

# «DJ» EMI Filter für Schaltnetzteil

Fachbericht

Windisch, 13.06.2019



**Hochschule** Hochschule für Technik - FHNW

**Studiengang** Elektro- und Informationstechnik

**Auftraggeber** Dr. Luca Dalessandro

**Betreuer** Prof. Dr. Sebastian Gaulocher  
Prof. Peter Niklaus  
Prof. Dr. Richard Gut  
Dr. Anita Gertiser  
Pascal Buchschacher

**Autoren** **Gruppe 1**  
Niklaus Schwegler  
Lukas von Däniken  
Pascal Puschmann  
Simon Rohrer  
Marco Binder

**Version** 2.0

## **Zusammenfassung**

### **Abstract**

Die Firma Schaffner entwickelt EMI-Filter. Um diese zu modellieren, benötigen sie eine Simulationssoftware. Die Hauptanwendung dieser Software besteht darin, die Einfügedämpfung in Abhängigkeit der parasitären Parameter der Bauteile, über ein Frequenzspektrum bis 30Mhz zu visualisieren. Die Software sollte plattformunabhängig und modular erweiterbar sein. Aus diesem Grund basiert das Programm auf der Model-View-Controller-Struktur, geschrieben in Java. Für die Berechnung wurden jeweils vereinfachte Ersatzschaltungen für Gleich- und Gegen-taktstörungen modelliert. Die einzelnen Bauteile sind modular in die Software implementiert. Diese einzelnen Bauteile werden nun anhand der Ersatzschaltung aneinander gereiht, wodurch diese berechnet wird. Auf der Benutzeroberfläche sind für alle verstellbaren Parameter Slider erzeugt, um die Werte um  $\pm 30\%$  zu verändern. Dabei werden die Auswirkungen für beide Signale direkt in zwei einzelnen Plots sichtbar. Die Software ist somit in der Lage, Filter direkt miteinander zu vergleichen. Sie kann zudem einfach auf andere Schaltbilder erweitert werden.

## Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
<b>2</b>	<b>Grundlagen</b>	<b>2</b>
2.1	Elektrotechnik . . . . .	2
2.1.1	Passive 2-Tore . . . . .	2
2.1.2	Kettenmatrix . . . . .	5
2.1.3	Streuparameter . . . . .	6
2.1.4	Parasitäre Paramter . . . . .	8
2.1.5	Einfügedämpfung . . . . .	8
2.1.6	Aufbau eines EMI-Filters . . . . .	9
2.1.7	Gleich- Gegentaktschaltung . . . . .	9
2.2	Programmieren . . . . .	10
2.2.1	MVC-Struktur . . . . .	10
<b>3</b>	<b>Ermitteln der Einfügedämpfung</b>	<b>11</b>
3.1	Gleichtaktschaltung . . . . .	11
3.2	Gegentaktschaltung . . . . .	15
<b>4</b>	<b>Software</b>	<b>18</b>
4.1	Übersicht . . . . .	18
4.2	View . . . . .	19
4.3	Controller . . . . .	22
4.4	Model . . . . .	22
4.5	Trace . . . . .	23
<b>5</b>	<b>Testkonzept</b>	<b>25</b>
5.1	Aufbau . . . . .	25
5.2	Validierung . . . . .	26
5.3	Erwartungen . . . . .	30
5.4	Resultate . . . . .	30
<b>6</b>	<b>Schluss</b>	<b>31</b>
	<b>Literatur</b>	<b>32</b>
<b>7</b>	<b>Anhang</b>	<b>33</b>
7.1	Testkonzept . . . . .	33

# 1 Einleitung

Der Einsatz von Schaltnetzteilen kann dazu führen, dass Störungen ins Netz eingespiesen werden. Diese Störungen beeinträchtigen andere mit dem Netz verbundenen Geräte. Um diesen Störungen entgegenzuwirken, werden EMI-Filter eingebaut. Umgekehrt schützen sie die Geräte vor Störungen, die vom Netz zurückfliessen. EMI-Filter werden anhand der Einfügungsdämpfung charakterisiert, welche den Transmissionsgrad des einkommenden Signals beschreibt. Um einen EMI-Filter zu dimensionieren, muss sichergestellt werden, dass die entsprechenden Normen eingehalten werden. Dafür muss der Einfluss von den Komponenten des EMI-Filters auf die Einfügungsdämpfung bekannt sein.

Damit ein EMI-Filter dimensioniert werden kann, soll eine Simulationssoftware das Verhalten grafisch darstellen. Dies ist das Ziel/ Aufgabe im pro2E, FHNW. Um die Einfügungsdämpfung zu ermitteln, soll der EMI-Filter bezüglich vorgegebener Gleich- und Gegentaktschaltung untersucht werden. Dies geschieht anhand zweier Funktionen für Gleich- und Gegentaktschaltung. Die Funktionen zeigen die Einfügungsdämpfung für einen Frequenzbereich bis 30MHz. Die beiden Schaltungen beinhalten zudem die parasitären Parameter der elektrischen Komponenten, sodass eine realitätsnahe Simulation möglich ist. Des Weiteren soll die entwickelte Software die Einfügungsdämpfung grafisch darstellen. Die Resultate werden für die Gleich- und Gegentaktschaltung in separaten Funktionen dargestellt. Es soll zudem die Möglichkeit bestehen die elektrischen Komponenten der Schaltungen in der Simulationssoftware zu variieren.

Das Entwickeln der Simulationssoftware wird in zwei wesentliche Schritte unterteilt. Der erste Schritt besteht darin, die gegebenen Schaltungen zu vereinfachen, sodass die Berechnungen der Einfügungsdämpfung in der Software implementiert werden können. Im zweiten deutlich umfangreicheren Teil wird die Software programmiert, welche auf dem Prinzip MVC(Model-View-Control) basiert. Es werden Eingabemöglichkeiten implementiert, die es erlauben Komponenten, wie Induktivitäten, Kapazität und Widerstände des EMI-Filter, zu variieren. Zur Visualisierung dienen Funktionen, die die Einfügungsdämpfung in einem weiten Frequenzbereich darstellen.

Der Fokus des Fachberichts liegt auf die zu entwickelte Simulationssoftware. In einem ersten Schritt werden die theoretischen Grundlagen aufgeführt, die zum Entwickeln der Software benötigt werden. Im nächsten Kapitel wird erläutert, wie die zu simulierenden Schaltungen vereinfacht sind, sodass sie berechnet werden können. Außerdem wird darauf eingegangen, wie die Berechnungen in der Software implementiert sind. Im darauffolgenden Teil werden die Komponenten der Benutzeroberfläche aufgeführt und im Detail beschrieben.

## 2 Grundlagen

In diesem Kapitel werden die technischen Grundlagen beschrieben, welche für die Kapitel 3 und Kapitel 4 benötigt werden.

### 2.1 Elektrotechnik

#### 2.1.1 Passive 2-Tore

Folgendes Kapitel basiert auf der Quelle [1] Passive 2-Tore sind Netzwerke mit 2 Klemmenpaaren, die in jedem Betriebszustand Leistung verbrauchen. Ein Klemmenpaar wird auch als Tor bezeichnet. Dabei wird ein Tor als Eingang für ein elektrisches Signal verwendet. Folglich wird am anderen Tor das Ausgangssignal abgegriffen. Ein Netzwerk, das intern nur aus beliebigen R-(ohmscher Widerstand), L-(Induktivität), C-(Kapazität) und M-(Gegeninduktivität) Komponenten aufgebaut ist, ist immer ein passives Netzwerk. Ein solches 2-Tor ist in der Abbildung 2.1 dargestellt.

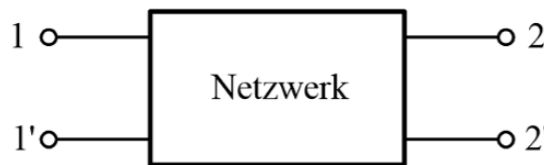


Abbildung 2.1: Zweitor (2-Tor) [1]

Ein solches Netzwerk ist zusätzlich noch reziprok. Reziproke Netzwerke haben die Eigenschaft, dass es egal ist, in welche Richtung sie betrieben werden, solange der Innenwiderstand der Quelle gleich gross wie die Lastimpedanz ist. Anhand vom folgenden Beispiel (Abbildung: 2.2) wird dies veranschaulicht. Die Leistung im Verbraucher ist in beiden Betriebszuständen dieselbe. Mit dieser Eigenschaft können in der Praxis die Berechnungen sehr vereinfacht werden.

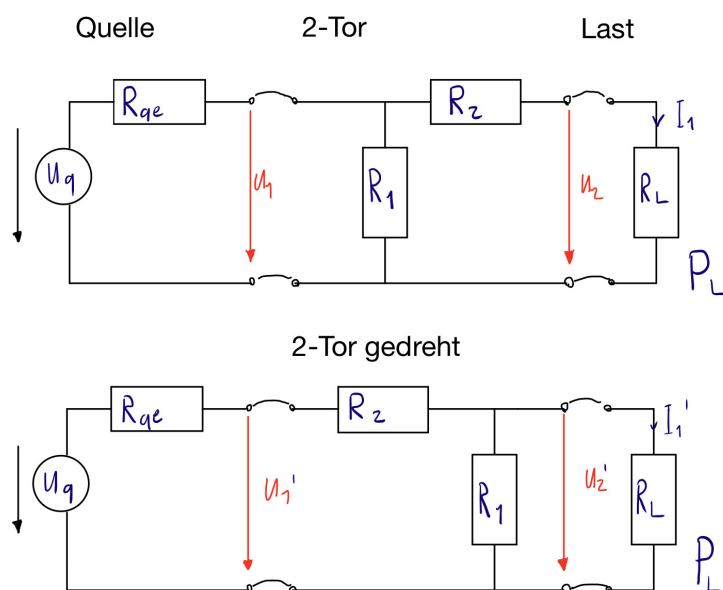


Abbildung 2.2: Beispiel Reziprozität

Als Beispiel sind  $R_{qe}$  und  $R_L$  jeweils 50 Ohm,  $R_1$  150 Ohm und  $R_2$  20 Ohm. Die Quelle hat eine Spannung von 100V.

In beiden Netzwerken werden jeweils aus der Quelle und dem 2-Tor der Ersatzwiderstand  $R_{q1}$  und  $R_{q2}$  berechnet.

$$R_{q1} = \frac{1}{\frac{1}{R_Q + R_2} + \frac{1}{R_1}} = 47.73\Omega \quad (2.1)$$

$$R_{q2} = \frac{1}{\frac{1}{R_Q} + \frac{1}{R_1}} + R_2 = 57.5\Omega \quad (2.2)$$

Aus dem Ersatzwiderstand berechnen wir den Gesamtstrom in beiden Schaltungen.

$$I_{tot1} = \frac{U_q}{R_{q1}} = 1.02A \quad (2.3)$$

$$I_{tot2} = \frac{U_q}{R_{q2}} = 0.93A \quad (2.4)$$

Die Spannungen  $U_1$  und  $U'_1$  sind unterschiedlich und werden per Spannungsteiler bestimmt.

$$U_1 = \frac{U_q \cdot R_{q1}}{R_{q1} + R_{qe}} = 48.84V \quad (2.5)$$

$$U'_1 = \frac{U_q \cdot R_{q2}}{R_{q2} + R_{qe}} = 53.48V \quad (2.6)$$

Auf der anderen Seite des Netzwerks hingegen bleiben die Spannungen gleich wie man an den Berechnungen sieht.

$$U_2 = \frac{U_1 \cdot R_L}{R_L + R_2} = 34.88V \quad (2.7)$$

$$U'_2 = \frac{U_2 \cdot \frac{1}{\frac{1}{R_L} + \frac{1}{R_1}}}{\frac{1}{\frac{1}{R_L} + \frac{1}{R_1}} + R_2} = 34.88V \quad (2.8)$$

Beim Strom verhält es sich durch den Stromteiler gleich

$$I_1 = \frac{I_{tot1} \cdot \frac{1}{\frac{1}{R_L + R_2} + \frac{1}{R_1}}}{R_2 + R_L} = 0.69A \quad (2.9)$$

$$I'_1 = \frac{I_{tot2} \cdot \frac{1}{\frac{1}{R_L} + \frac{1}{R_1}}}{R_L} = 0.69A \quad (2.10)$$

Dadurch ist auch die Leistung  $P_L$  in beiden Fällen gleichgross

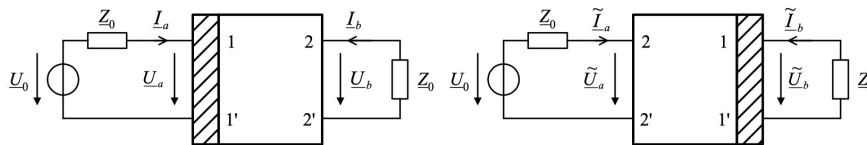
$$P_L = U_2 \cdot I_1 = 23.3W \quad (2.11)$$

Im dargestellten Netzwerk 2.3 gilt somit

$$\underline{I}_b = \tilde{\underline{I}}_b \quad (2.12)$$

sowie

$$\underline{U}_b = \tilde{\underline{U}}_b \quad (2.13)$$

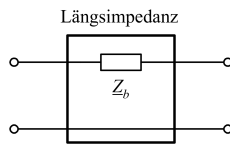


**Abbildung 2.3:** Reziprozität [1]

### 2.1.2 Kettenmatrix

Folgendes Kapitel erklärt die Grundlagen der Kettenmatrix. Die Grundlagen basieren auf den Quellen [2][3]. Die Kettenmatrix ist eine Variante, um das Verhalten von 2-Toren zu beschreiben. Andere Varianten sind die Z-Matrix oder die Y-Matrix. Die Kettenmatrix hat jedoch den Vorteil, dass man in Serie geschaltene 2-Tore ohne grossen Aufwand zusammen rechnen kann. Sobald man die einzelnen Kettenmatrizen gebildet hat und die Schaltung soweit vereinfacht ist, dass nur noch in Serie geschaltene Ketten-Matrizen vorzufinden sind, können diese miteinander multipliziert werden. Das Matrix-Produkt stellt dann die Kettenmatrix der Gesamtschaltung dar. Folgende gängige Schaltungen helfen, die Kettenmatrizen der einzelnen Schaltungsteilen zu bilden (siehe Anhang: 2-Tor Tabellen).

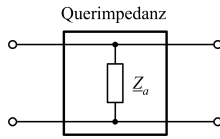
Die Längsimpedanz lässt sich anhand der Kettenmatrix  $A_L$  (Formel 2.14) darstellen



$$A_L = \begin{bmatrix} 1 & Z_b \\ 0 & 1 \end{bmatrix} \quad (2.14)$$

Abbildung 2.4: Längsimpedanz 7

Die Querimpedanz lässt sich anhand der Kettenmatrix  $A_Q$  (Formel 2.15) darstellen



$$A_Q = \begin{bmatrix} 1 & 0 \\ \frac{1}{Z_a} & 1 \end{bmatrix} \quad (2.15)$$

Abbildung 2.5: Querimpedanz 7

Sobald die Kettenmatrix einer Schaltung gebildet wurde, kann diese direkt in die Streuparameter umgewandelt werden. Der  $s_{21}$  Parameter kann wie in Formel 2.16 beschrieben, durch Einsetzen der Kettenmatrix bestimmt werden. Für den Widerstand  $R_w$  muss die verwendete Bezugsimpedanz eingesetzt werden.

$$s_{21} = \frac{2}{A_{11} + \frac{A_{12}}{R_w} + A_{21} * R_w + A_{22}} \quad (2.16)$$

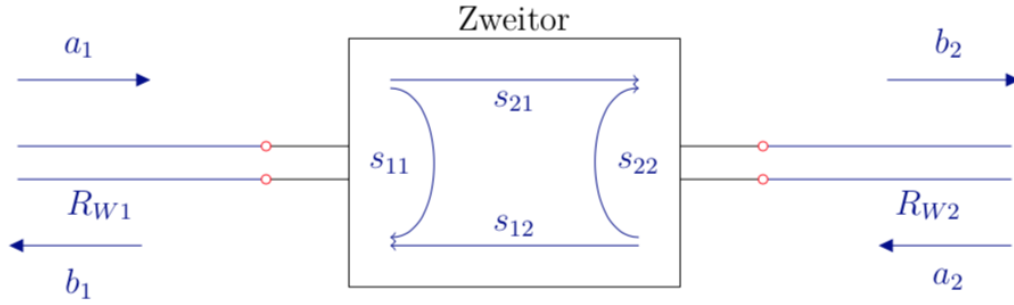
Die Indexierung der Kettenmatrix wird in der Formel 2.17 gezeigt.

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \quad (2.17)$$



### 2.1.3 Streuparameter

Dieses Kapitel ist eine Kurzeinführung in die Thematik der Streuparameter. Als Grundlage dazu dienen die Quellen [2][3]. Streuparameter (S-Parameter) werden in der Hochfrequenztechnik verwendet, um das Verhalten von n-Toren zu beschreiben. Bei einem 2-Tor sind vier Streuparameter nötig, um das Verhalten zu beschreiben. Sie beschreiben die Transmission von Tor 1 zu Tor 2, sowie von Tor 2 zu Tor 1. Des weiteren zeigen sie die Reflexion an den Toren auf. Die Abbildung 2.6 zeigt die Streuparameter an einem 2-Tor.



**Abbildung 2.6:** 2-Tor Wellengrößen und Anschlussleitungen [2]

Bei den S-Parametern werden die Eingangs- und Ausgangsgrößen nicht direkt anhand elektrischer Ströme und Spannungen beschrieben, sondern mithilfe von Wellengrößen, wobei  $a_i$  die einlaufenden Wellen sind und  $b_i$  die reflektierenden Wellen. Der Index  $i$  stellt den Torindex dar. Formel 2.18 und 2.19 zeigen wie die Wellengrößen  $a_i$  sowie  $b_i$  definiert sind. Die Quadrate der Beträge der Wellenstärken  $a$  und  $b$  entsprechen gerade den Leistungen, die mit diesen Wellen übertragen werden.

$$a_i = \sqrt{P_{vor}}, a = \text{Wellenstärke der vorlaufenden Welle} \quad (2.18)$$

$$b_i = \sqrt{P_{rück}}, b = \text{Wellenstärke der rücklaufenden Welle} \quad (2.19)$$

Aus der Abbildung 2.3 lässt sich folgende Streumatrix darstellen (Formel 2.20):

$$\begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \quad (2.20)$$

Die Elemente der S-Matrix sind:

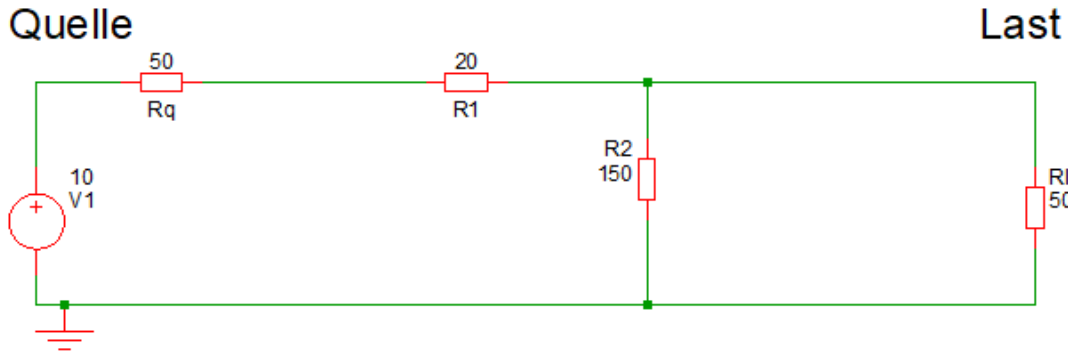
$$s_{11} = b_1/a_1 \text{ Eingangsreflexionsfaktor bei angepasstem Ausgang } (a_2=0) \quad (2.21)$$

$$s_{12} = b_1/a_2 \text{ Rückwärtstransmissionsfaktor bei angepasstem Eingang } (a_1=0) \quad (2.22)$$

$$s_{21} = b_2/a_1 \text{ Vorwärtstransmissionsfaktor bei angepasstem Ausgang } (a_2=0) \quad (2.23)$$

$$s_{22} = b_2/a_2 \text{ Ausgangsreflexionsfaktor bei angepasstem Eingang } (a_1=0) \quad (2.24)$$

In der folgenden Schaltung (Abbildung 2.7) wird Schritt für Schritt erklärt, wie der  $s_{21}$ -Parameter der Streumatrix berechnet wird.



**Abbildung 2.7:** Beispielschaltung Streuparameter

Der Streuparameter  $s_{21}$  ist, wie in Formel 2.16 definiert. Somit muss im ersten Schritt die Kettenmatrix der Gesamtschaltung ermittelt werden. Die Kettenmatrix bezieht sich auf die beiden Widerstände  $R_1$  und  $R_2$ . Der Widerstand  $R_1$  stellt eine Längsimpedanz dar, welche somit in die passende Matrix eingesetzt wird (Formel 2.14).  $A_1$  (Formel 2.25) stellt die Kettenmatrix der Längsimpedanz  $R_1$  dar.

$$A_1 = \begin{bmatrix} 1 & R_1 \\ 0 & 1 \end{bmatrix} \quad (2.25)$$

Der Widerstand  $R_2$  stellt eine Querimpedanz dar, welche in die Formel 2.15 eingesetzt wird.  $A_2$  (Formel 2.26) stellt die Kettenmatrix der Querimpedanz  $R_2$  dar.

$$A_2 = \begin{bmatrix} 1 & 0 \\ \frac{1}{R_2} & 1 \end{bmatrix} \quad (2.26)$$

Durch Kaskadieren der beiden Kettenmatrizen wird die Kettenmatrix der Gesamtschaltung gebildet. Kaskadieren bedeutet das Bilden des Matrixproduktes.

$$A_{tot} = A_1 \cdot A_2 = \begin{bmatrix} 1 & R_1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ \frac{1}{R_2} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 20 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ \frac{1}{150} & 1 \end{bmatrix} = \begin{bmatrix} 1.1333 & 20.000 \\ 0.0067 & 1.0000 \end{bmatrix} \quad (2.27)$$

Der nächste Schritt besteht darin, die Kettenmatrix in Bezug zu den Wellenimpedanzen darzustellen. In diesem Beispiel entspricht die Wellenimpedanz des Innenwiderstandes sowie des Lastwiderstandes  $50\Omega$ . Aus der Formel 2.28 wird diese durch Einsetzen gebildet.

$$A'_{tot} = \begin{bmatrix} A_{11} \cdot \sqrt{\frac{R_l}{R_q}} & \frac{A_{12}}{\sqrt{R_q \cdot R_l}} \\ A_{21} \cdot \sqrt{R_q \cdot R_l} & A_{22} \cdot \sqrt{\frac{R_q}{R_l}} \end{bmatrix} = \begin{bmatrix} 1.1333 \cdot \sqrt{\frac{50}{50}} & \frac{20.000}{\sqrt{50 \cdot 50}} \\ 0.0067 \cdot \sqrt{50 \cdot 50} & 1.0000 \cdot \sqrt{\frac{50}{50}} \end{bmatrix} = \begin{bmatrix} 1.1333 & 0.4000 \\ 0.3333 & 1.0000 \end{bmatrix} \quad (2.28)$$

Aus der Matrix  $A'_{tot}$  kann mithilfe der Formel 2.29 der Streuparameter  $s_{21}$  direkt ausgerechnet werden.

$$s_{21} = \frac{2}{A'_{11} + A'_{12} + A'_{21} + A'_{22}} = \frac{2}{1.1333 + 0.4000 + 0.3333 + 1.0000} = 0.6977 \quad (2.29)$$

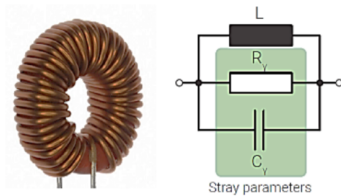
Der Streuparameter  $s_{21}$  entspricht dem Transmissionsfaktor der eingehenden Welle. Daraus ergibt sich die Formel 2.30. Dies dient der Überprüfung der vorigen Berechnungen. Die Resultate der beiden Berechnungen stimmen überein.

$$s_{21} = \sqrt{\frac{P_l}{P_{AV}}} = \sqrt{\frac{0.2434W}{0.5000W}} = 0.6977 \quad (2.30)$$

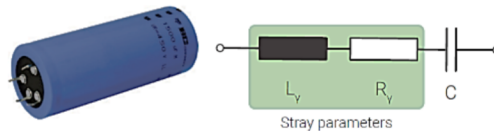
### 2.1.4 Parasitäre Parameter

In diesem Unterkapitel werden grundsätzlich die Einflüsse und Eigenschaften von Parasitären Parametern in Realen Bauteilen, besonders Spule und Kondensator, erklärt. Das Kapitel basiert auf Quelle [4].

Ideale Bauteile beschreiben eine Funktion. Da reale Bauteile aus Materialien mit physikalischen Eigenschaften bestehen, treten bei der Umsetzung dieser Funktion Nebeneffekte auf. Sie entstehen, weil die einzelnen Bauteile im Betrieb elektrische Felder oder Magnetfelder erzeugen. Oder einfach durch die Leitfähigkeit eines Materials. Diese physikalisch bedingten Effekte werden als parasitär bezeichnet. Sie treten als Widerstand, Induktivität oder Kapazität auf. Da sie gut klassifiziert werden können, werden sie als Parameter bezeichnet. Um eine Schaltung präzise zu simulieren ist es unerlässlich, die elektrischen Bauelemente mit den passenden parasitären Parametern zu ergänzen. In den Abbildungen 2.8 und 2.9 werden die parasitären Parameter von Spule und Kondensator gezeigt. Man stellt sie als zusätzliche Bauteile dar.



**Abbildung 2.8:** Parasitäre Elemente einer Induktivität [4]



**Abbildung 2.9:** Parasitäre Elemente einer Kapazität [4]

### 2.1.5 Einfügedämpfung

Das Kapitel basiert auf Quelle [2]. Die Einfügedämpfung (engl. Insertion loss) ist eine Grösse, die verwendet wird, um das Verhalten einer Schaltung zu beschreiben. Sie beschreibt das Verhältnis zwischen der eingehenden Leistung zur abgegebenen Leistung. Sie wird in Dezibel (dB) angegeben. Es handelt sich um eine logarithmische Grösse. Um die Einfügedämpfung in einem Bereich von bis 30MHz abzudecken, wird die Formel 2.31 verwendet. In der Formel 2.31 wird die Einfügedämpfung mittels Streuparameter (S-Parameter) berechnet. Sie wird in Dezibel (dB) angegeben. Der Streuparameter  $S_{21}$  beschreibt im wesentlichen den Transmissionsgrad des eingehenden Signals. Der Transmissionsgrad beschreibt, welchen Anteil des eingehenden Signals am Ausgang wieder herauskommt.

$$IL = -20 * \log(|S_{21}|) \quad (2.31)$$

### 2.1.6 Aufbau eines EMI-Filters

Ein EMI-Filter ist ein lineares Netzwerk aus R-, L-, C-Gliedern und einem Transformator. Somit besitzen sie eine reziproke Übertragungssymmetrie, was eine einfache Berechnung von verschiedenen Zusammenhängen erlaubt. In der Abbildung 2.10 ist die Schaltung eines EMI-Filters dargestellt.

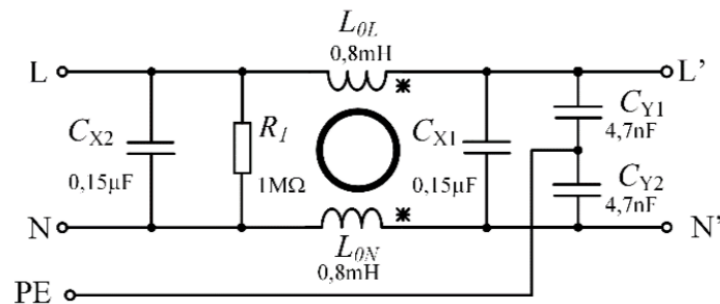


Abbildung 2.10: Original Schaltung [4]

### 2.1.7 Gleich- Gegentaktschaltung

Dieses Kapitel basiert auf der Quelle [1]. In der realen Stromverteilung wird beabsichtigt, dass der Stromfluss über einen Zuleiter zum Verbraucher hinein-, respektive über einen Ableiter herausgeführt wird. Diese Art der Signalübertragung wird als Gegentakt-Betrieb bezeichnet. Im realen Stromnetz ist allerdings auch der sogenannte Gleichtakt-Betrieb vorhanden. Dabei wirken alle Leiter als Zuleiter, der gesamte Strom wird durch die Erde weggeleitet. Durch das Gesetz der Superposition ist es möglich, den Gleich- und den Gegentaktanteil getrennt voneinander zu betrachten. Dieses Phänomen wird in der Abbildung 2.11 dargestellt.

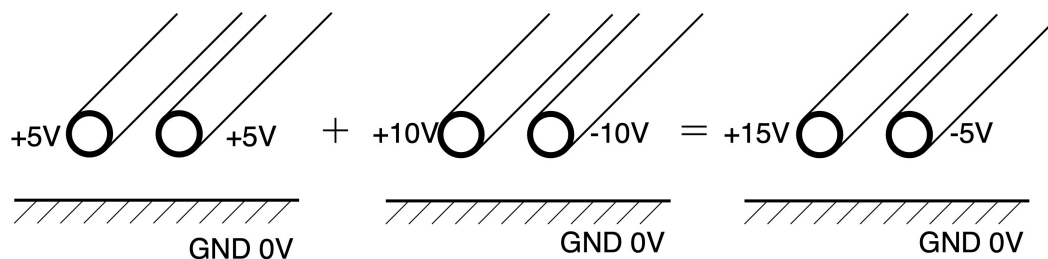


Abbildung 2.11: Beispiel aufgetrennten Leitung

An einen geerdeten Verbraucher sind 2 Phasen angeschlossen. An der Zuleitung liegt eine Spannung von 15 Volt an, an der Rückleitung liegen -5 Volt an. Diese Leitung wird nun aufgeteilt in eine Gleichtaktleitung, bei welcher über beide Phasen 5 Volt eingespeist werden und in eine Gegentaktleitung, in welcher durch die Zuleitung 10 Volt, respektive in der Rückleitung -10V eingespeist werden. Während in der Gleichtaktleitung die addierten 10 Volt gegenüber der Erde anliegen, werden sie in der Gegentaktleitung abgeführt.

## 2.2 Programmieren

### 2.2.1 MVC-Struktur

Das MVC-Framework wird zur Softwarestrukturierung verwendet. Durch diese Strukturierung werden die Berechnungen der Daten (eng. model), die Steuerung (engl. controller) und dessen graphischer Repräsentation (engl. view) getrennt. In der Abbildung 2.12 ist dieser Aufbau in einem Beispielklassendiagramm dargestellt. [5]

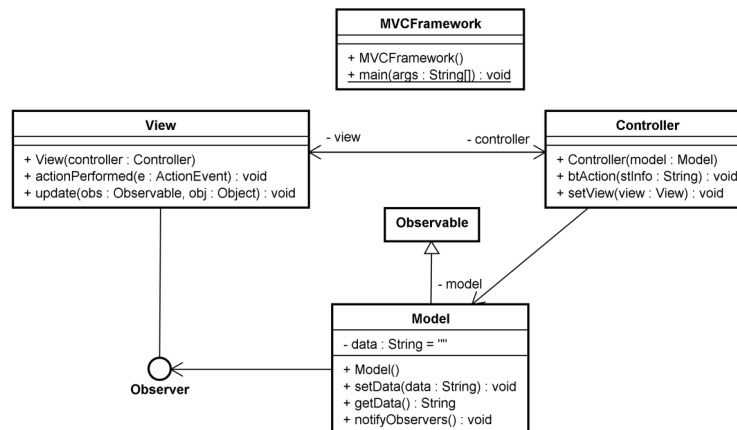


Abbildung 2.12: MVC Beispielklassendiagramm [6]

Der Ablauf dieser Struktur ist wie folgt:

1. Benutzereingabe löst Event aus
2. Die Aktion wird dem Controller übergeben. Dieser holt die Daten in der View, leitet diese dem Model weiter und löst die Berechnungen aus
3. Das Model führt die Berechnungen aus und informiert den Observer
4. Der Observer löst ein Event in der View aus. View kann die Daten vom Model holen und ausgeben

### 3 Ermitteln der Einfügungsdämpfung

In diesem Kapitel wird ausführlich beschrieben, wie aus den gegebenen Gleich- und Gegentaktschaltung (siehe Anhang Aufgabenstellung) die Einfügungsdämpfung ermittelt wird. Um die einzelnen Schritte konsistent zu beschreiben, wird an den entsprechenden Stellen Bezug das Kapitel 2 genommen. Der erste Abschnitt behandelt die Gleichtaktschaltung 3.1 und in einem zweiten Abschnitt wird die Gegentaktschaltung 3.2 behandelt. In diesen beiden Kapiteln wird beschrieben, wie aus den Schaltungen aus der Aufgabenstellung die Kettenmatrizen der beiden Schaltungen ermittelt werden. Aus der Kettenmatrix kann somit, wie in Kapitel 2.1.5 beschrieben, die Einfügungsdämpfung berechnet werden.

#### 3.1 Gleichtaktschaltung

Um die Einfügungsdämpfung zu ermitteln, wird im ersten Schritt die Schaltung weitgehend vereinfacht. Die reduzierte Schaltung wird anhand der Kettenmatrix (siehe 2.1.2) beschrieben. Aus der Kettenmatrix wird dann für den vorgegebenen Frequenzbereich die Einfügungsdämpfung berechnet.

#### Reduktion der Schaltung

Die Abbildung 3.1 zeigt die Schaltung aus der Aufgabenstellung.

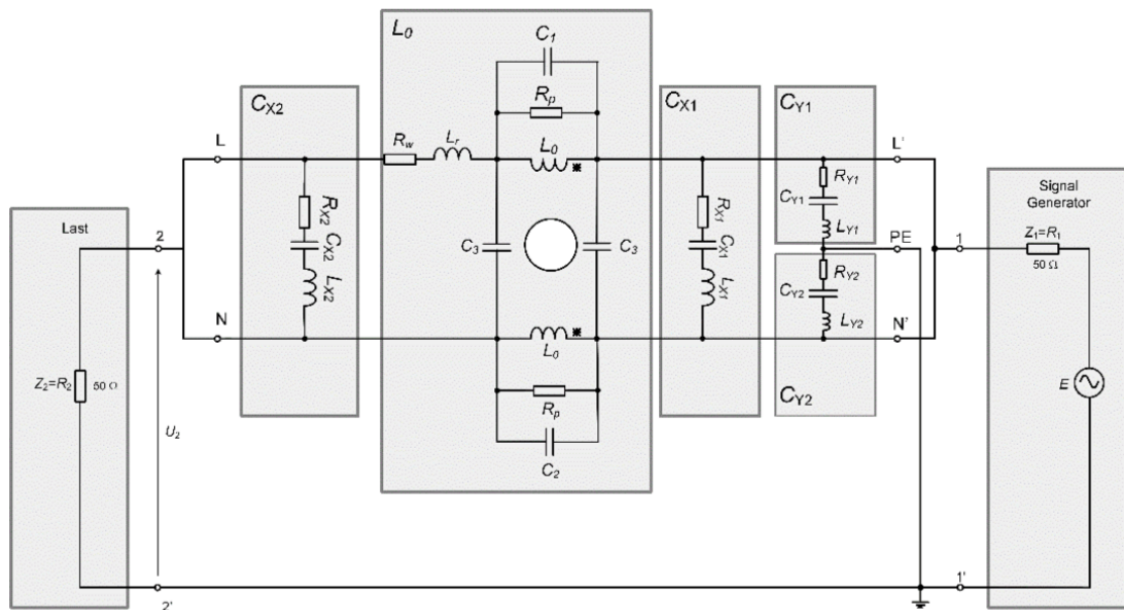
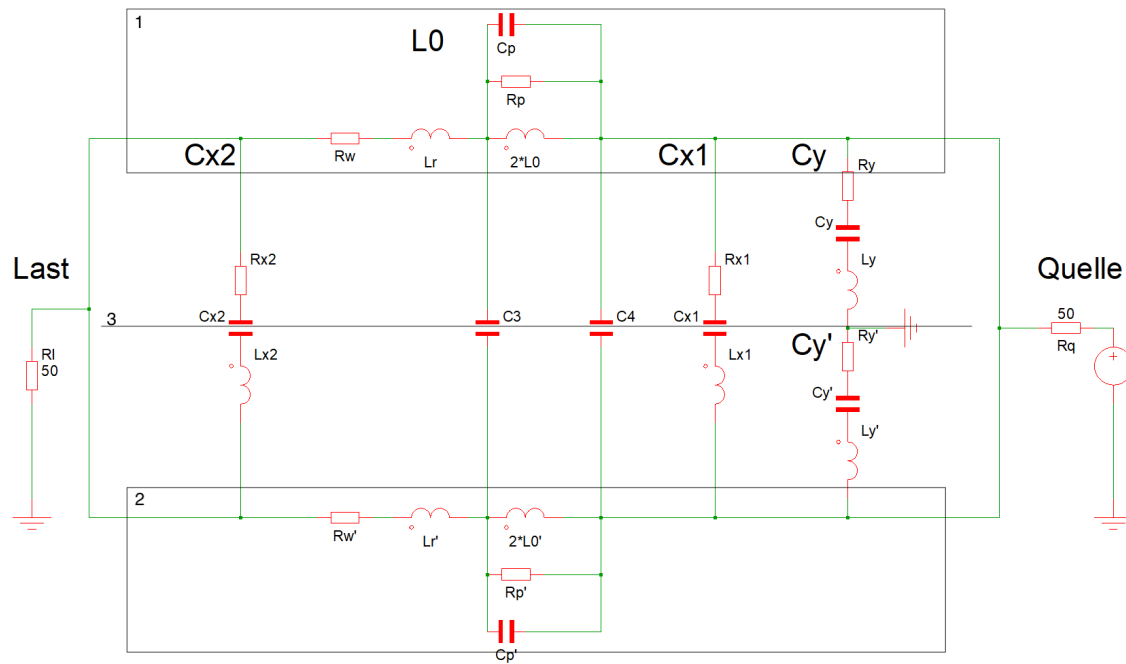


Abbildung 3.1: Originale Gleichtaktschaltung[4]

Die Originalschaltung wird mit den Komponenten  $R_w$  und  $L_r$  ergänzt, sodass sie symmetrisch ist. Dies macht es möglich, dass die Schaltung weiter vereinfacht werden kann. Zudem wurden  $C_1$  und  $C_2$  in  $C_p$  und  $C'_p$  umbenannt um zu zeigen, dass es sich um die gleiche Kapazität handelt. Durch Weglassen des Eisenkerns, wird  $L_0$  und  $L'_0$  mit dem Vorfaktor 2 ergänzt, da sich die Magnetfelder überlagern. Durch diese Änderungen ergibt sich die Schaltung in Abbildung 3.2.



**Abbildung 3.2:** Ergänzte Gleichtaktschaltung

Der obere Strang (siehe Abbildung 3.2, Nr. 1) und der untere Strang (siehe Abbildung 3.2, Nr. 2) sind identisch. Da es keinen Potentialunterschied zwischen ihnen gibt, kann die Schaltung entlang der Symmetrie-Achse (siehe Abbildung 3.2, Nr. 3) aufgetrennt werden. Dies ergibt die Schaltung in Abbildung 3.3.

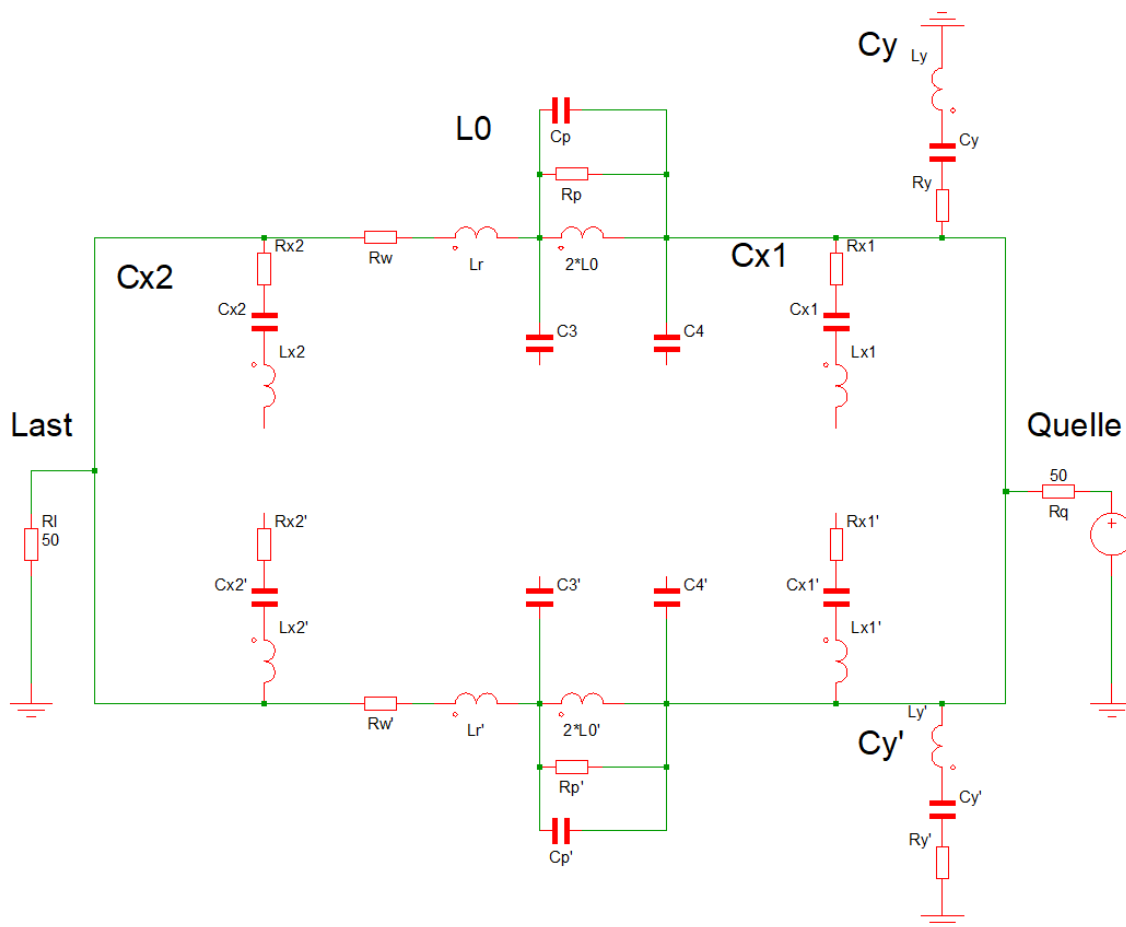


Abbildung 3.3: Aufgetrennte Gleichtaktschaltung

Komponenten die nicht mehr fest verbunden sind, fallen weg. Dies gilt für die Kondensatoren  $C_3$ ,  $C_4$ ,  $C_{x1}$  und  $C_{x2}$ . Die weiter reduzierte Schaltung wird in Abbildung 3.4 abgebildet.

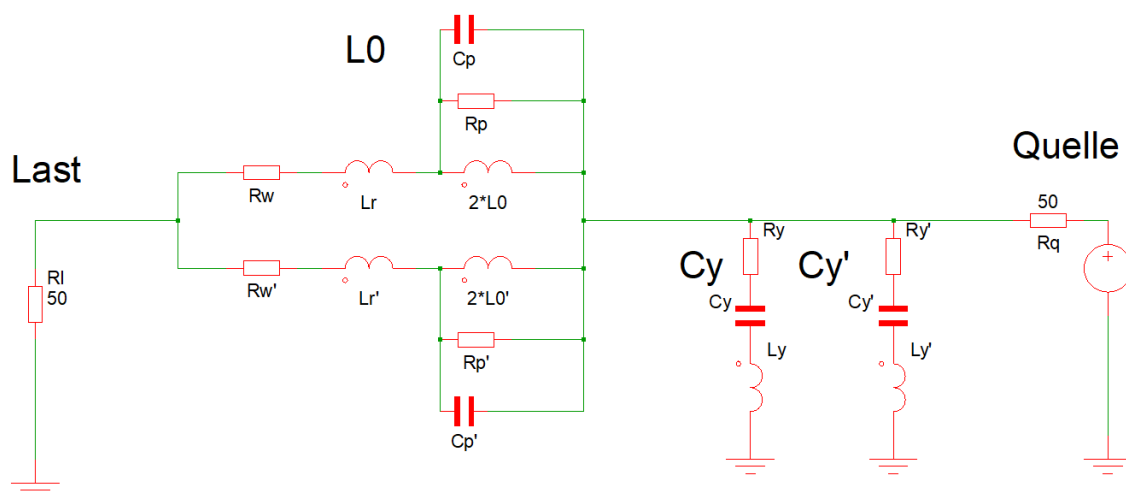


Abbildung 3.4: Vereinfachte Gleichtaktschaltung



Die übrigen Komponenten von  $L_0$  bilden eine Parallelschaltung, welche sich durch halbieren der Widerstände und Induktivitäten und verdoppeln der Kapazitäten zusammenfassen lässt. Zusätzlich werden die beiden  $C_y$  und  $C'_y$  parallel auf das Bezugspotential geschaltet. Da  $C_y$  und  $C'_y$  identisch sind, werden sie wie in Abbildung 3.5 zusammengefasst. Diese vereinfachte Schaltung bildet die Grundlage für die Berechnungen der Software.

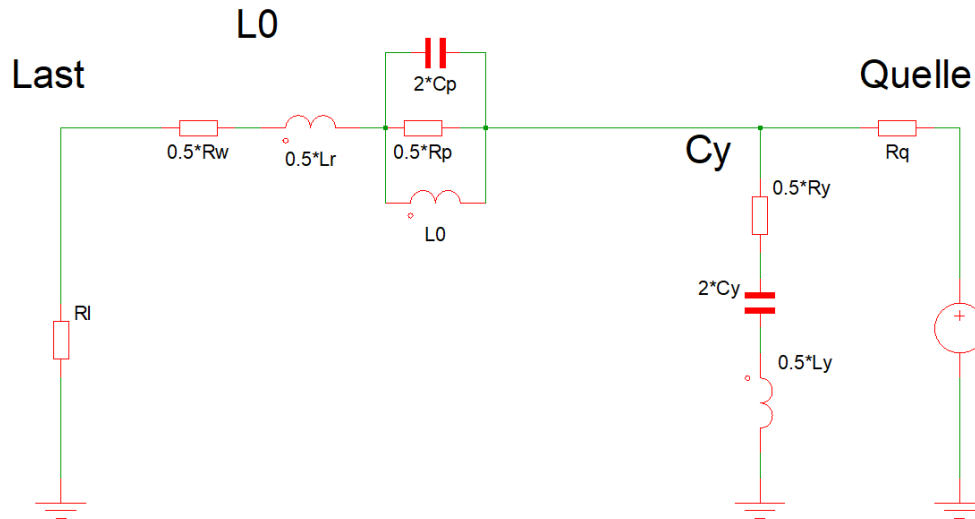


Abbildung 3.5: komplett reduzierte Gleichtaktschaltung

### Bilden der Kettenmatrix

Damit die Kettenmatrix der Gesamtschaltung gebildet werden kann, wird in einem ersten Schritt die reduzierte Schaltung in Längs- und Querimpedanzen (siehe Kapitel 2.1.2) eingeteilt. Diese werden in Abbildung 3.6 mit den Kennzeichnungen „QI“ und „LI“ vermerkt, wobei „QI“ für Querimpedanz steht und „LI“ für Längsimpedanz. Die Impedanzen der einzelnen Schaltungsteile werden in die Kettenmatrizen für Quer- und Längsimpedanz eingesetzt. Die Kettenmatrizen werden durch Kaskadierung zur Kettenmatrix der Gesamtschaltung zusammengefasst.

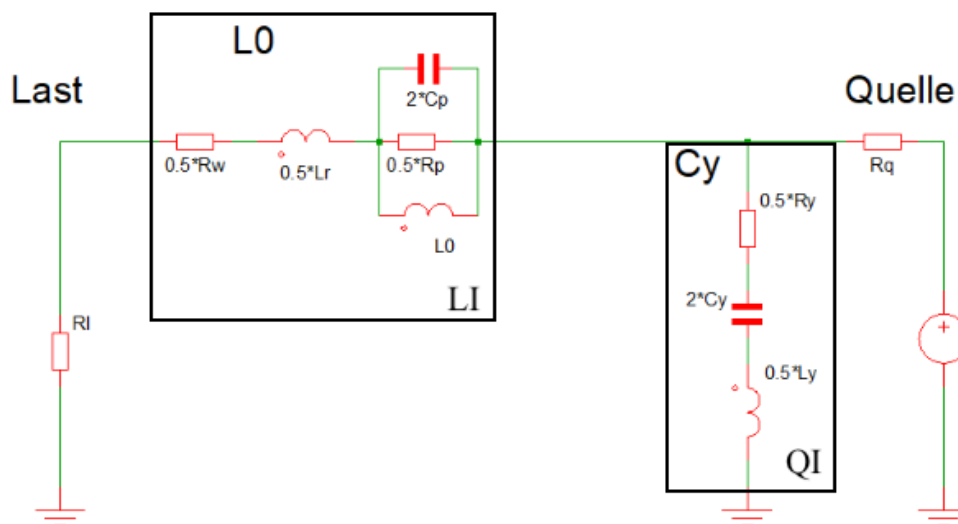


Abbildung 3.6: Einteilung der Gleichtaktschaltungsteile

Anhand der Kettenmatrix der Gesamtschaltung kann die Einfügungsdämpfung direkt ermittelt werden, gemäss Kapitel 2.1.3.

## 3.2 Gegentaktschaltung

### Reduktion der Schaltung

Folgender Abschnitt legt Schritt für Schritt dar, wie die Gegentaktschaltung vereinfacht wird. Die Abbildung 3.7 zeigt die Originalschaltung, wie sie der Aufgabenstellung entnommen wurde. In einem ersten Schritt wird die gekoppelte Spule (im Bild  $L_0$ ) vereinfacht. Die Magnetfelder der beiden Induktivitäten  $L_0$  kompensieren sich gegenseitig. Sie fallen weg, da sie niederohmig werden. Somit verschwindet  $L_0$  mit den parasitären Elementen  $C_1$ ,  $C_2$  und die beiden Widerstände  $R_p$ .

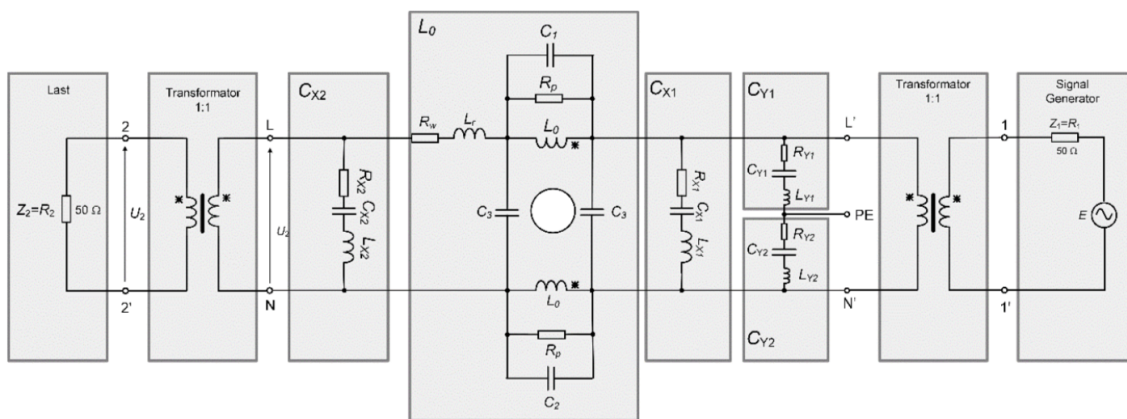


Abbildung 3.7: Originale Gegentaktschaltung



Der obere und untere Teil der aufgetrennten Schaltung ist weitgehend identisch. Somit reduziert sich die Schaltung auf einen der beiden Stränge. Abbildung 3.10 zeigt die komplett vereinfachte Schaltung, welche die Grundlagen für die Gegentaktberechnungen bildet.

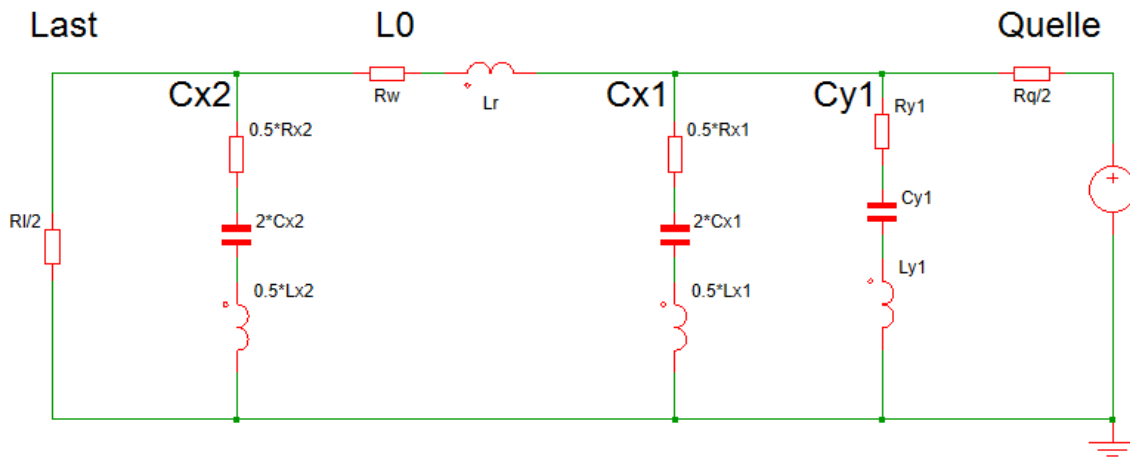


Abbildung 3.10: komplett vereinfachte Gegentaktschaltung

### Bilden der Kettenmatrix

Abbildung 3.11 zeigt die Einteilung der reduzierten Schaltung in Längs- und Querimpedanzen.

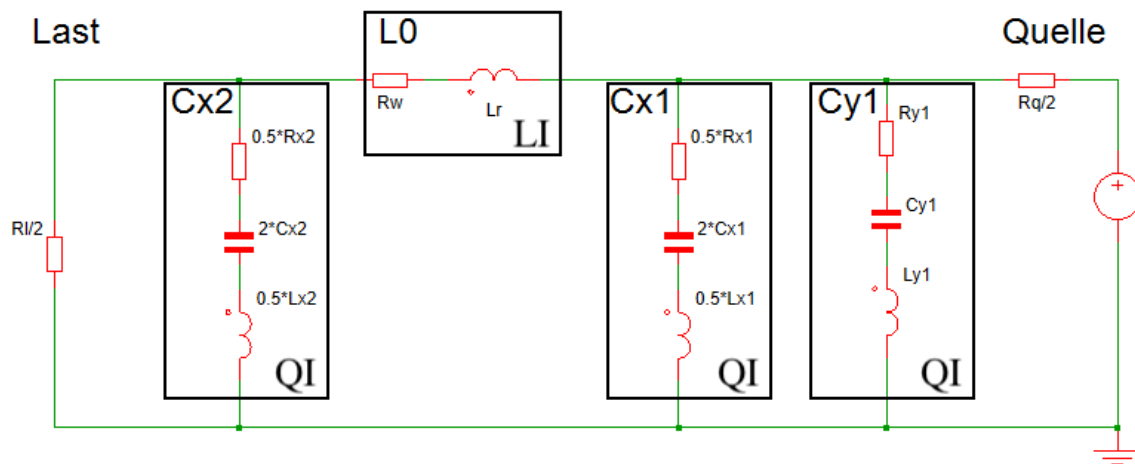


Abbildung 3.11: Einteilung der Gegentaktschaltungsteile

Im nächsten Schritt werden die Impedanzen der einzelnen Schaltungsteile gebildet, welche in die passenden Kettenmatrizen eingesetzt werden. Wiederum durch Kaskadierung der einzelnen Kettenmatrizen wird die Kettenmatrix der Gesamtschaltung gebildet. Aus der Kettenmatrix wird wie bei der Gleichtaktschaltung die Einfügungsdämpfung berechnet.

## 4 Software

In diesem Kapitel wird die Umsetzung der Berechnung (Kapitel 3) und der anderen Anforderungen an die Software in Java dokumentiert. Die programmietechnischen Grundlagen sind in Kapitel 2 beschrieben.

### 4.1 Übersicht

Das Programm simuliert die Einfügedämpfung eines EMI-Filters. Solche Filter werden häufig in Schaltnetzteile verbaut, um zu verhindern, dass Störungen zurück ins Netz gespeist werden. Die Benutzeroberfläche des Programms ist in der Abbildung 4.1 ersichtlich.

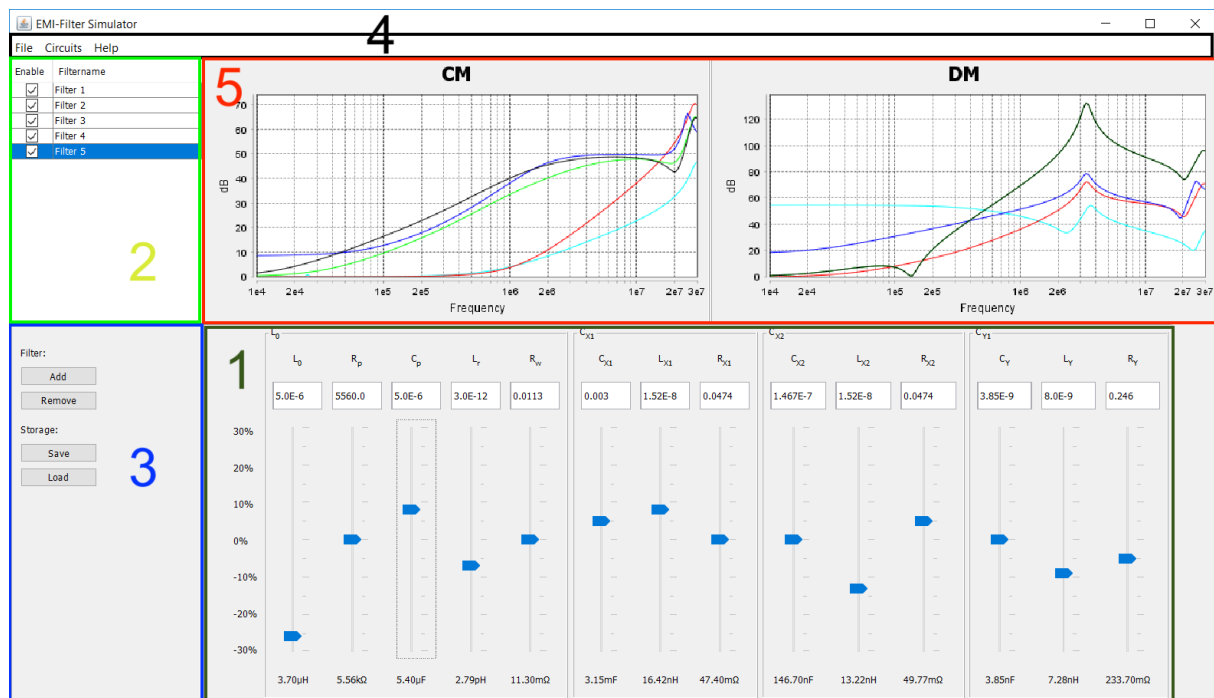


Abbildung 4.1: GUI

Die Software ist nach dem Model-View/Controller Entwurfsmuster [5] aufgebaut. Das Klassendiagramm der Software befindet sich im Anhang 7). Die View ist in fünf Panels unterteilt. Das InputPanel (1) beinhaltet für jeden Filterparameter ein Textfeld, einen Slider und ein Anzeigelauf, welche die Parameterwerte definieren. Das FiltertablePanel (2) verwaltet die verschiedene Filter in einer Tabelle, in der auch die Parameterwerte hinterlegt werden. Das ButtonPanel (3) besitzt vier Buttons, die für die Speicher- und Filterverwaltung zuständig sind. Die Menubar (4) besitzt verschiedene Menüs für die Bedienung des Programms. Das PlotPanel (5) dient zur grafischen Darstellung der Einfügeverluste des Filters. Der Controller leitet Informationen von der View zu dem Model weiter. Das Model besitzt Klassen, um mit den Komponenten der modellierten realen Bauteile zu rechnen. Via eines Observer wird ein Update in der View ausgelöst.

## 4.2 View

Die View ist als GridBagLayout organisiert und enthält die im Kapitel 4.1 erwähnten Panels.

### InputPanel

Das InputPanel hat den Controller und ist im GridBagLayout organisiert. Das Panel wird in mehreren Subpanels unterteilt. Diese sind in der Abbildung 4.2 abgebildet.

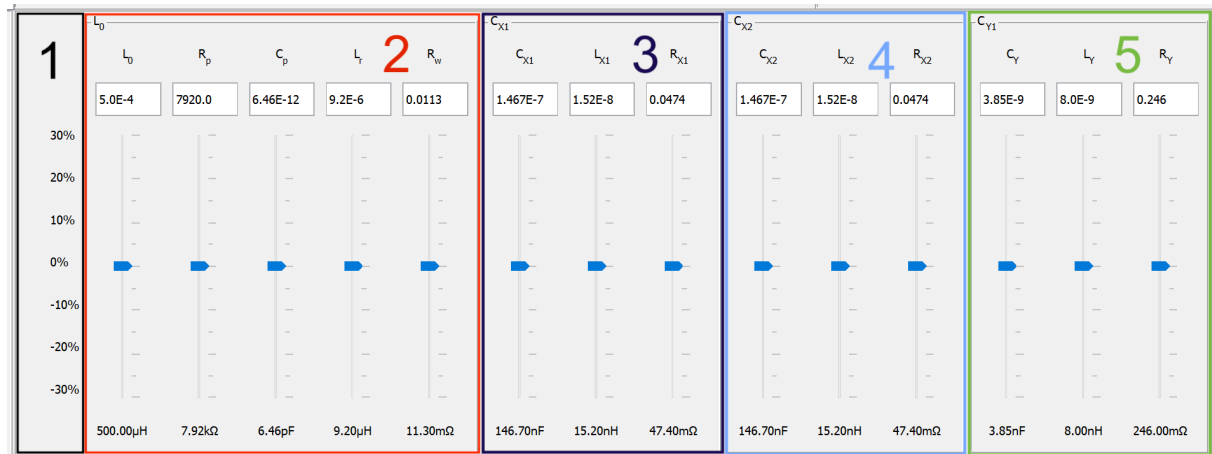


Abbildung 4.2: InputSubPanels

Das InformationPanel(1) ist im Null-Layout organisiert. Es werden die Prozentzahlen der Schieberegler dargestellt. Diese sind an einer festen Position, da die Grösse des InputPanels nicht verändert werden kann. Die weiteren Subpanels (2-5) sind im GridBagLayout organisiert. In diesen Panels befinden sich die Komponenten der modellierten realen Bauteile. Jede Komponente besitzt ein Textfeld, einen Slider und ein Label. Im Textfeld kann der Benutzer seine Werte für die Komponenten eintragen. Beim Aufstarten des Programmes wird das Feld mit einem Standardwert, der vom Auftragsdokument übernommen wurde, geladen. Mit der Klasse JEngineerField werden die Eingaben geprüft. Es ist nur möglich Zahlen einzugeben. Grosse und kleine Zahlen können zur Vereinfachung in wissenschaftlicher Schreibweise (18e-12) oder in Einheiten-Schreibweise (18p) eingetragen werden. Die Ausgabe ist als Einheiten-Schreibweise vordefiniert. Mit dem Schieberegler wird der im Textfeld eingegebene Wert um  $\pm 30\%$  verändert und im Label als effektiver Wert ausgegeben. Das Textfeld besitzt einen ActionListener und der Slider einen ChangeListener. Bei einem Ereignis wird die dazugehörige Methode im Controller aufgerufen.

### FiltertablePanel

Das FiltertablePanel hat den Controller und ist im GridBagLayout organisiert. Es enthält eine Tabelle, in der die Parameterwerte hinterlegt und zu dem entsprechenden Filter zugeordnet werden. Jeder Filter in der Tabelle besitzt eine Checkbox, um den Filter zu aktivieren und ein Textfeld, um den Filter zu benennen. Die Tabelle hat einen TableModelListener und einen ListSelectionListener. Bei einem Ereignis wird die dazugehörige Methode im Controller aufgerufen.

## ButtonPanel

Das ButtonPanel hat den Controller und ist im Null-Layout organisiert. Es enthält die vier Buttons und zwei dazugehörige Labels. Mit diesen kann die Filtertabelle verwaltet und die Speicherverwaltung aufgerufen werden. Jeder Button hat einen ActionListener. Bei einem Ereignis wird die dazugehörige Methode im Controller aufgerufen.

## Plotpanel

Das PlotPanel ist im BorderLayout organisiert. Mit Hilfe des Package JfreeChart [7] wird die Einfügedämpfung des EMI-Filters in einem Plot dargestellt. Das Package übernimmt viele Funktionen wie das Zoomen des Plots oder das Ändern der Darstellung. Diese Möglichkeiten sind in der Abbildung 4.3 dargestellt. Löst der Observer ein Ereignis aus, werden die Daten vom Model geholt und geplottet.

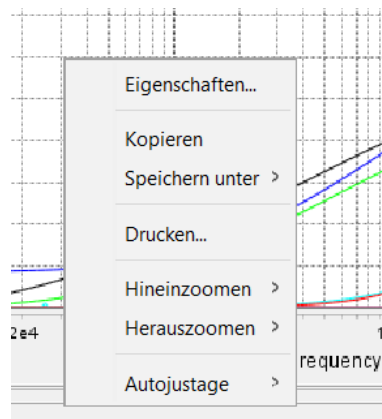
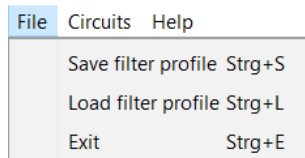


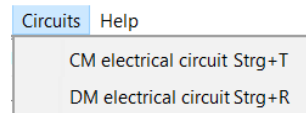
Abbildung 4.3: Einstellungen

## Menubar

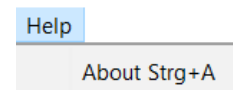
Das Menubar hat den Controller und ist im Null-Layout organisiert. Es besitzt die Menus File (Abbildung 4.4), in dem die Speicherverwaltung aufgerufen werden kann, Circuits (Abbildung 4.5), in dem die Schaltungen, von denen die Berechnungen ausgehen angezeigt werden und Help (Abbildung 4.6), in dem Informationen über das Programm hinterlegt sind. Die Menüs haben einen ActionListener. Bei einem Ereignis wird die dazugehörige Methode im Controller aufgerufen.



**Abbildung 4.4:** Menüpunkt File



**Abbildung 4.5:** Ersatzschaltbilder



**Abbildung 4.6:** Menüpunkt Help



### 4.3 Controller

Der Controller leitet die entsprechenden Aufgaben an das Model weiter. Es umfasst neben dem Konstruktor 14 Methoden, die entweder Daten direkt an das Model weiterleiten, um die Berechnungen durchführen zu können, oder Methoden wie `getEffectiveParameterValues()`, die als Schnittstelle dienen, um Daten vom Model in den `StorageManager` zu verschieben, damit diese abgespeichert werden können.

### 4.4 Model

Die Hauptaufgabe des Models besteht darin, die Einfügungsdämpfung zu berechnen. Die Methode `calculate()` berechnet anhand der übergebenen Komponenten der Schaltung die Einfügungsdämpfung. Die Struktur des Models ist dem Klassendiagramm im Anhang zu entnehmen. Logarithmisch verteilt werden 400 Werte in einem Bereich von 0Hz bis 30MHz für die beiden Schaltungen (siehe Kapitel 3) berechnet. Die Berechnungen werden separat für Gegen- und Gleichtakt durchgeführt.

Dafür werden, wie in Kapitel 2.1.2 beschrieben, die einzelnen Schaltungsteile anhand von Kettenmatrizen beschrieben. Da die Kettenmatrix die Impedanz der einzelnen Schaltungsteilen beinhaltet, gibt es Klassen für Spule, Kondensator und Widerstand (Induktor, Capacity, Resistor). Dem Konstruktor dieser Klassen wird der Hauptwert (beim Kondensator z.B. die Kapazität), sowie die Parasitären Parameter (beim Kondensator der Serie Widerstand und die Serie Kapazität) übergeben. Diese Objekte stellen die Methode `.getImpedance(frequency)` zu Verfügung. Sie gibt für eine Frequenz die Impedanz zurück.

Die Klasse `Mikromatlab` dient als Bibliothek für die Berechnungen. Sie beinhaltet die Methoden `getSeriesImpedanceMatrix` (`SeriesImpedance`) und `getShuntImpedanceMatrix` (`ShuntImpedance`). Diese Methoden geben für eine Impedanz (Quer oder Längs) die entsprechende Kettenmatrix zurück (Kapitel 2.1.2).

Die Methode `cascade(1. Matrix, 2. Matrix)` ist eine weitere Methode der `Mikromatlab` Klasse. Sie ermöglicht die Kaskadierung der einzelnen Schaltungsteile (Kettenmatrizen). Sie führt eine Matrix Multiplikation der übergebenen Matrizen durch. Die einzelnen Schaltungsteile werden miteinander multipliziert. Das Produkt stellt die Gesamtkettenmatrix der Schaltung dar.

Die Formel 2.16 legt dar, wie aus der Gesamtkettenmatrix der Streuparameter `s21` berechnet wird. In der Methode `calculate` wird die beschriebene Berechnung in einer for-Schleife für die logarithmisch eingeteilte Frequenz-Achse durchgeführt. Die logarithmische Einteilung wird anhand der Methode `logspace()` der Klasse `Mikromatlab` erstellt. Die berechneten `s21`-Parameter werden logarithmisch umgerechnet und in den Attributen `cmData` und `dmData` abgespeichert. Diese beiden Attribute sind drei dimensionale Arrays, wobei die erste Dimension die Filter der Filtertabelle nummeriert. Die zweite Dimension spezifiziert die x und y-Achse. Die dritte Dimension beinhaltet die errechneten Werte.

Sobald die Berechnungen gemacht sind, wird anhand der Methode `notifyObserver()` bei der View ein update ausgelöst. Die Methoden `getCM` und `getDM` des Models ermöglichen der View, die berechneten Daten zu holen.

Wie im Kapitel 4.2 beschrieben, ist es möglich die Filter anhand der Checkboxes sichtbar und unsichtbar zu stellen. Um diese Funktion zu gewährleisten, wird der Methode `calculate()` ein Kennzeichen `visibility` mitgegeben. Eine 1 bedeutet sichtbar und eine 0 unsichtbar. Falls die Funktion des entsprechenden Filters unsichtbar sein soll, wird dies in den Datensätzen mit einer - 1 gekennzeichnet.

## 4.5 Trace

Um den Ablauf innerhalb der Software nachvollziehen zu können, wird nachfolgend die sogenannte "Trace" beim Aufstarten der Software (Abbildung 4.7) und beim Auslösen eines Ereignisses (Abbildung 4.8) aufgezeigt. Die "Trace" wird innerhalb der Software mithilfe der Klasse traceV4 in der Konsole ausgegeben.

### Trace beim Aufstarten

```
Start via main(String args[])
|----->Attribute von team1.P2Framework2019@1082056158 werden initialisiert ...
|----->Attribute von team1.model.Model@1894530797 werden initialisiert ...
|----->Konstruktor team1.model.Model():team1.model.Model@1894530797 wird ausgeführt ...
|----->Attribute von team1.userinterface.Controller@164660197 werden initialisiert ...
|----->Konstruktor team1.userinterface.Controller():team1.userinterface.Controller@164660197 wird ausgeführt ...
|----->Attribute von team1.userinterface.View@1338627808 werden initialisiert ...
|----->Attribute von team1.userinterface.PlotPanel@2035861432 werden initialisiert ...
|----->Konstruktor team1.userinterface.PlotPanel():team1.userinterface.PlotPanel@2035861432 wird ausgeführt ...
|----->Attribute von team1.userinterface.PlotPanel@1365045686 werden initialisiert ...
|----->Konstruktor team1.userinterface.PlotPanel():team1.userinterface.PlotPanel@1365045686 wird ausgeführt ...
|----->Konstruktor team1.userinterface.View():team1.userinterface.View@1338627808 wird ausgeführt ...
|----->Attribute von team1.userinterface.ButtonPanel@431655273 werden initialisiert ...
|----->Konstruktor team1.userinterface.ButtonPanel():team1.userinterface.ButtonPanel@431655273 wird ausgeführt ...
|----->Attribute von team1.userinterface.FiltertablePanel@1548227145 werden initialisiert ...
|----->Attribute von team1.userinterface.StorageManager@1016480013 werden initialisiert ...
|----->Konstruktor team1.userinterface.StorageManager():team1.userinterface.StorageManager@1016480013 wird ausgeführt ...
|----->Konstruktor team1.userinterface.FiltertablePanel():team1.userinterface.FiltertablePanel@1548227145 wird ausgeführt ...
|----->Attribute von team1.userinterface.InputPanel@758954426 werden initialisiert ...
|----->Konstruktor team1.userinterface.InputPanel():team1.userinterface.InputPanel@758954426 wird ausgeführt ...
|----->Attribute von team1.userinterface.InformationPanel@1928946347 werden initialisiert ...
|----->Konstruktor team1.userinterface.InformationPanel():team1.userinterface.InformationPanel@1928946347 wird ausgeführt ...
|----->Attribute von team1.userinterface.InputSubPanel@1751738947 werden initialisiert ...
|----->Konstruktor team1.userinterface.InputSubPanel():team1.userinterface.InputSubPanel@1751738947 wird ausgeführt ...
|----->Attribute von team1.userinterface.InputSubPanel@792671192 werden initialisiert ...
|----->Konstruktor team1.userinterface.InputSubPanel():team1.userinterface.InputSubPanel@792671192 wird ausgeführt ...
|----->Attribute von team1.userinterface.InputSubPanel@1914644909 werden initialisiert ...
|----->Konstruktor team1.userinterface.InputSubPanel():team1.userinterface.InputSubPanel@1914644909 wird ausgeführt ...
|----->Attribute von team1.userinterface.InputSubPanel@143914969 werden initialisiert ...
|----->Konstruktor team1.userinterface.InputSubPanel():team1.userinterface.InputSubPanel@143914969 wird ausgeführt ...
|----->Attribute von team1.userinterface.InputSubPanel@1716253350 werden initialisiert ...
|----->Konstruktor team1.userinterface.InputSubPanel():team1.userinterface.InputSubPanel@1716253350 wird ausgeführt ...
|----->Attribute von team1.userinterface.InputSubPanel@737303207 werden initialisiert ...
|----->Konstruktor team1.userinterface.InputSubPanel():team1.userinterface.InputSubPanel@737303207 wird ausgeführt ...
|----->Attribute von team1.userinterface.InputSubPanel@1773337982 werden initialisiert ...
|----->Konstruktor team1.userinterface.InputSubPanel():team1.userinterface.InputSubPanel@1773337982 wird ausgeführt ...
|----->Attribute von team1.userinterface.InputSubPanel@799305098 werden initialisiert ...
|----->Konstruktor team1.userinterface.InputSubPanel():team1.userinterface.InputSubPanel@799305098 wird ausgeführt ...
|----->Attribute von team1.userinterface.InputSubPanel@820530598 werden initialisiert ...
|----->Konstruktor team1.userinterface.InputSubPanel():team1.userinterface.InputSubPanel@820530598 wird ausgeführt ...
|----->Attribute von team1.userinterface.InputSubPanel@1991261365 werden initialisiert ...
|----->Konstruktor team1.userinterface.InputSubPanel():team1.userinterface.InputSubPanel@1991261365 wird ausgeführt ...
|----->Attribute von team1.userinterface.InputSubPanel@1307843125 werden initialisiert ...
|----->Konstruktor team1.userinterface.InputSubPanel():team1.userinterface.InputSubPanel@1307843125 wird ausgeführt ...
|----->Attribute von team1.userinterface.InputSubPanel@797640608 werden initialisiert ...
|----->Konstruktor team1.userinterface.InputSubPanel():team1.userinterface.InputSubPanel@797640608 wird ausgeführt ...
|----->Attribute von team1.userinterface.InputSubPanel@1601553816 werden initialisiert ...
|----->Konstruktor team1.userinterface.InputSubPanel():team1.userinterface.InputSubPanel@1601553816 wird ausgeführt ...
|----->Attribute von team1.userinterface.InputSubPanel@470052835 werden initialisiert ...
|----->Konstruktor team1.userinterface.InputSubPanel():team1.userinterface.InputSubPanel@470052835 wird ausgeführt ...
|----->Attribute von team1.userinterface.MenuBar@1073322827 werden initialisiert ...
|----->Konstruktor team1.userinterface.MenuBar():team1.userinterface.MenuBar@1073322827 wird ausgeführt ...
|----->Konstruktor team1.P2Framework2019():team1.P2Framework2019@1082056158 wird ausgeführt ...
```

Abbildung 4.7: Trace beim Aufstarten

## Trace bei einem Event

```

Methode team1.userinterface.InputSubPanel@9039371.stateChanged() wird durch Ereignis ausgelöst ...
|----->Methode team1.userinterface.InputSubPanel@9039371.refreshComponents() wird ausgeführt ...
|----->Methode team1.userinterface.Controller@2109919717.updateParamterValues() wird ausgeführt ...
|----->Methode team1.userinterface.InputPanel@293375778.getEffectiveParameterValues() wird ausgeführt ...
|----->Methode team1.userinterface.FiltertablePanel@1282544970.getSelectedRow() wird ausgeführt ...
|----->Methode team1.model.Model@33990878.updateEffectiveParameterValues() wird ausgeführt ...
|----->Methode team1.userinterface.InputPanel@293375778.getUserInputParameterValues() wird ausgeführt ...
|----->Methode team1.userinterface.FiltertablePanel@1282544970.getSelectedRow() wird ausgeführt ...
|----->Methode team1.model.Model@33990878.updateUserInputParameterValues() wird ausgeführt ...
|----->Methode team1.userinterface.Controller@2109919717.calculateInsertionLoss() wird ausgeführt ...
|----->Methode team1.userinterface.FiltertablePanel@1282544970.getSelectedRow() wird ausgeführt ...
|----->Methode team1.userinterface.FiltertablePanel@1282544970.getSelectedRowVisibility() wird ausgeführt ...
|----->Methode team1.model.Model@33990878.calculate() wird ausgeführt ...
|----->Methode team1.model.Model@33990878.notifyObservers() wird ausgeführt ...
|----->Methode team1.userinterface.View@557838922.update() wird ausgeführt ...
|----->Methode team1.userinterface.PlotPanel@1899123859.update() wird ausgeführt ...
|----->Methode team1.model.Model@33990878.getCM() wird ausgeführt ...
|----->Methode team1.userinterface.PlotPanel@1899123859.setData() wird ausgeführt ...
|----->Methode team1.userinterface.PlotPanel@1549744292.update() wird ausgeführt ...
|----->Methode team1.model.Model@33990878.getDM() wird ausgeführt ...
|----->Methode team1.userinterface.PlotPanel@1549744292.setData() wird ausgeführt ...
|----->Methode team1.userinterface.InputPanel@293375778.update() wird ausgeführt ...

```

Abbildung 4.8: Trace beim event

## 5 Testkonzept

### 5.1 Aufbau

In der unteren Tabelle (Abbildung 5.1) wird aufgeführt, welche Tests durchgeführt werden, um einen Überblick zu geben wie das Testkonzept aufgebaut ist. In der unteren Abbildung (Abbildung 5.2) werden die 3 Testphasen visualisiert. Die detaillierten Testprotokolle, die für die internen und externen Tests verwendet werden, sind im Anhang.

#### Überblick der Tests

Tests Übersicht	Intern	Extern	Ablauf	Ziel des Tests
Code vor der Implementation Testen	X		Die Methoden werden einzeln ausgeführt	Fehler von Vorhinein zu Verhindern
Struktur des Codes überprüfen.	X		Es wird geschaut, ob die geplante Struktur vorhanden ist.	Übersichtlichkeit der Software gewährleisten
Berechnete Werte Validieren	X		Die Berechneten Resultate werden mit Hilfe von MATLAB und MPLAB kontrolliert.	Folgefehler vermeiden.
Kompatibilitätstest	X	X	Die Software wird auf verschiedenen Betriebssystemen und Displays ausgeführt.	Soll eine fehlerfreie Darstellung der Software gewährleisten.
Fehleingaben	X	X	Es werden Eingaben getätigt, die die Software an die Grenzen bringen dürfte.	Die Software soll so stabil wie möglich sein.
Den optische Aufbau der GUI betrachten.	X	X	Beurteilen ob der Aufbau der GUI Sinnvoll ist.	Die GUI Soll schlüssig aufgebaut sein.
Die einzelnen Funktion der GUI testen.	X	X	Überprüfen Slider, Buttons, Menu etc. die richtige Action auslösen.	Die Funktionalität der GUI wird gewährleistet.
Test durch den Auftraggeber		X	Nach Vollendung der Version 0.9.5 wird die Software dem Auftraggeber abgegeben, damit er seine Meinung und Ideen einbringen kann.	Der Auftraggeber soll zufrieden mit dem Endprodukt sein.

Abbildung 5.1: Überblick der Tests

#### Testphasen

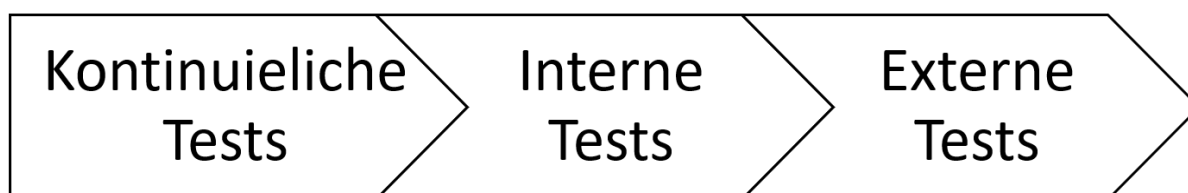
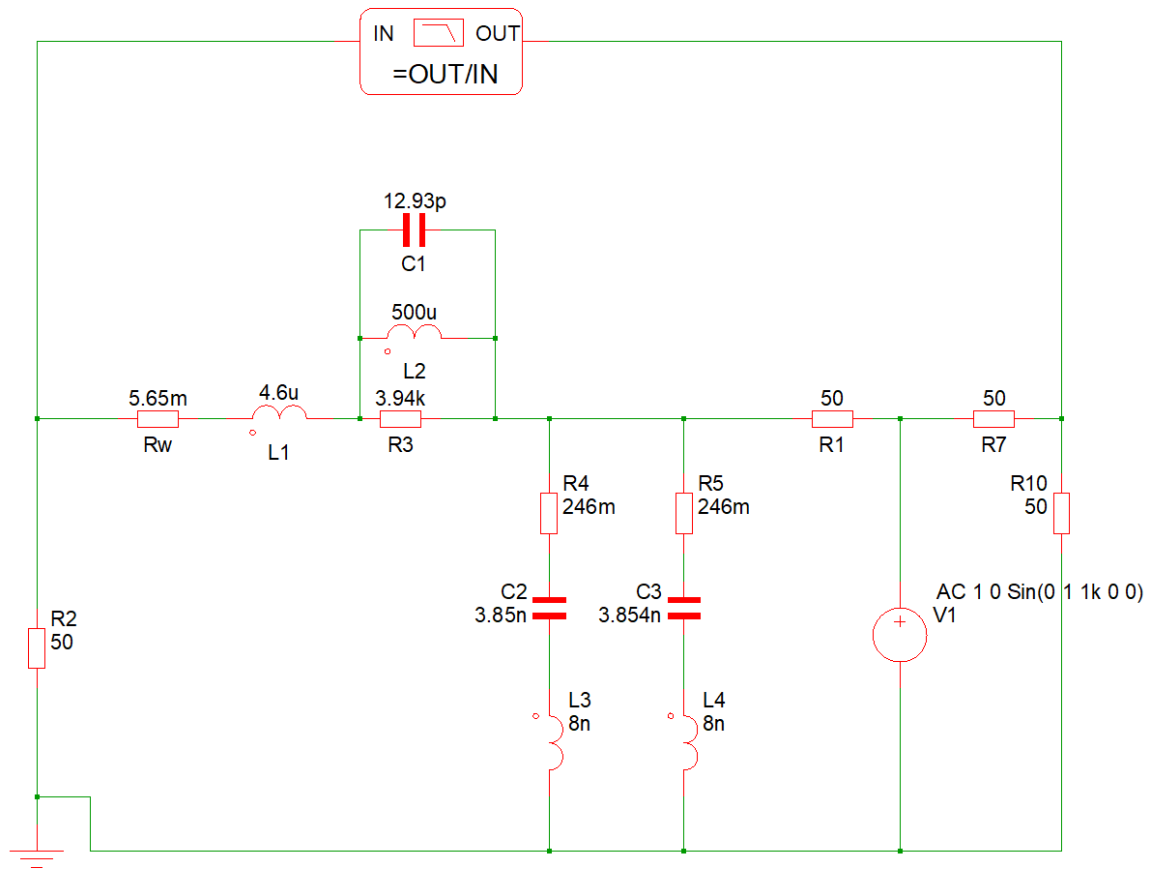


Abbildung 5.2: Testphasen

## 5.2 Validierung

Um sicherzustellen, dass die Simulationen richtig sind und auch korrekt dargestellt werden, werden die Ergebnisse mit entsprechenden Simulationen der Simulationssoftware MPLAB mindi verglichen. Abbildung 5.3 zeigt die Simulierte Gleichtaktschaltung in MPLAB mindi.



**Abbildung 5.3:** Simulationsschaltung der Gleichtaktschaltung in MPLAB mindi

Abbildung 5.4 zeigt die Simulationsergebnisse in MPLAB mindi, welche optisch identisch sind mit den Messresultaten der entwickelten Simulationssoftware (Abbildung 5.5).

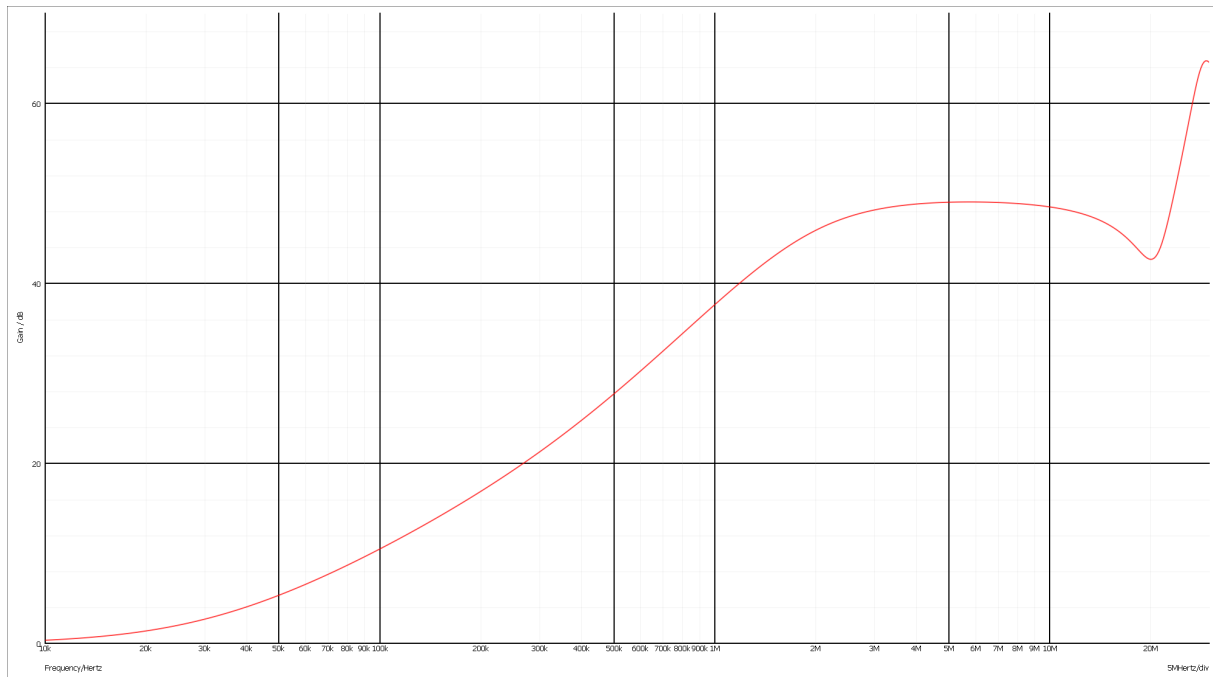


Abbildung 5.4: Simulationsergebnisse der Gleichtaktschaltung in MPLAB mindi

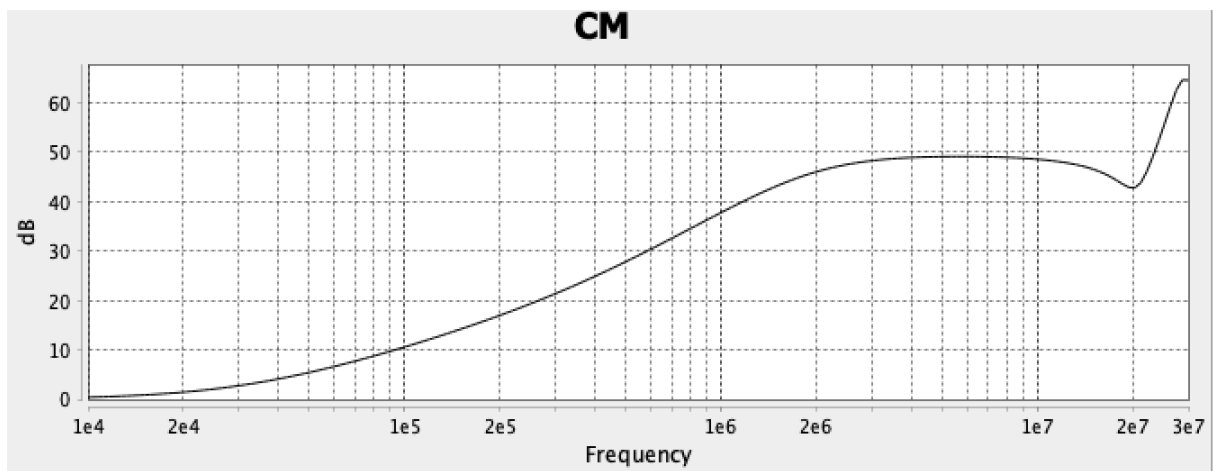
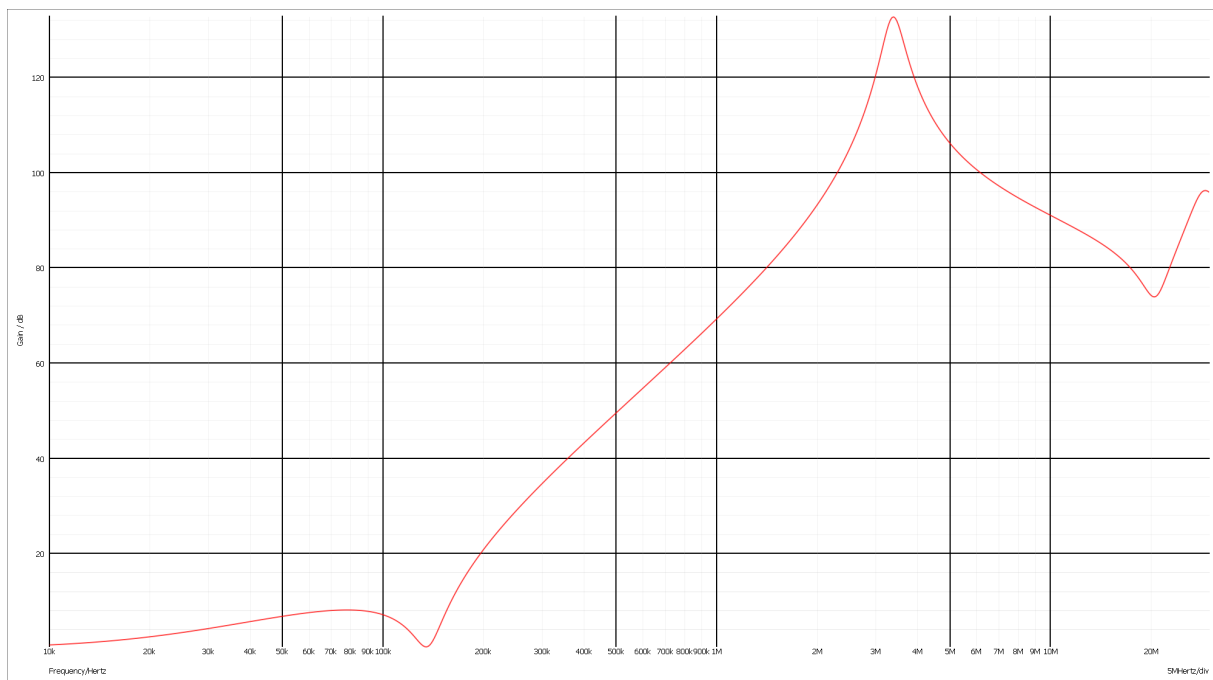


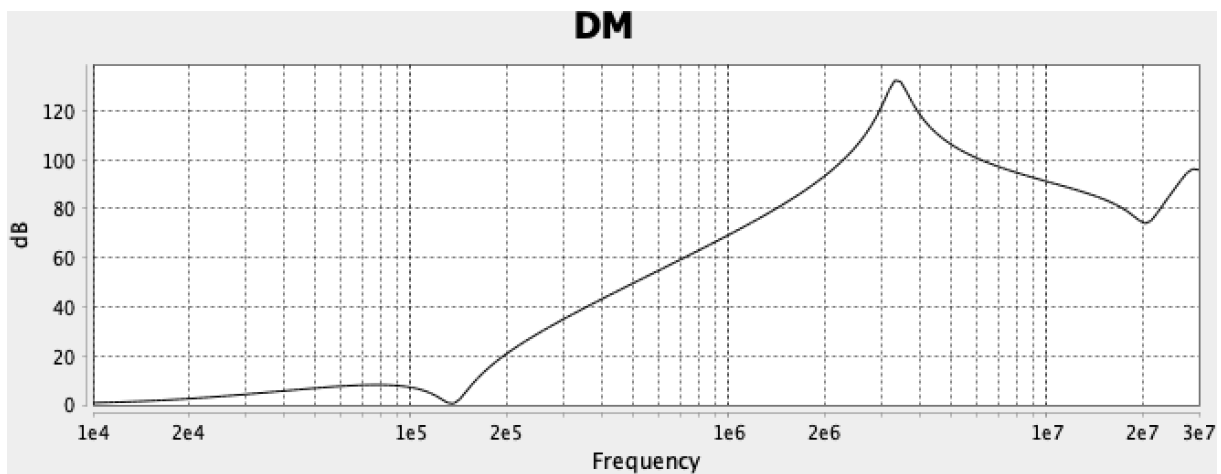
Abbildung 5.5: Simulationsergebnisse der Gleichtaktschaltung in der EMI-Filter Simulationssoftware





**Abbildung 5.7:** Simulationsergebnisse der Gegentaktschaltung in MPLAB mindi

Die Ergebnisse von MPLAB mindi decken sich wiederum mit den Messresultate der entwickelten Simulationssoftware. Abbildung 5.8 zeigt die Resultate der entwickelten Simulationssoftware.



**Abbildung 5.8:** Simulationsergebnisse der Gegentaktschaltung in der EMI-Filter Simulationssoftware

Aus den oben beschriebenen Vergleichen wird geschlossen, dass die Berechnungen korrekt in die Software implementiert wurden.



### 5.3 Erwartungen

Die internen Tests werden die groben Fehler herausfiltern und schaffen eine stabile Grundlage, auf die aufgebaut werden kann. Zudem wird überprüft, ob alle Ziele erreicht wurden. Bei den Fachpersonen wird das Feedback höchst wahrscheinlich sehr umfangreich ausfallen. Es ist zu erwarten, dass Fehler entdeckt werden, die noch nicht bekannt sind oder die Software gar zum Absturz gebracht wird.

Die fachfremden Tester werden dies nicht erreichen, jedoch erhalten wir eine hilfreiche Rückmeldung, was die Benutzerfreundlichkeit betrifft, weil diese Personen einen anderen Blick auf das grosse Ganze haben.

Das Feedback des Auftraggebers wird sehr detailliert ausfallen, weil er genaue Vorstellungen hat was er von dem Produkt haben möchte.

### 5.4 Resultate

In der Abbildung 5.9 sind die Testresultate aufgeführt. Die genauen Test-Beschreibungen sind im Anhang zu finden. Bei den Tests konnten Bewertungen (1-3) abgegeben werden, wie gut die Software auf den jeweiligen Umstand reagiert. Bei den Testpersonen handelt es sich um Teammitglieder und externe Tester.

Bei den Tests 4 und 5 wurden die meisten Fehler gefunden. Beispielsweise reagierte die Software nicht mehr, falls zu viele Filterprofile erstellt wurden. Bei anderen Nutzern haben nicht alle Shortcuts wunschgemäss funktioniert. Der Test 6 (Kompatibilität) konnte nicht von allen Testern durchgeführt werden, weil die dazu nötige Hardware fehlte.

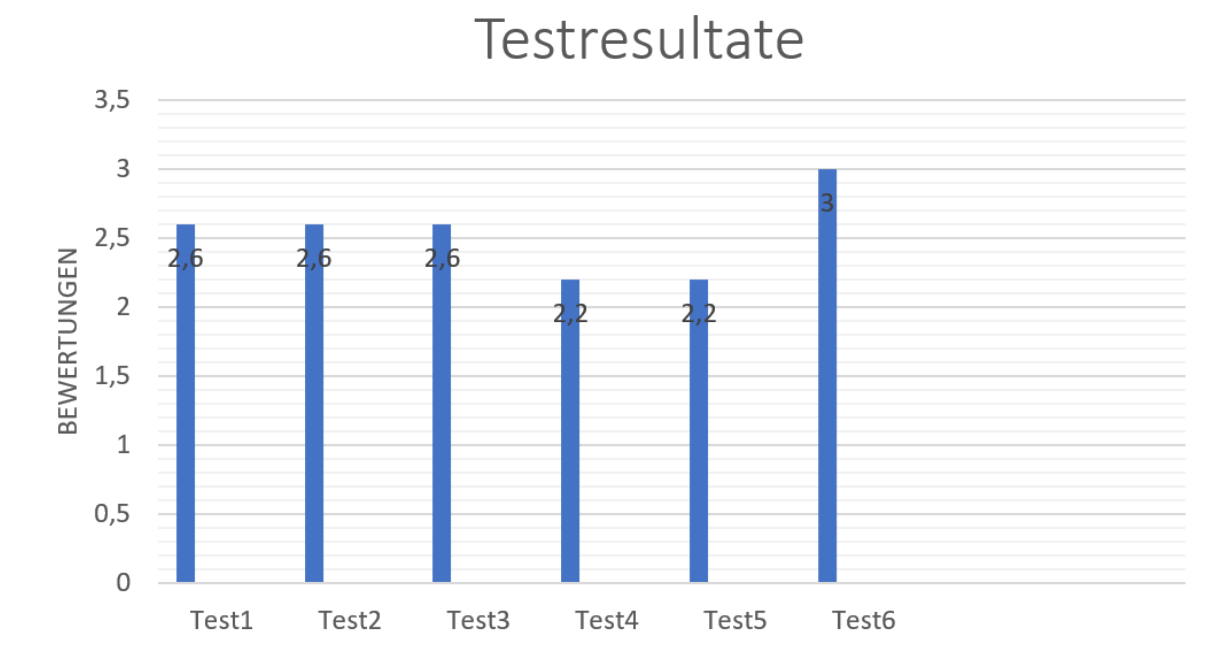


Abbildung 5.9: Bewertungen

## 6 Schluss

Im Laufe des Frühlingssemesters 2019 hat unser Team an einer Software gearbeitet, die die Einfügedämpfung von EMI-Filtern, in Abhängigkeit der Frequenz, darstellen kann. In den ersten Wochen wurde ein Pflichtenheft erstellt, um die technischen Grundlagen zu erarbeiten. Zudem wurde auch ein organisatorisches Pflichtenheft geschrieben, um einen sinnvollen Zeitplan festzulegen und die Zusammenarbeit im Team zu regeln. Im Rahmen einer Zwischenpräsentation wurden diese Pflichtenhefte und den damaligen Stand der Software vorgestellt. In der Projektwoche wurde die Software fertiggestellt (Version 0.9.5). Ausserdem wurde innerhalb dieser Woche ein Grossteil dieses Fachberichtes geschrieben. Anschliessend wurde die letzte Testphase eingeleitet, um die letzten Fehler in der Software auszumerzen.

Nach den erfolgreich durchgeführten Tests, wird die Software dem Auftraggeber zur Verfügung gestellt. Alle im Pflichtenheft gesteckten Muss-Ziele wurden zufriedenstellend umgesetzt. Die Software ist in der Lage die Einfügedämpfung eines EMI-Filters grafisch darzustellen. Ausserdem ist es möglich verschiedene Filterprofile anzulegen und sie zu speichern. Die verschiedenen Filter können gleichzeitig im Plot angezeigt werden.

Die grösste Herausforderung während des Projektes bestand aus der Erarbeitung der elektrotechnischen Grundlagen und die Berechnung der Einfügedämpfung des Netzwerkes in DM und CM. Die Entwicklung des Softwaregrundgerüsts ging zügig vorwärts. Die Schwierigkeiten bei der Software waren beispielsweise die Implementierung der durchgeführten Berechnungen in den Code und die Verarbeitung der Werte, die in die GUI eingegeben werden. Um die Performance nicht einzuschränken, können nur 10 Filterprofile gleichzeitig angezeigt werden.

Schlussendlich konnten alle Schwierigkeiten gut gemeistert werden. Die Software könnte natürlich noch weiterentwickelt werden. Eine weitere sinnvolle Funktion wäre eine Monte-Carlo Analyse. Diese wurde jedoch aus Zeitgründen nicht implementiert. Als abschliessendes Fazit ist zu sagen, dass dieses Projekt erfolgreich und termingerecht durchgeführt werden konnte.

## Literatur

- [1] P. Niklaus, *2-Tore*, März 2019.
- [2] S. Rupp, A. Maier, „Hochfrequenztechnik Teil 2 - Anwendungen“, *Duale Hochschule Baden-Württemberg*, März 2019. Adresse: [http://www.lehre.dhbw-stuttgart.de/%20srupp/HF/Hochfrequenztechnik\\_T2ELN3001\\_Teil\\_2\\_SR.pdf](http://www.lehre.dhbw-stuttgart.de/%20srupp/HF/Hochfrequenztechnik_T2ELN3001_Teil_2_SR.pdf).
- [3] H. Bernstein, *NF- und HF-Messtechnik - Messen mit Oszilloskopen, Netzwerkanalysatoren und Spektrumanalysator*, 1. Aufl. Berlin Heidelberg New York: Springer-Verlag, 2015, ISBN: 978-3-658-07378-7.
- [4] L. Dalessandro, „EMI-Filter“, *Windisch: Hochschule für Technik der Fachhochschule Nordwestschweiz (FHNW)*, 21. März 2019.
- [5] Jörg Czeschla und javabeginners.de, *MVC Design Pattern*, März 2018. Adresse: [https://javabeginners.de/Design\\_Patterns/Model-View-Controller.php](https://javabeginners.de/Design_Patterns/Model-View-Controller.php) (besucht am 5. Apr. 2019).
- [6] Prof. Dr. Richard Gut, „MVC Pattern“, 2019.
- [7] A. Viklund. (2019). JfreeChart, Adresse: <http://www.jfree.org/jfreechart> (besucht am 5. Apr. 2019).

## 7 Anhang

### 7.1 Testkonzept

# Testprotokoll Team 1

Name des Testers:

---

Datum :

---

Softwareversion: 0.9




---




Test Art (intern oder extern):




---




**Test Übersicht**



<b>1. Test</b>	Software öffnen und schliessen. Schieberegler und Buttons auf die Funktion prüfen. Das Fenster grösser kleiner machen.
<b>2. Test</b>	Das Aussehen der GUI betrachten. Auf Vollständigkeit testen. (Gesamteindruck)
<b>3. Test</b>	Kontrollieren, ob der Plot gezeichnet wird. Und Filtertypen gespeichert werden können.
<b>4. Test</b>	Die Menu-Funktionen testen. <del>Shurcuts</del> ausprobieren
<b>5. Test</b>	Fehleingaben machen.
<b>6. Test</b>	Verschiedene Betriebssysteme und Displays verwenden (falls vorhanden).
<b>7. Test</b>	Code auf Übersichtlichkeit prüfen. (Experten)

Test Nr.	Der Test	Bewertung	Anmerkungen (Fehler)
1.	Software öffnen und schliessen. Schieberegler und Buttons auf die Funktion prüfen. Das Fenster grösser kleiner machen.	Gut  Mässig  Schlecht 	
<b>Kommentar / Verbesserungsvorschlag</b>			




Test Nr.	Der Test	Bewertung	Anmerkungen (Fehler)
2.	Das Aussehen der GUI betrachten.	Gut  Mässig  Schlecht 	
<b>Kommentar / Verbesserungsvorschlag</b>			

Test Nr.	Der Test	Bewertung	Anmerkungen (Fehler)
3.	Kontrollieren, ob der Plot gezeichnet wird. Und Filtertypen gespeichert werden können.	Gut  Mässig  Schlecht 	
<b>Kommentar / Verbesserungsvorschlag</b>			

Test Nr.	Der Test	Bewertung	Anmerkungen (Fehler)
4.	Die Menu-Funktionen testen	Gut  Mässig  Schlecht 	
<b>Kommentar / Verbesserungsvorschlag</b>			




Test Nr.	Der Test	Bewertung	Anmerkungen (Fehler)
5.	Fehleingaben machen.	Gut  Mässig  Schlecht 	

**Kommentar / Verbesserungsvorschlag**

Test Nr.	Der Test	Bewertung	Anmerkungen (Fehler)
6.	Verschiedene Betriebssysteme und Displays verwenden (falls vorhanden).	Gut  Mässig  Schlecht 	

**Kommentar / Verbesserungsvorschlag**



Test Nr.	Der Test	Bewertung	Anmerkungen (Fehler)
7.	Code auf Übersichtlichkeit prüfen.(Experten).	Gut  Mässig  Schlecht 	
<b>Kommentar / Verbesserungsvorschlag</b>   			

## Abnahmeprotokoll Software

**Ersteller:** \_\_\_\_\_

**Auftraggeber:** \_\_\_\_\_

**Auftrag:** \_\_\_\_\_

**Version:** \_\_\_\_\_

**Datum:** \_\_\_\_\_

### **Ziele**

Berechnungen und GUI getrennt	
Berechnungszeit < 500 ms	
Verstellbare Parameter	
CM-, DM-Berechnung	
CM-, DM-Darstellung	
Feinjustierung mit Schieberegler +/- 30%	
Darstellung im Frequenzbereich bis 30MHz	
Zahlenwerte könne eingegeben werden	
Mehrere Plots gleichzeitig darstellen	
Unabhängigkeit von Betriebssystemen	
Schutz vor Fehleingaben	
Anpassung von Farbe, Darstellung und Schrift	
Zoom Möglichkeit	

**Kommentare**

--

**Test**

Kontinuierliche Tests	
Interne Tests	
Externe Tests durch Fachpersonen und Laien	
Test der 0.9.5 durch den Auftraggeber	

**Projektleiter****Auftraggeber**

---

Unterschrift, Ort, Datum

---

Unterschrift, Ort, Datum