derivation, some assumptions in analyzing the circuit of Figure 1 are made. The analysis is purely in baseband, the mixing circuit is assumed ideal and is not taken into account. It is also assumed that the circuit is perfectly symmetric, so the opposite sides are assumed to be actually perfectly opposite.

Since the circuit is double-sided and assumed perfectly symmetrical, an equivalent circuit can comprise of only half the complete circuit. However, removing one side should be done with caution in this case as there is cross-coupling over R_{a2} and C_{a2} (Figure 1).

First, a ground line is drawn in the middle of the circuit, separating the positive and negative sides of the circuit as much as possible. Since perfect symmetry is assumed, the ground line also includes virtual grounds. To complete the separation, some components need to be split to create a virtual ground in between. In this case, R_{a2} and C_{a2} are split as illustrated in Figure 4. Splitting the two components is done according to the Miller theorem [5]. This theorem states that an impedance connecting two voltages can be split into two grounded impedances, given that the voltages are proportional to each other.

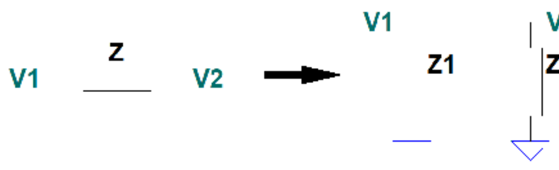


Figure 2: Illustration of the application of the Miller theorem. When $V_2 = kV_1$ with k a constant, Z can be split into two grounded impedances with $Z_1 = \frac{Z}{1-k}$ and $Z_2 = \frac{kZ}{k-1}$ [5].

The Miller theorem is illustrated in Figure 2. If the voltages V_1 and V_2 are proportional to each other according to $V_2 = kV_1$ with k a constant, impedance Z can be split into Z_1 and Z_2, with

$$\begin{aligned} Z_1 &= \frac{Z}{1-k} \\ Z_2 &= \frac{kZ}{k-1} \text{ [5]} \end{aligned}$$

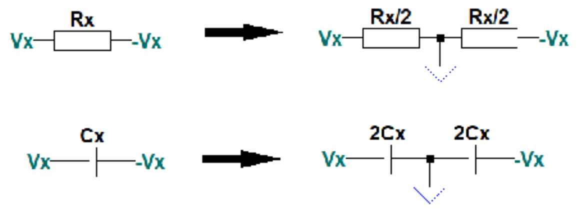


Figure 3: The Miller theorem [5] applied to a resistor and capacitor both connected between opposite voltages, so $k = -1$. For the resistor, $Z = R_x$ so $Z_1 = Z_2 = \frac{R_x}{2}$. For the capacitor, $Z = \frac{1}{sC_x}$ so $Z_1 = Z_2 = \frac{1}{s \cdot 2C_x}$.

Figure 3 illustrates how a resistor and capacitor each connected between opposite voltages can be split using the Miller theorem. With opposite voltages, $k = -1$ so a resistor with impedance $Z = R_x$ can be split into

$$\begin{aligned} Z_1 &= \frac{R_x}{1 - -1} = \frac{R_x}{2} \\ Z_2 &= \frac{-R_x}{-1-1} = \frac{R_x}{2} \end{aligned}$$

A capacitor C_x has impedance $Z = \frac{1}{sC_x}$, which can be split into

$$\begin{aligned} Z_1 &= \frac{\frac{1}{sC_x}}{1 - -1} = \frac{1}{s \cdot 2C_x} \\ Z_2 &= \frac{-\frac{1}{sC_x}}{-1-1} = \frac{1}{s \cdot 2C_x} \end{aligned}$$

Finally, filling in R_{a2} and C_{a2} gives the component values

$$R_{Z1} = R_{Z2} = \frac{R_{a2}}{2}$$
$$C_{Z1} = C_{Z2} = 2C_{a2}$$

which are shown in Figure 4. Now, the ground line includes the virtual grounds in between the split components.

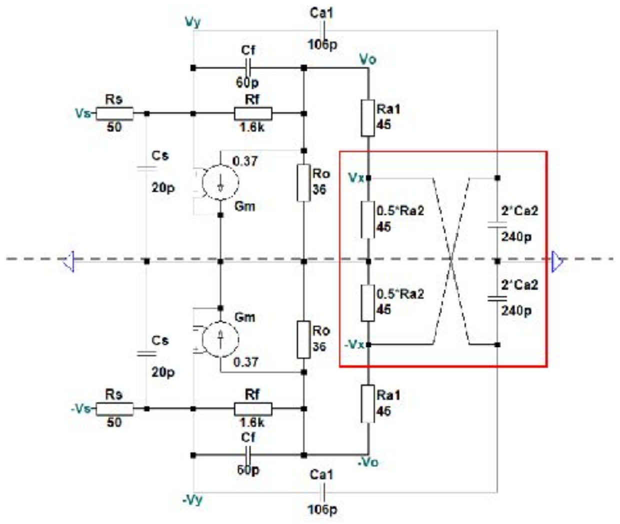
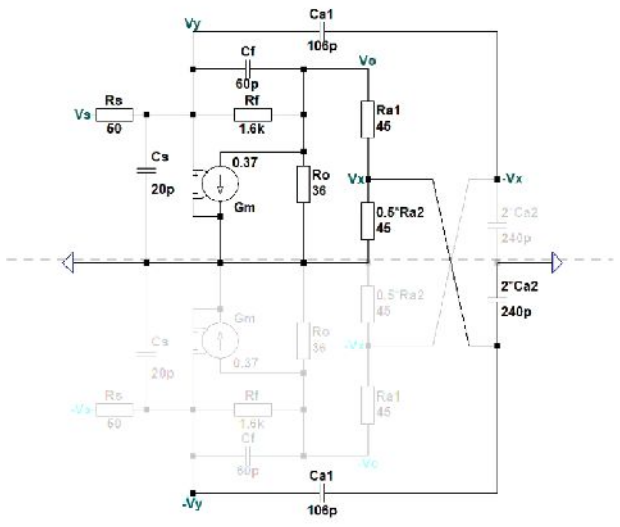
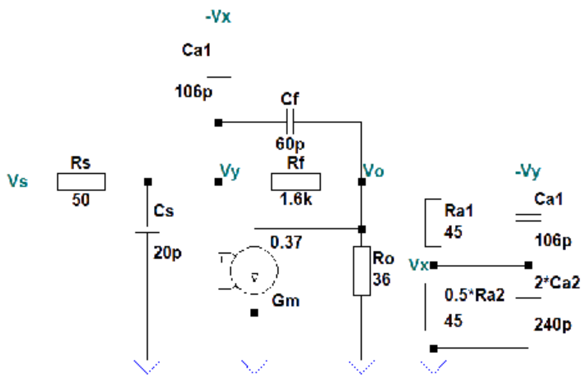


Figure 4: Schematic circuit model of the BB TIA filter with an added ground line. R_{a2} and C_{a2} are split according to the Miller theorem [5] and highlighted in the red box.

The circuit of Figure 4 is further simplified by removing most of the negative side of the circuit, but since there is still cross-coupling, some components cannot be removed. To retain all characteristics of the original circuit, all currents in and out of each positive node should be taken into account. Assuming perfect symmetry, the negative nodes are defined as the opposite of their corresponding positive nodes. Therefore, in the simplification of the circuit of Figure 4, all current paths from the positive nodes to any ground or negative nodes are taken into account. The rest is removed as shown in Figure 5a. Then the circuit is redrawn for clarity as shown in Figure 5b.



(a)



(b)

Figure 5: Equivalent schematic models of the BB TIA filter with most of the negative side of the circuit removed. Figure 5b is simply a redraw of Figure 5a. All currents going in and out of the positive nodes V_s, V_x, V_y, V_o are fully defined. Since perfect symmetry is assumed, the negative nodes $-V_x$ and $-V_y$ are defined as the opposite of their corresponding positive nodes.

The final simplification of the equivalent circuit is done by combining parallel impedances. The equivalent impedance of two parallel impedances is given by

$$Z_{eq} = \frac{Z_1 Z_2}{Z_1 + Z_2}$$

In the equivalent circuit of Figure 5b, there are two instances with parallel resistor and capacitor, C_f, R_f and $\frac{R_{a2}}{2}, 2C_{a2}$. The respective equivalent impedances Z_f and Z_{a2} are therefore

$$Z_f = \frac{R_f \cdot \frac{1}{sC_f}}{R_f + \frac{1}{sC_f}} = \frac{R_f}{1 + sR_fC_f}$$
$$Z_{a2} = \frac{\frac{R_{a2}}{2} \cdot \frac{1}{s \cdot 2C_{a2}}}{\frac{R_{a2}}{2} + \frac{1}{s \cdot 2C_{a2}}} = \frac{\frac{R_{a2}}{2}}{1 + R_{a2}C_{a2}}$$

The parallel components are replaced with their equivalent impedances into the circuit of Figure 5b. The final equivalent circuit of the circuit of Figure 1 is shown in Figure 6. This circuit will be analyzed in the next section, so the directions of the currents are also defined.

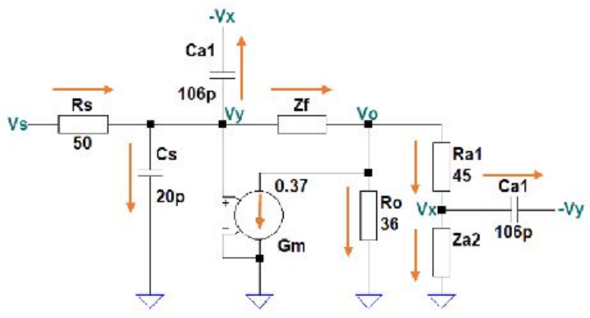


Figure 6: Equivalent schematic circuit model of the BB TIA filter. The arrows indicate how the directions of the currents are defined for analysis. The impedances Z_f and Z_{a2} are equivalent impedances of parallel components C_f, R_f and $\frac{R_{a2}}{2}, 2C_{a2}$, respectively. Their expressions are $Z_f = \frac{R_f}{1+sR_fC_f}$ and $Z_{a2} = \frac{\frac{R_{a2}}{2}}{1+R_{a2}C_{a2}}$.

2.2 Symbolic analysis of the equivalent circuit

The next step towards the TF and Z_{in} is to analyze the equivalent circuit of Figure 6. The directions of the currents are already defined. First, the node current equations are derived at the nodes V_y, V_o and V_x according to Kirchoff's law of currents:

$$I_{Rs} - I_{Cs} - I_{Ca1} - I_{Zf} = 0$$
$$I_{Zf} - I_{Gm} - I_{Ro} - I_{Ra1} = 0$$
$$I_{Ra1} - I_{Za2} - I_{Ca1} = 0$$

Next, the expressions for all currents are filled in:

$$\frac{V_s - V_y}{R_s} - V_y \cdot sC_s - (V_y + V_x) \cdot sC_{a1} - \frac{V_y - V_o}{Z_f} = 0$$
$$\frac{V_y - V_o}{Z_f} - V_y \cdot G_m - \frac{V_o}{R_o} - \frac{V_o - V_x}{R_{a1}} = 0$$
$$\frac{V_o - V_x}{R_{a1}} - \frac{V_x}{Z_{a2}} - (V_y + V_x) \cdot sC_{a1} = 0$$

Then the equations are rewritten to a system of equations for the node voltages V_y, V_o and V_x:

$$V_y = V_s\left(\frac{1}{\frac{R_s}{R_s//R_f} + sR_s\left(C_s + C_f + C_{a1}\right)}\right) - V_x\left(\frac{s\left(R_s//R_f\right)C_{a1}}{1 + s\left(R_s//R_f\right)\left(C_s + C_f + C_{a1}\right)}\right) +$$

$$+ V_o\left(\frac{1 + sR_fC_f}{\frac{R_f}{R_s//R_f} + sR_f\left(C_s + C_f + C_{a1}\right)}\right)$$

$$V_x = V_o\left(\frac{R_{a2}}{2R_{a1} + R_{a2} + sR_{a1}R_{a2}\left(C_{a1} + 2C_{a2}\right)}\right) - V_y\left(\frac{sR_{a1}R_{a2}C_{a1}}{2R_{a1} + R_{a2} + sR_{a1}R_{a2}\left(C_{a1} + 2C_{a2}\right)}\right)$$

$$V_o = V_x\left(\frac{1}{1 + \frac{R_{a1}}{R_f//R_o} + sR_{a1}C_f}\right) + V_y\left(\frac{1 - G_mR_f + sR_fC_f}{1 + \frac{R_f}{R_{a1}//R_o} + sR_fC_f}\right)$$

The system is solvable, since there are three equations with three unknowns V_y, V_o and V_x. The source V_s is a given parameter and is needed to calculate the transfer function from the system of equations.

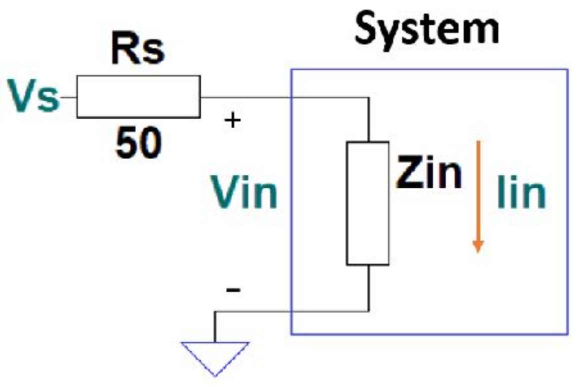


Figure 7: Schematic representation of the input impedance of the BB TIA filter. The source voltage V_s and source resistance R_s are not part of the TIA filter itself. The input voltage of the circuit is therefore not equal to V_s. If V_{in} and I_{in} are known, the input impedance is calculated by $Z_{in} = \frac{V_{in}}{I_{in}}$.

For the impedance matching of the circuit, the input voltage and current need to be known. For this, the system can be drawn like Figure 7. Comparing this to Figure 6, it can be seen that $V_{in} = V_y$ and $I_{in} = \frac{V_s - V_y}{R_s}$. Finally, adding this equation for I_{in} to the system of equations makes four equations with four unknowns, since I_{in} adds no other new variables to the system.

Working out the system of equations by hand proved to be very complicated and prone to errors, as the expressions became very large. This is why the further derivation of the TF was done using symbolic analysis tools in MATLAB, version R2016a. The full MATLAB script for solving the system of equations can be found in Appendix A. In the rest of this section, the script will be explained in more detail.

First, the components need to be defined as symbols so no numeric value is given to them yet. Also, for easier implementation, some parallel resistances found in the system of equations are defined beforehand as follows:

```
syms s Rs Rf Ra Rb Ro Gm Cs Cf Ca Cb;    %Create component symbols
Rsf = Rs*Rf/(Rs+Rf);                     %Create parallel resistances for
Rfo = Rf*Ro/(Rf+Ro);                     % easier implementation
Rao = Ra*Ro/(Ra+Ro);
```

Next, the system of equations needs to be defined as follows. For clarity, the relations between the three node voltages are defined separately as subfunctions H, following first the letter of the node voltage left of the equator, then the letter of the node voltage it relates to.

```
syms Vs Vx Vy Vo Iin;                        %Create node voltage symbols
EQ1 = Vo == Hox*Vx + Hoy*Vy;                 %Create system of equations
EQ2 = Vy == Hys*Vs - Hyx*Vx + Hyo*Vo;
EQ3 = Vx == Hxo*Vo - Hxy*Vy;
EQ4 = Iin == (Vs - Vy)/Rs;
```

The system can be solved simply using the command "solve". Finally, the expressions for the TF and Z_{in} can be derived from these solutions:

```
TF = solve([EQ1, EQ2, EQ3, EQ4], [Vx, Vy, Vo, Iin]);  %Find solutions
TFo = (TF.Vo)/Vs;        %Vo/Vs      %Find transfer functions
TFx = (TF.Vx)/Vs;        %Vx/Vs
TFy = (TF.Vy)/Vs;        %Vy/Vs
Zin = (TF.Vy)/(TF.Iin); %Vy/Iin     %Find input impedance
```

Now that MATLAB holds the symbolic expressions for the TF and Z_{in}, a numeric solution can be found. These expressions were all found to be of the form

$$\frac{n_1 + n_2 s + n_3 s^2}{d_1 + d_2 s + d_3 s^2 + d_4 s^3}$$

with each their own numerator coefficients $\{n_1, ..., n_3\}$ and denominator coefficients $\{d_1, ..., d_4\}$. Defining numeric values for the components and using the command "eval" would give numeric expressions for the TF and Z_{in}, but the coefficients would not be approximated as double precision variables. Instead, their exact values were given as fractions of very large numbers. To be able to check whether the found TF and Z_{in}, their coefficients first needed to be converted to double precision variables. For this, they also needed to be separated from the expressions, as the expressions could not be converted to double precision themselves.

The coefficients were found as follows. First, the expressions were split into numerators and denominators using "numden". Then, the coefficients could be derived from them using "coeffs":

```
[TFo_n, TFo_d] = numden(TFo);    %Find numerators & denominators

Nos = coeffs(TFo_n, s);       %Find coefficients with first the DC term
Dos = coeffs(TFo_d, s);       % (N(1), D(1)), then increasing orders of s
```

Finally, the numeric values of the coefficients can be found using "eval". However, these numeric values needed to be defined separately and initialized first for efficiency, as for-loops were used. This way, the numeric values were automatically double precision.

```
No = zeros(length(Nos));     %Initialize numeric coefficients
Do = zeros(length(Dos));

for i = 1:length(Nos)        %Calculate all coefficients into doubles
    No(i) = eval(Nos(i));    % for use in Linear Analysis tool
end
for i = 1:length(Dos)
    Do(i) = eval(Dos(i));
end
```

Again, the full MATLAB script can be found in Appendix A.

The TF's for V_o, V_y and V_x and the expression for Z_{in} were found to be as follows:

$$\frac{V_o}{V_s} = \frac{-3.83\cdot 10^{6} - s\cdot 2.97\cdot 10^{-2} + s^2\cdot 4.84\cdot 10^{-12}}{5.42\cdot 10^{5} + s\cdot 1.09\cdot 10^{-2} + s^2\cdot 1.33\cdot 10^{-10} + s^3\cdot 2.26\cdot 10^{-20}}$$

$$\frac{V_y}{V_s} = \frac{4.10\cdot 10^{5} + s\cdot 4.70\cdot 10^{-3} + s^2\cdot 4.84\cdot 10^{-12}}{5.42\cdot 10^{5} + s\cdot 1.09\cdot 10^{-2} + s^2\cdot 1.33\cdot 10^{-10} + s^3\cdot 2.26\cdot 10^{-20}}$$

$$\frac{V_x}{V_s} = \frac{-1.91\cdot 10^{6} - s\cdot 9.41\cdot 10^{-4} - s^2\cdot 1.48\cdot 10^{-12}}{5.42\cdot 10^{5} + s\cdot 1.09\cdot 10^{-2} + s^2\cdot 1.33\cdot 10^{-10} + s^3\cdot 2.26\cdot 10^{-20}}$$

$$Z_{in} = \frac{4.10\cdot 10^{5} + s\cdot 4.70\cdot 10^{-3} + s^2\cdot 4.84\cdot 10^{-12}}{2.65\cdot 10^{3} + s\cdot 1.25\cdot 10^{-4} + s^2\cdot 2.56\cdot 10^{-12} + s^3\cdot 4.53\cdot 10^{-22}}$$

The expressions found will be validated by simulation in Section 3.

2.3 Component influences

The circuit of Figure 1 is simulated in LTspice IV to examine the influences of certain components to the TF of V_x: C_s, C_{a1} and C_{a2}. An LTspice directive will be used to sweep the component values of these capacitors (.step param). The resulting TF's will be plotted into one graph. Removing each capacitor will also be examined by setting their value to $C = 0$F, as their impedance becomes $Z_C = \frac{1}{sC} = \infty$, simulating an open circuit. Since LTspice is limited in plotting clear multiple graphs, the plot data is imported into MATLAB to be able to make clearer plots. The procedure used for importing LTspice plot data into MATLAB is described in Appendix B. The values of all components that are not being examined are as shown in Figure 1, if not stated otherwise.

First, the influence of C_s is examined. The resulting TF's of the parameter sweep are shown in Figure 8. The first thing that can be seen is that C_s mostly influences high-frequency attenuation, as higher values attenuate more. Also the cut-off frequency f_{-3dB} is lowered with higher values of C_s.

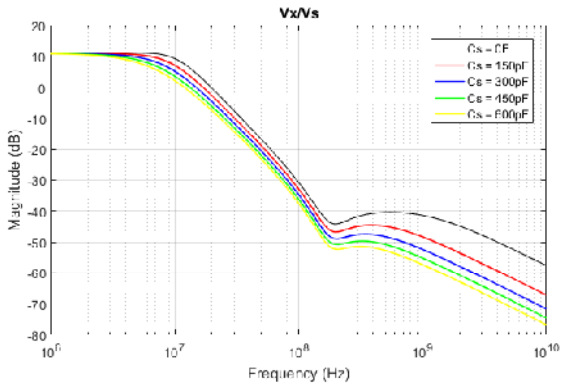


Figure 8: Resulting V_x/V_s for different values of C_s. All other components have their original values. The black line simulates the circuit while C_s is removed.

Next, C_{a1} is examined. This is done with two different values of C_{a2}, with first its original value of 120pF as shown in Figure 1. The resulting TF's are shown in Figure 9. The most notable influence is that setting C_{a1} around 300pF introduces overshoot in the TF. Higher or lower values have much less notable overshoot. Increasing the value also lowers the f_{-3dB} point and smoothens the roll-off around 100MHz. Around $C_{a1} = 150pF$, the TF has a sharp roll-off with little overshoot.

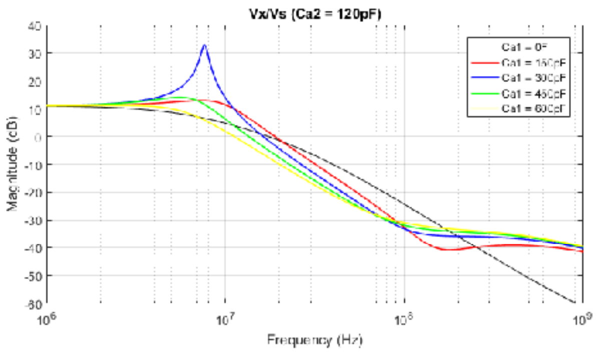


Figure 9: Resulting V_x/V_s for different values of C_{a1}. All other components have their original values. The black line simulates the circuit while C_{a1} is removed.

Again, C_{a1} is examined with the same steps, but now C_{a2} is doubled to 240pF. The resulting TF's are shown in Figure 10. The most notable difference with Figure 9 is that there is less overshoot at values of C_{a1} up to 300pF. However, higher values seem to have more offset than before.

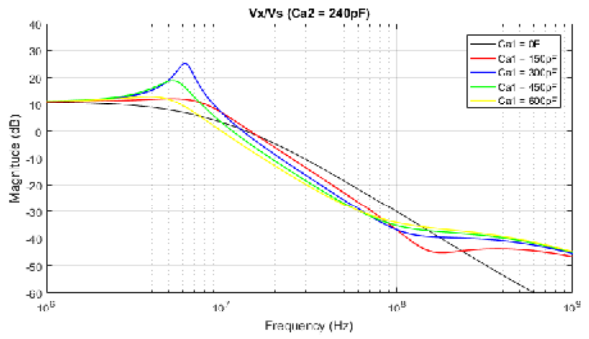


Figure 10: Resulting V_x/V_s for different values of C_{a1}. C_{a2} is set to 240pF, the rest of the components have their original values. The black line simulates the circuit while C_{a1} is removed.

Next, C_{a2} will be examined for two different values of C_{a1}, with first its original value of 106pF as shown in Figure 1. The resulting TF's are shown in Figure 11. The most notable thing is that C_{a2} sharpens the roll-off around 10MHz. C_{a2} also causes the TF to be first order above 1GHz, as removing it flattens the TF from there. Again, increasing the value decreases the f_{-3dB} point.

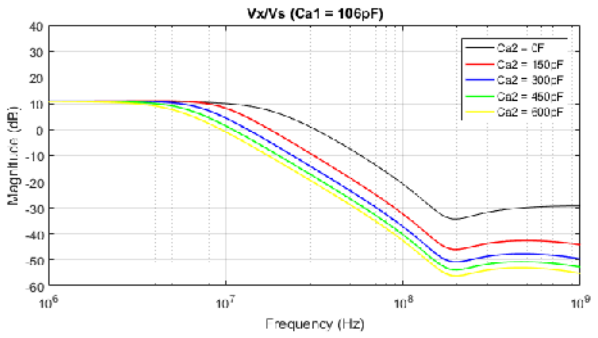


Figure 11: Resulting V_x/V_s for different values of C_{a2}. All other components have their original values. The black line simulates the circuit while C_{a2} is removed.

Finally, C_{a2} will be examined with the same steps, but with C_{a1} doubled to 212pF. The resulting TF's are shown in Figure 12. As seen earlier in Figure 9, setting C_{a1} around 300pF introduces overshoot. It can also be seen that C_{a2} dampens this overshoot more with increasing values.

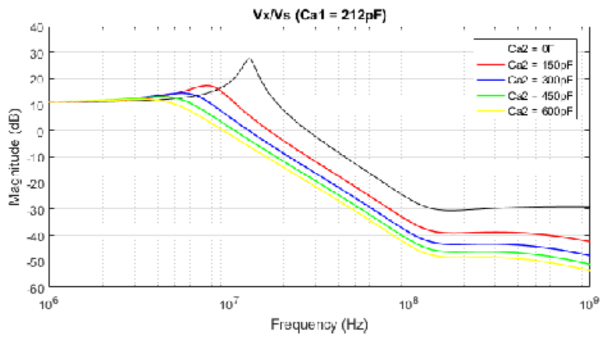


Figure 12: Resulting V_x/V_s for different values of C_{a2}. C_{a1} is set to 212pF, the rest of the components have their original values. The black line simulates the circuit while C_{a2} is removed.

The parameter sweeps provide possibilities for optimizing the TIA filter by tuning the three examined capacitors. Overshoot can be decreased by decreasing C_{a1} or increasing C_{a2}. High frequencies can be attenuated more by increasing C_s or C_{a2}. However, it should be taken into account that each of the three capacitors influences the f_{-3dB} point, so proper trade-offs should be made during the optimization process.

In Section 4.3, the values of C_{a1} and C_{a2} will be tuned to optimize the filtering for a f_{-3dB} point of 10MHz, a maximum overshoot of +1.5dB and an optimal roll-off between 10MHz and 100MHz.

3 Model validation

In this section, the TF's found in Section 2.2 through symbolic analysis will be validated by simulation. First, Simulink and the Linear Analysis tool in MATLAB will be used to plot the frequency response of the found TF's. They will then be compared with the frequency response of the circuit of Figure 1 simulated in LTspice.

Simulink contains a block called "Transfer Function", which needs the TF coefficients as inputs. In Section 2.2, these coefficients have all been found using MATLAB. Filling in the coefficients in the Simulink block would look like Figure 13. The coefficients were filled in manually, which is not very efficient. It would be interesting to find a way to fill in coefficients automatically, especially when large amounts of higher order expressions need to be analyzed.

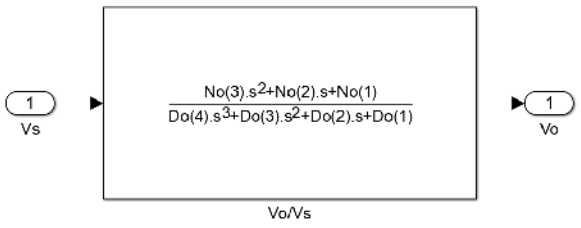
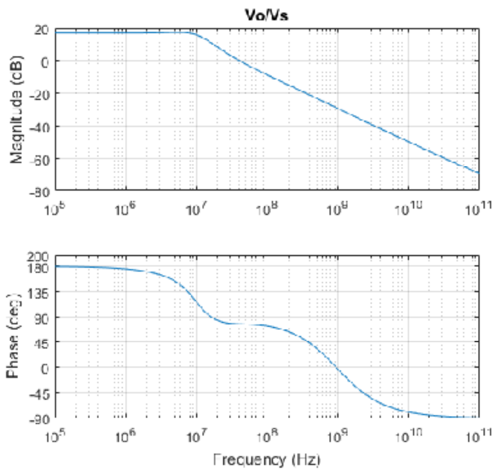


Figure 13: Simulink model of the TF V_o/V_s. The coefficients have all been calculated using the MATLAB script in Appendix A. To use the Linear Analysis tool on the block, its input and output need to be connected to any other block.

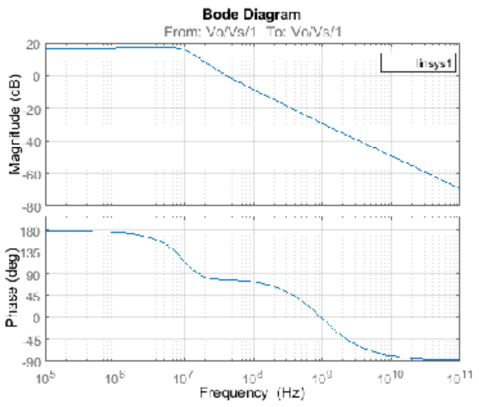
The input and output of the Transfer Function block need to be connected to another block, because only then the Linear Analysis tool works. This tool is used to create a Bode plot of the TF.

LTspice simulation of the complete circuit is used to find the frequency response of the circuit. This data is then exported to MATLAB to be able to make the comparison with the Linear Analysis of the found TF easier. Again, this is done as described in Appendix B.

A side-by-side comparison of the frequency responses can be found in Figures 14, 15, 16 and 17. The responses found by simulation in LTspice are shown in Figures 14a, 15a, 16a and 17a, the responses found by Linear Analysis in Simulink are shown in Figures 14b, 15b, 16b and 17b. Since each plot of the found TF's and Z_{in} is identical to its corresponding plot found by simulating the complete circuit, the TF's and Z_{in} found in Section 2.2 must be correct.

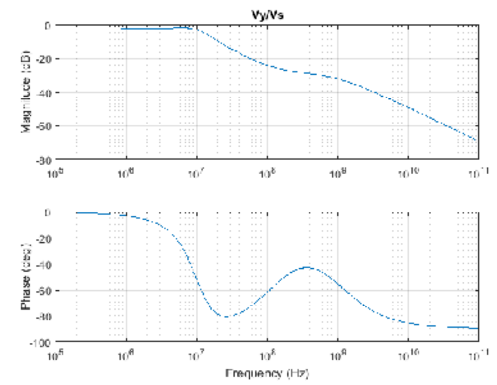


(a) Frequency response of the TIA filter at V_o, simulated in LTspice. The data was exported to MATLAB, where the plot is made.

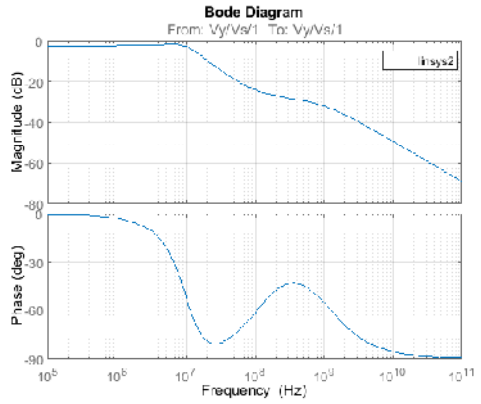


(b) Frequency response of the TIA filter at V_o, found by symbolic analysis in MATLAB.

Figure 14

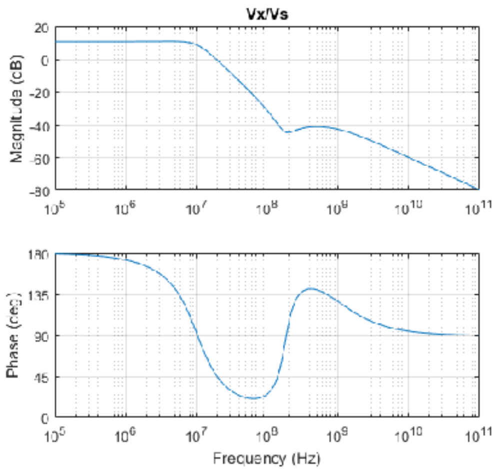


(a) Frequency response of the TIA filter at V_y, simulated in LTspice. The data was exported to MATLAB, where the plot is made.

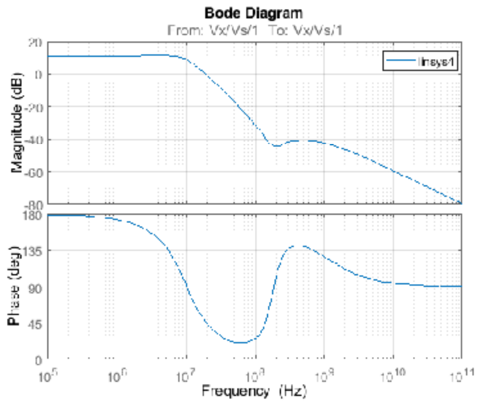


(b) Frequency response of the TIA filter at V_y, found by symbolic analysis in MATLAB.

Figure 15

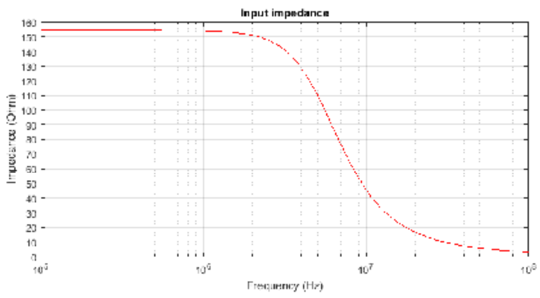


(a) Frequency response of the TIA filter at V_x, simulated in LTspice. The data was exported to MATLAB, where the plot is made.

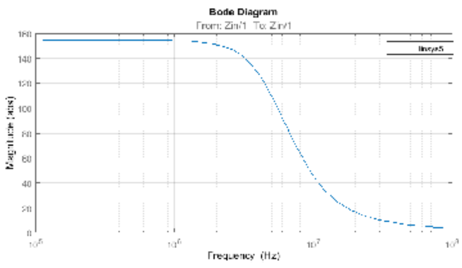


(b) Frequency response of the TIA filter at V_x, found by symbolic analysis in MATLAB.

Figure 16



(a) Frequency dependency of input impedance Z_{in} of the TIA filter, simulated in LTspice. The data was exported to MATLAB, where the plot is made.



(b) Frequency dependency of input impedance Z_{in} of the TIA filter, found by symbolic analysis in MATLAB.

Figure 17

4 Design exploration

4.1 System properties from the TF

From the TF found in Section 2.2, various system properties can be extracted. The TF $\frac{V_x}{V_s}$ is of most interest as it showed the steepest roll-off (Figure 16). An approximation can be made that omits high frequency components of the TF, which is therefore only viable up to a certain frequency. Omitting the s and s^2 terms in the numerator and the s^3 term in the denominator results in the following TF:

$$\frac{V_x}{V_s} = \frac{-1.91 \cdot 10^6}{5.42 \cdot 10^5 + s \cdot 1.09 \cdot 10^{-2} + s^2 \cdot 1.33 \cdot 10^{-10}}$$

Next, dividing by the coefficient of the s^2 term in the denominator gives a standard form of a second order low-pass filter:

$$\frac{V_x}{V_s} = \frac{-1.44 \cdot 10^{16}}{s^2 + 8.23 \cdot 10^7 s + 4.08 \cdot 10^{15}}$$

To validate this approximation, its frequency response is plotted together with the response of the original TF, which was already found to be correct in Section 3. The frequency responses are shown in Figure 18. It can be seen that the approximation is valid for the frequency range of interest, up to 100MHz. The passband and second order roll-off fall within this range.

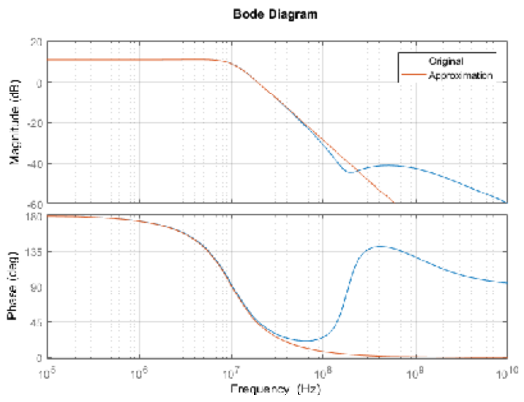


Figure 18: Bode plot of the TF V_x/V_s and its approximation. It is clear that the approximation is not valid for frequencies higher than 100MHz.

The standard form of a second order low-pass filter is as follows:

$$H(s) = \frac{K\omega_0^2}{s^2 + \frac{\omega_0}{Q}s + \omega_0^2}$$

with K the passband gain, Q the quality factor and $f_0 = \frac{\omega_0}{2\pi}$ the natural frequency of the TF. Using this standard form, the passband gain, natural frequency and quality factor of the TIA filter can be extracted from the approximation of $\frac{V_x}{V_s}$. The bandwidth is defined as $\Delta f = \frac{f_0}{Q}$:

$$\begin{aligned}
K &= \frac{-1.44 \cdot 10^{16}}{4.08 \cdot 10^{15}} = -3.53 = 11.0\text{dB} \\
f_0 &= \frac{\omega_0}{2\pi} = \frac{\sqrt{4.08 \cdot 10^{15}}}{2\pi} = 10.2\text{MHz} \\
Q &= \frac{\omega_0}{8.23 \cdot 10^7} = \frac{\sqrt{4.08 \cdot 10^{15}}}{8.23 \cdot 10^7} = 0.78 \\
\Delta f &= \frac{10.2 \cdot 10^6}{0.78} = 13.1\text{MHz}
\end{aligned}$$

The calculation of these system properties can be implemented into the MATLAB script in Appendix A, since it calculates all coefficients of the TF. The system properties directly relate to these coefficients.

4.2 Input impedance

In Section 2.2, an expression for Z_{in} has also been found. However, to save time, the frequency response was plotted using simulation in LTspice instead. Again, the data is exported to MATLAB as described in Appendix B. Z_{in} is shown in Figure 17. One notable thing is that Z_{in} converges to 0Ω near 100MHz, where C_s shorts V_{in} to ground due to the high frequency (Figure 1).

Impedance matching would need an input impedance of 50Ω, but Z_{in} seems to be around 155Ω. However, the actual input impedance needed for impedance matching is this TIA input impedance Z_{in} in parallel with the mixer circuit output impedance [1]. The analysis in this report does not include the mixer circuit, so the shown Z_{in} of the TIA filter is not representable for impedance matching.

4.3 Optimizing roll-off 10MHz-100MHz

In this section, the parameter sweeps from Section 2.3 will be used to tune the values of C_{a1} and C_{a2} to find a specific optimum. This being a f_{-3dB} point close to 10MHz, an overshoot of at most +1.5dB and an optimal roll-off between 10MHz and 100MHz at V_x.

Looking at Figure 10, setting the values $C_{a2} = 240$pF and $C_{a1} = 150$pF shown as the red line provides a solid starting point. The overshoot is +1.17dB, the f_{-3dB} point is around 9.3MHz and the roll-off between 10MHz and 100MHz is -43.8dB. Tuning C_{a2} to 220pF sets the f_{-3dB} point closer to 10MHz while keeping the overshoot within the acceptable range. The resulting frequency response at V_x is shown in Figure 19, together with the frequency response at V_x of the circuit with its original values from Figure 1. The responses are found using LTspice simulation, after which the data is imported into MATLAB to create the plot as described in Appendix B.

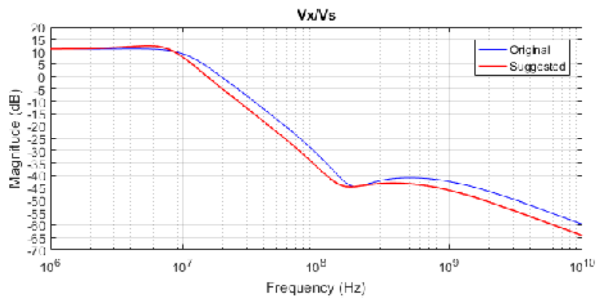


Figure 19: Frequency response of the TIA filter with both its original component values (blue line) and with tuned C_{a1} and C_{a2} (red line). The tuned frequency response has a f_{-3dB} point closer to 10MHz and a more optimal roll-off between 10MHz and 100MHz.

The original circuit has an overshoot of less than +0.1dB, an f_{-3dB} point at 11MHz and a roll-off of -40dB between 10MHz and 100MHz. The circuit with tuned values has an overshoot of +1.3dB, its f_{-3dB} point is at 9.7MHz and the roll-off between 10MHz and 100MHz is -44dB. This proves that the tuned values slightly improve the filtering with respect to the original circuit, according to the given requirements.

5 Conclusion

A TIA filter circuit proposed by [1] is analyzed. An equivalent circuit is derived, which enabled the derivation of a system of equations. MATLAB was used to rewrite those equations to expressions for the transfer function and input impedance of the TIA filter. By approximating the TF, the natural frequency, quality factor and bandwidth of the TF are found.

By simulation, more insight is given in the influence of some capacitors, and they are tuned to achieve optimal roll-off between 10MHz - 100MHz. The goal of this assignment has thus partly been achieved, as further optimization of the TIA filter could be done using the TF and the found system properties. MATLAB implementation has been proven to significantly ease the derivation of large equations, and can be used to automate much of the design and optimization process.

6 Appendices

Appendix A: MATLAB script, solving the system of equations

```
%% TF and Zin, symbolic analysis %%

syms s Rs Rf Ra Rb Ro Gm Cs Cf Ca Cb;   %Create component symbols
Rsf = Rs*Rf/(Rs+Rf);                    %Create parallel resistances for
Rfo = Rf*Ro/(Rf+Ro);                    % easier implementation
Rao = Ra*Ro/(Ra+Ro);

Hy1 = Rsf/(1+s*Rsf*(Cs+Cf+Ca));             %Create subfunctions found
Hys = Hy1/Rs;                               % from basic circuit theory
Hyx = Hy1*s*Ca;
Hyo = Hy1*(1+s*Rf*Cf)/Rf;
Hox = 1/(1+Ra/Rfo+s*Ra*Cf);
Hoy = (1-Gm*Rf+s*Rf*Cf)/(1+Rf/Rao+s*Rf*Cf);
Hxo = Rb/(2*Ra+Rb+s*Ra*Rb*(Ca+2*Cb));
Hxy = Hxo*s*Ra*Ca;

syms Vs Vx Vy Vo Iin;                       %Create node voltage symbols
EQ1 = Vo == Hox*Vx + Hoy*Vy;                %Create system of equations
EQ2 = Vy == Hys*Vs - Hyx*Vx + Hyo*Vo;
EQ3 = Vx == Hxo*Vo - Hxy*Vy;
EQ4 = Iin == (Vs - Vy)/Rs;
TF = solve([EQ1, EQ2, EQ3, EQ4], [Vx, Vy, Vo, Iin]);  %Find solutions
TFo = (TF.Vo)/Vs;       %Vo/Vs      %Find transfer functions
TFx = (TF.Vx)/Vs;       %Vx/Vs
TFy = (TF.Vy)/Vs;       %Vy/Vs
Zin = (TF.Vy)/(TF.Iin); %Vy/Iin     %Find input impedance

[TFo_n, TFo_d] = numden(TFo);   %Find numerators & denominators
[TFx_n, TFx_d] = numden(TFx);
[TFy_n, TFy_d] = numden(TFy);
[Zin_n, Zin_d] = numden(Zin);
Nos = coeffs(TFo_n, s);      %Find coefficients with first the DC term
Dos = coeffs(TFo_d, s);      % (N(1), D(1)), then increasing orders of s
Nxs = coeffs(TFx_n, s);
Dxs = coeffs(TFx_d, s);
Nys = coeffs(TFy_n, s);
Dys = coeffs(TFy_d, s);
NZs = coeffs(Zin_n, s);
DZs = coeffs(Zin_d, s);

DCos = Nos(1)/Dos(1);        %Find DC gains
DCxs = Nxs(1)/Dxs(1);
DCys = Nys(1)/Dys(1);
DC_Zs = NZs(1)/DZs(1);       %Find DC input impedance

%% TF and Zin, numeric solution %%

Rs = 50;              %Create component values
```

```
Rf = 1600;
Ra = 45;
Rb = 90;
Ro = 36;
Gm = 0.37;
Cs = 20*10^(-12);
Cf = 60*10^(-12);
Ca = 106*10^(-12);
Cb = 120*10^(-12);

No = zeros(length(Nos));     %Initialize numeric coefficients
Do = zeros(length(Dos));
Nx = zeros(length(Nxs));
Dx = zeros(length(Dxs));
Ny = zeros(length(Nys));
Dy = zeros(length(Dys));
NZ = zeros(length(NZs));
DZ = zeros(length(DZs));

for i = 1:length(Nos)        %Calculate all coefficients into doubles
    No(i) = eval(Nos(i));    % for use in Linear Analysis tool
end
for i = 1:length(Dos)
    Do(i) = eval(Dos(i));
end
for i = 1:length(Nxs)
    Nx(i) = eval(Nxs(i));
end
for i = 1:length(Dxs)
    Dx(i) = eval(Dxs(i));
end
for i = 1:length(Nys)
    Ny(i) = eval(Nys(i));
end
for i = 1:length(Dys)
    Dy(i) = eval(Dys(i));
end
for i = 1:length(NZs)
    NZ(i) = eval(NZs(i));
end
for i = 1:length(DZs)
    DZ(i) = eval(DZs(i));
end

DCo = 20*log10(abs(eval(DCos)));     %Calculate DC gains in dB
DCx = 20*log10(abs(eval(DCxs)));
DCy = 20*log10(abs(eval(DCys)));
DC_Z = eval(DC_Zs);                  %Calculate DC Zin
```

Appendix B: Exporting LTspice plots to MATLAB

Many of the LTspice data that had to be exported to MATLAB were frequency responses. Selecting the graph window and clicking File→Export enables the user to make a .txt file with all datapoints of the graph. Frequency plots are saved in the following format:

```
Freq.   V(n006)
1.00000000000000e+006   (1.09737893154537e+001dB,1.72885052280355e+002deg)
1.02329299228075e+006   (1.09744159919974e+001dB,1.72717846567782e+002deg)
1.04712854805090e+006   (1.09750704152713e+001dB,1.72546640850514e+002deg)
...
```

The first column is the frequency. The magnitude and phase are saved with their units, between brackets and separated with a comma.

MATLAB is able to read .txt files in columns and rows, but only if there are just numbers. This is why the .txt file of the LTspice data often had to be edited to remove the brackets, commas and units. Notepad++ was used for this, since it enables to edit multiple rows at the same time in 'column mode'. Using Edit→Begin/End Select and Alt+Shift+↓, the brackets, units and commas of the entire dataset could be removed at the same time. The result should look like this:

```
Freq.   V(n006)
1.00000000000000e+006   1.09737893154537e+001   1.72885052280355e+002
1.02329299228075e+006   1.09744159919974e+001   1.72717846567782e+002
1.04712854805090e+006   1.09750704152713e+001   1.72546640850514e+002
...
```

Now, MATLAB can read the .txt file. An example of reading an exported file with frequency, magnitude and phase is shown below.

```
data = dlmread('filename.txt','',1,0);  %Read from (edited) .txt file
Freq = data(1:length(data),1);          %Find frequency (column 1)
Mag = data(1:length(data),2);           %Find magnitude (column 2)
Pha = data(1:length(data),3);           %Find phase (column 3)
```

The parameters (' ',1,0) in the dlmread function means that no delimiters are used, and the data is read starting from the second row and first column. Delimiters are the characters with which the data is separated, but the dlmread command allows only one character as delimiter. In this case, there are multiple delimiters with multiple characters, so the delimiter parameter cannot be used and the .txt file has to be edited manually. Perhaps a better way can be found to automate the exporting of data from LTspice to MATLAB more in the future.

References

[1] Y. Lien, E. Klumperink, B. Tenbroek, J. Strange, B. Nauta (2017). **A Mixer-First Receiver with Enhanced Selectivity by Capacitive Positive Feedback Achieving +39dBm IIP3 and ¡3dB Noise Figure for SAW-Less LTE Radio**.

[2] M. Darvishi et al. (2013). **Design of Active N-Path Filters**. *IEEE J. Solid State Circuits*, vol. 48, NO. 12. p. 2962-2976.

[3] R. Chen et al. (2015). **Dual-Carrier Aggregation Receiver With Reconfigurable Front-End RF Signal Conditioning**. *IEEE J. Solid State Circuits*, vol. 50, NO. 8, p. 1874-1888.

[4] A. Nejdel et al. (2015). **A Positive Feedback Passive Mixer-First Receiver Front-End**. *IEEE Radio Frequency Integrated Circuits Symp.*, p. 79-82.

[5] J. Millman, C. C. Halkias. (1972). **Integrated Electronics: Analog and Digital Circuits and Systems**. International Student Edition. Tokyo: *McGraw-Hill Kogakusha.* Chapter 8-11, **Miller's Theorem and its Dual**; p. 255-258.