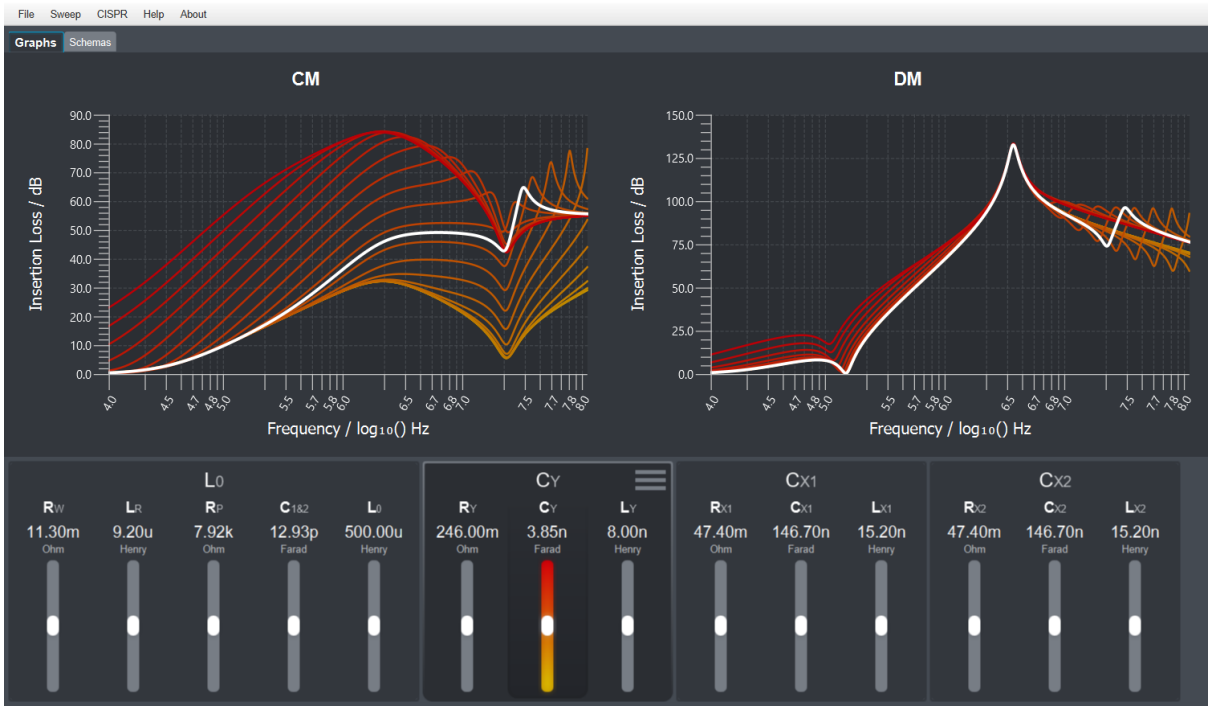
Fachhochschule Nordwestschweiz
Hochschule für Technik

Fachbericht



«DJ» EMI Filter für Netzteil

Pro2E - Team 5

Auftraggeber: Luca Dalessandro

Dozierende: Anita Gertiser
Pascal Buchschacher
Peter Niklaus
Sebastian Gaulocher
Richard Gut

Projektteam: Marina Taborda, Projektleiterin
Michel Alt, Stv. Projektleiter
Frank Imhof
Luca Krummenacher
Richard Britt
Fady Hanna

Windisch, 13. Juni 2019

Abstract

Nahezu jedes Elektrogerät enthält heutzutage einen Filter. Der Filter dient dem Gerät, um andere Geräte nicht zu stören und umgekehrt. Dieser Filter besteht aus verschiedenen realen Bauteilen, welche einen Einfluss auf die Dämpfungen des Filters haben. Für die Herstellung von solchen EMI Filtern ist es sinnvoll, den Einfluss der einzelnen Bauteile zu kennen. Dafür eignet sich ein Tool, in dem man die Werte der verschiedenen Bauteile verändern kann und die Veränderung der Einfügungsdämpfung graphisch dargestellt wird. Die Firma Schaffner Group produziert EMI Filter und hat der Fachhochschule Nordwestschweiz den Auftrag zu dieser Arbeit gegeben.

Das Ziel dieser Arbeit ist es, eine Software zu programmieren, welche eine Sensitivitätsanalyse der Einfügungsdämpfung eines EMI-Filters ermöglicht. Dazu sollen die einzelnen Parameter eines Bauelements um ± 30 % verstellt werden können, die Veränderung des Filterverhaltens soll in einer Einfügungsdämpfung-Frequenz-Kurve ersichtlich sein. Ausserdem müssen die Schaltungen des Filters analysiert werden und so vereinfacht werden, dass die Berechnungen möglichst einfach in die Software implementiert werden können. Die Darstellung der Software soll in Form eines «DJ-Mischpults» sein. Die Werte der Parameter der Schaltung sollen mit einem Slider und durch nummerische Eingabe verändert werden können. Zusätzlich soll die Software die beiden Graphen und die Schemas der Schaltung aufzeigen.

Für die Software eignet sich die objektorientierte Programmierung in Java. Das Framework JavaFX simplifiziert zusätzlich die Programmierung der interaktiven Bedienoberfläche, indem das Design klar von den Berechnungen getrennt wird. Um den Auftrag angemessen ausführen zu können, sind die elektrotechnischen Aspekte hinter den Störungsverhalten eines EMI-Filters erarbeitet und anschliessend in die Software implementiert worden.

Nach dem Start des Programms erscheinen zwei Diagramme mit Graphen zu den jeweiligen Störungsarten. Durch «Doppelklick» auf einen der beiden Graphen soll dessen Ansicht vergrössert oder wieder verkleinert werden. Darunter befindet sich das Mischpult, welches jedem Parameter des Filters einen Schieberegler zuordnet. Wenn mit der Maus über einen Schieberegler gefahren wird, werden die dazugehörigen Abweichungen von ± 30% im Graphen farbig angezeigt. Durch Anklicken der Schieberegler, passen sich die Werte des Parameters und die Kurve an. Die Hamburger oben rechts in den Komponentenboxen ermöglichen den Wechsel zur Textfeldanzeige, in denen dann die Werte der einzelnen Komponenten manuell im ENG-Format eingegeben werden können. Die Eingabe wird anschliessend übernommen und die resultierende Kurve dargestellt.

In der Menüleiste können die Referenzwiderstände und die Min/Max-Werte der Schieberegler eingestellt werden. Ausserdem können die Tooltipps ein oder ausgeschaltet werden. Unterhalb der Menüleiste kann zwischen verschiedenen Tabs gewechselt werden. Im ersten Tab sieht man die Graphen der beiden Störungsarten. Im zweiten Tab sind die Schaltbilder mit den Bezeichnungen der Komponenten ersichtlich.

Inhaltsverzeichnis

1 Einleitung

Damit Elektrogeräte keine Störungen ins Netz aussenden und nicht selber von diesen beeinflusst werden, enthalten Geräte einen EMI (elektromagnetische Interferenzen) Filter. Jedes Bauteil des Filters hat wiederum unterschiedliche Auswirkungen auf die Einfügungsdämpfung eines Filters. Damit die Firma Schaffner, welche solche Filter produziert, die Einflüsse der verschiedenen Parameter auf die Einfügungsdämpfung simulieren kann, wird die entsprechenden Dämpfung graphisch dargestellt.

Der Auftrag des Moduls pro2E des Studiengangs Elektro- und Informationstechnik an der Fachhochschule Nordwestschweiz ist es, ein Anwendungsprogramm zu entwickeln, welche es ermöglicht die Einfügungsdämpfung eines EMI-Filters in Abhängigkeit zur Frequenz darzustellen. Anhand eines GUIs, welches ein Mischpult mit Schiebereglern enthalten soll, lassen sich die Werte der Parameter beliebig verändern. Anschliessend werden die Einflüsse auf das Verhalten in einem Graphen aufgezeigt. Die Berechnungen zu den Graphen sind in der Software implementiert und sorgen für eine korrekte Darstellung der Kurven.

Gemäss Aufgabenstellung (im Anhang E) soll das GUI gebrauchstauglich (nach DIN EN ISO 9241-11) sein. Somit ist ein Ziel dieser Arbeit, dass ein Laie das Programm ohne Erklärung und ohne Anleitung bedienen kann. Ausserdem soll die Softwareanwendung auf den Betriebssystemen MacOS (ab Version 10.11) und auf Windows (7 oder neuer) problemlos funktionieren. Die Ziele der Berechnungen sind, dass die Simulationen der Schaltungen mit deren Berechnungen übereinstimmen und korrekt sind.

Das Programm wird in Java geschrieben und basiert auf dem Model-View-Controller-Framework, um die Software später einfach verändern zu können. Die Berechnungen finden im Model statt. In der View soll die Bedienoberfläche gebaut und die Graphen gezeichnet werden. Der Controller dient als Schnittstelle zwischen View und Model. Damit die Präsentationslogik Layout des Programms möglichst von den Berechnungen zu trennen, eignet sich das JavaFX Framework. Damit kann können Layout und Design mit Hilfe von Cascading Style Sheets (CSS) schlang und einfach hält. Für die Berechnung der Graphen liest das Model die Daten aus und berechnet die Kurven. Der Controller verbindet die View und das Model miteinander.

Die Basis für die Berechnungen bilden die Schemata, welche von Schaffner zur Verfügung gestellt wurden. Für die Berechnungen der Gleichtakt- und Gegentaktstörungen sind die Schemata vereinfacht worden und die darin enthaltenen Bauelemente sind in Längs- und Querimpedanzen eingeteilt. Diese sind anschliessend zu einer Gesamtmatrix zusammengeführt worden. Mit Hilfe von Matlab berechnen sich daraus die Einfügungsverluste und stellt diese in einem Kurvendiagramm dar.

Um in die fachlichen Kapitel einzuleiten, werden in diesem Bericht als erstes die Grundlagen erläutert. Danach wird im Kapitel «Elektrotechnik» die Theorie des EMI- Filters und die Vorgehensweise bei den Berechnungen genauer aufgezeigt. Die Softwarestruktur und die Bedienoberfläche sind im Kapitel 3 erklärt. Anschliessend folgt die Überprüfung der Berechnungen und der Software.

2 Theoretische Grundlagen

In den theoretischen Grundlagen wird erklärt, was ein EMI Filter genau ist, was für Störungsarten es genau gibt und wie die Einfügungsdämpfungen definiert sind. Ausserdem wird genauer auf die parasitären Elemente des Filters eingegangen.

2.1 EMI Filter

Nahezu jedes elektrische Gerät besitzt ein Schaltnetzteil, um die Netzspannung auf die benötigte Spannung zu regeln. Betrachtet man die Eingangsspannung ohne Netzfilter, wird man auf dem ganzen Frequenzspektrum, d.h. von Netzfrequenz bis zu mehreren MHz Störungen feststellen. Die Aufgabe vom EMI (elektromagnetische Interferenzen) Filter ist es, diese Störungen zu filtern, so dass keine anderen Geräte gestört werden. Die EMI Frequenzen sind im Frequenzspektrum von 150kHz bis 30MHz bestimmt. Damit elektrische Geräte eingesetzt werden dürfen, muss jedes Schaltnetzteil sich an bestimmte Normen im Bereich EMV halten.

Ein EMI Filter für einphasige Geräte besteht nur aus wenigen Bauteilen. Zwei X-Kondensatoren, zwei Y-Kondensatoren, einer Drossel mit zwei Wicklungen, welche um einen Ring gewickelt sind und einem Widerstand. Diese Schaltung kann sehr kompakt verbaut werden, was in folgendem Filter (Abbildung 1) von Schaffner sichtbar wird. Die Beschriftungen im Schema, werden von EMI-Fachpersonen so verwendet.

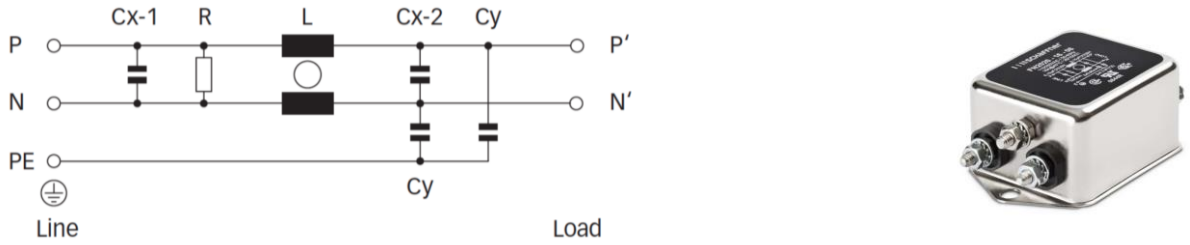


Abbildung 1:Schaltung des FN 2020 Filters (Schaffner), sowie der Filter selbst

Die gekoppelte Spule (L) ist in der Lage, Gleichtaktstörungen (CM) zu filtern. Diese Störungen treten gleichzeitig auf beiden Leitungen auf. Die Spule sollte einen möglichst grossen Kopplungsfaktor besitzen, idealerweise sollte dieser 1 sein.

Die Y-Kondensatoren, welche gegen Erde geschaltet sind, sind ebenfalls dazu da, um CM-Störungen zu filtern. Diese müssen jedoch eine sehr hohe Überspannungsfestigkeit besitzen, um beispielsweise bei einem Blitzschlag keinen Kurzschluss im Gehäuse zu verursachen.

Störungen zwischen den Zuleitungen, so genannte Gegentaktstörungen (DM), werden mithilfe der X-Kondensatoren gedämpft.

Der Widerstand, welcher parallel zu CX-1 liegt, wird aus Sicherheitsgründen benötigt. Dieser ist typischer Weise 1 MΩ gross. Er entlädt den CX-1 Kondensator, nachdem das Filter vom Netz getrennt wird. [1], [2]

2.2 Störungsarten

Die existierenden EMV Normen gelten für die Gesamtstörungen. Doch in der Praxis wird einfachheitshalber die Gesamtstörung in Gleichtaktstörungen, Common Mode (CM) und Gegentaktstörungen, Differential Mode (DM) gesprochen.

Gleichtaktstörungen wirken auf beide Leitungen gleichermassen ein. Die Störspannungen besitzen in beiden Leitungen die gleiche Amplitude, sowie Phasenlage. CM Störungen entstehen oft durch kapazitive Kopplung, wie in Abbildung 2 sichtbar ist. Diese entstehen aufgrund von verschiedenen Potentialen entlang des Übertragungsweges. Damit das Signal wirklich gestört wird, ist ein langer Stromweg nötig. Dies ist oft der Fall bei gemeinsamen Bezugssignalen, zum Beispiel einer gemeinsamen Masse oder Erde.

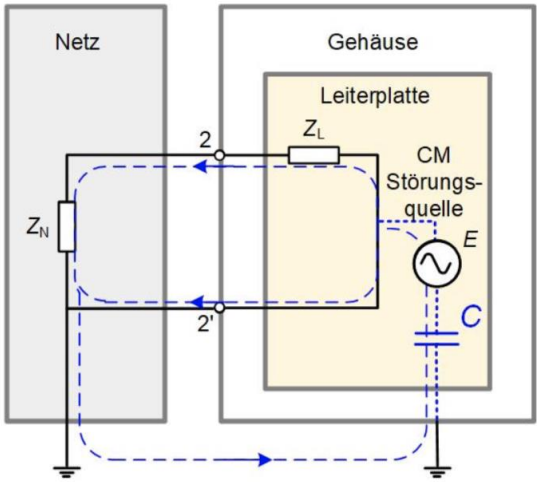


Abbildung 2: Stromzirkulation der Störungen im CM Mode

Die Gegentaktstörungen fliessen in die genau gleiche Richtung, wie die Nutzsignale. Das heisst sie Überlagern das eigentliche Signal. Die Ursache bei diesen Störungen kann eine induktive Kopplung sein. Dabei beeinflussen benachbarte Wechselstrom Signale aufgrund ihres Magnetfeldes die Nutzleiter. Der Leiter erzeugt mit seinem Magnetfeld in der gestörten Schaltung eine Spannung, welche sich wie eine zusätzliche Quellspannung verhält. Diese ist eine Ursache für das Gegentaktstörungen. In der Abbildung 3 sind die Störungen innerhalb einer Schaltung dargestellt.

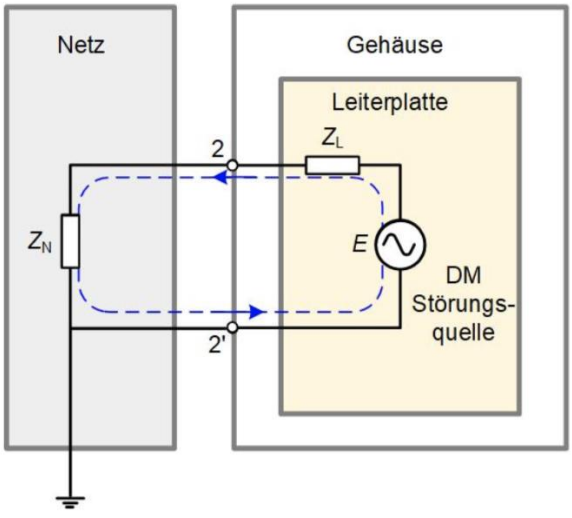


Abbildung 3: Stromzirkulation der Störungen im DM Mode

2.3 Definition Einfügungsdämpfung «Insertion loss»

Die Leistung eines EMI Filters wird mit den Einfügungsdämpfung in Abhängigkeit der Frequenz bestimmt. Diese Funktion lautet:

$$|H(j\omega)| = 20\,log\,\frac{|U_{20}|}{|U_2|}$$

U_{20}: Lastspannung gemessen ohne Filter mit einer Last von 50 Ω

U_2: Lastspannung gemessen mit EMI Filter

Die Einfügungsdämpfung kann auch mit Hilfe der Leistung berechnet werden:

$$|H(j\omega)| = 10\,log\,\frac{|P_{20}|}{|P_2|}$$

P_{20}: Leistung gemessen ohne Filter mit einer Last von 50 Ω

P_2: Leistung gemessen mit EMI Filter

Aus der Definition geht hervor, dass die Einfügungsdämpfung nicht mit einer Messung bestimmt werden kann. Zuerst wird die Schaltung ohne Filter gemessen (Abbildung 4) danach mit Filter (Abbildung 5). Daraus kann man mit obiger Formel die Einfügungsdämpfung berechnen.

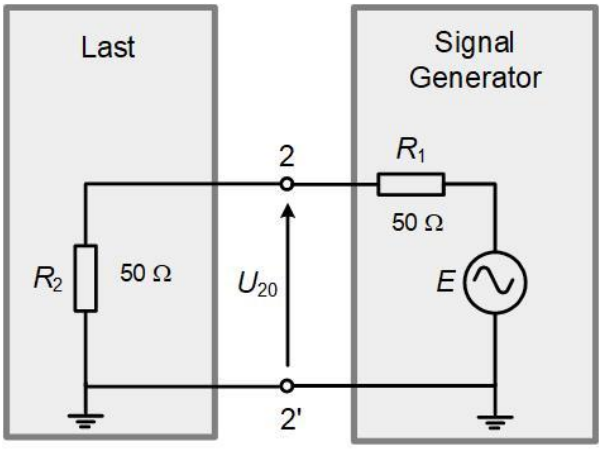


Abbildung 4: Lastspannung ohne Filter

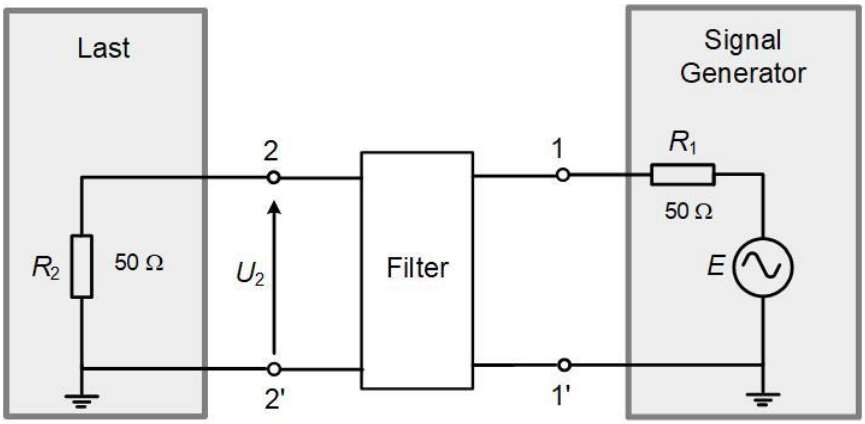


Abbildung 5: Lastspannung mit EMI Filter

2.4 Parasitäre Parameter

In der Realität verhalten sich die Bauteile eines EMI-Filters leider nicht genau so, wie es idealerweise angenommen wird. Jedes Bauteil hat aufgrund der physikalischen Gegebenheiten eine Ersatzschaltung mit den parasitären Parametern. Die Ersatzschaltung für eine Spule ist in Abbildung 6 ersichtlich, jene für den Kondensator in Abbildung 7. Jedoch sind diese auch nur eine Näherung der Realität.

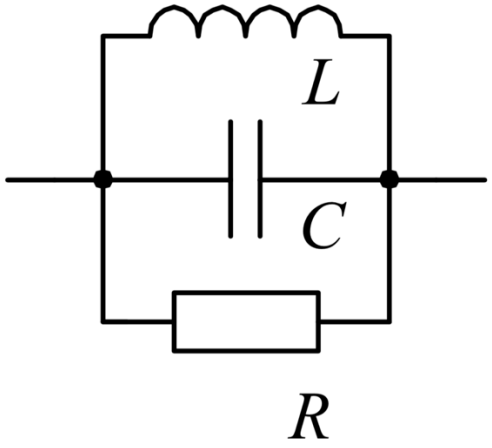


Abbildung 6: Ersatzschaltung Spule

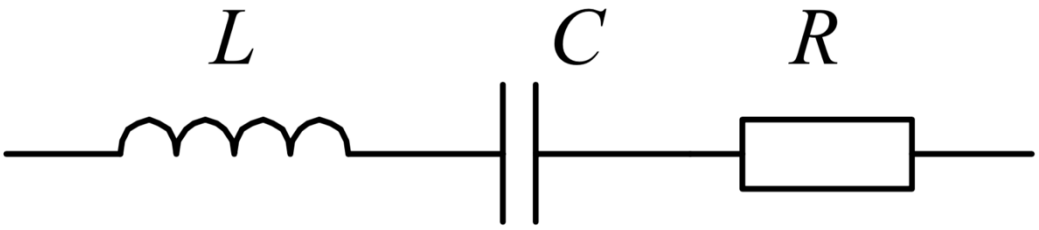


Abbildung 7: Ersatzschaltung für einen realen Kondensator

Betrachtet man diese Ersatzschaltbilder, sieht man deutlich einen Serie- und Parallelschwingkreis. Weit unterhalb der Resonanzfrequenz haben die parasitären Parameter jedoch noch keinen grossen Einfluss. Im Resonanzfall haben wir aber einen Serie- oder Parallelschwingkreis, wobei die reaktiven Elemente wegfallen. Danach wirkt die Spule nicht mehr als Induktivität und der Kondensator nicht mehr als Kapazität, sondern genau umgekehrt. Die Resonanzfrequenzen liegen dabei je nach Wert der Elemente bei unterschiedlichen Frequenzen. Der reale Widerstand kann auch mit einem Ersatzschema beschrieben werden, dies wurde aber in dieser Schaltung nicht berücksichtigt, da der Widerstand selbst einen vernachlässigbaren Einfluss auf die Gesamtschaltung hat.

3 Software

Die Software «EMI-DJ» ist eine Desktop-Anwendung für die Betriebssysteme MacOS und Windows. Sie ist in der Programmiersprache Java geschrieben und verwendet das Framework JavaFX. In diesem Kapitel wird der Aufbau und das Verhalten der Software «EMI-DJ» genauer beschrieben.

3.1 Topdown

Das Frequenzverhalten des EMI-Filters von Schaffner kann mit der Applikation genauer untersucht werden: Sie bietet Schieberegler und Textfelder, mit denen die Komponenten der Schaltung dimensioniert werden können. Wird eine Komponente neu dimensioniert, berechnet die Applikation automatisch die CM und DM Einfügungsverluste und stellt diese grafisch als schwarze Kurve dar. Eine Besonderheit der Applikation ist in der Abbildung 8 zu sehen: die Auswirkungen eines parasitären Parameters auf die Einfügungsverluste werden als zusätzliche farbige Kurven gezeichnet. Die Farben gelb-rot wurden bewusst gewählt, da sich Farben, welche sich ähnlich sind besser für farbenblinde Menschen eignen.

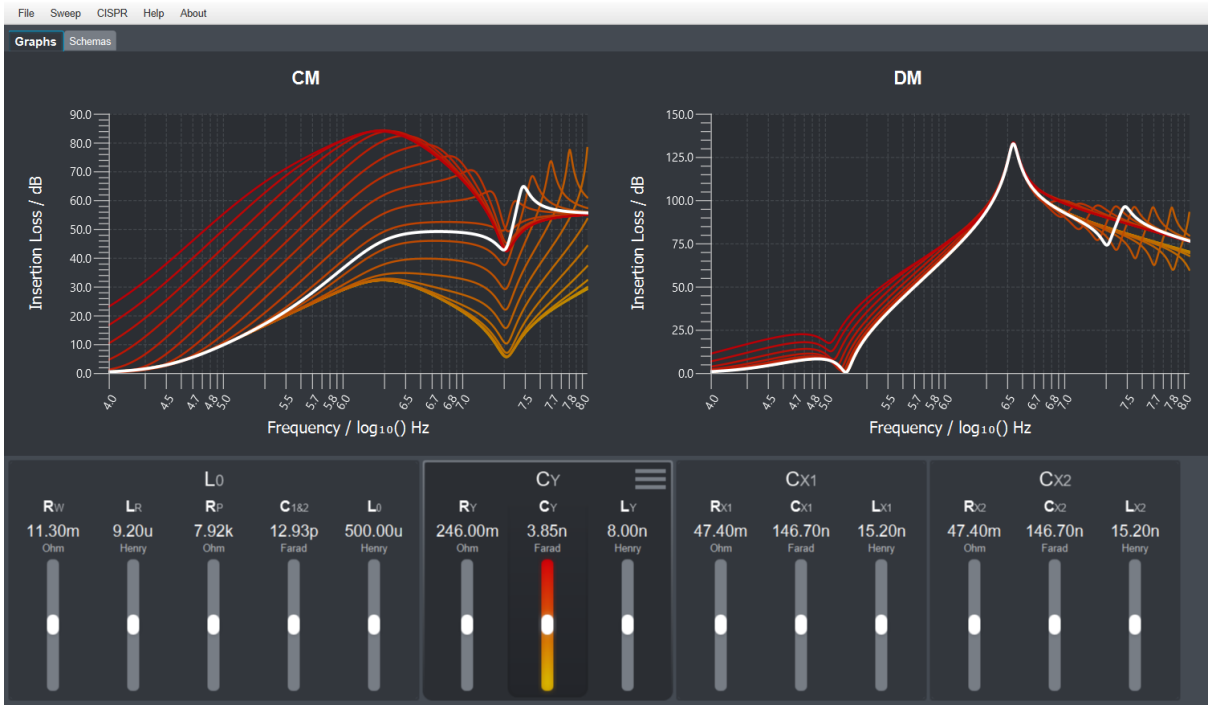


Abbildung 8: Bedienoberfläche

Die Applikation ist im klassischen Model-View-Controller und Observable/Observer Entwurfsmuster, gemäss Klassendiagram im Anhang D, programmiert und besteht aus folgenden Elementen: Circuit, View, Controller und Model.

Das **Circuit** stellt die Schaltung des Filters als Java-Objekt dar. Hier sind alle Zweitore und deren Komponenten gespeichert.

Die **View** dient als Schnittstelle zwischen Nutzenden und dem Circuit. Sie beinhaltet eine Custom-MenuBar für globale Einstellungen und Aktionen, eine TabPane für die Einsicht in Grafen und Schemas und eine CircuitPane, wo die Komponenten des Circuits grafisch repräsentiert werden. Die View besitzt eine Referenz auf Circuit und Controller. Damit können die in der CicuitPane enthaltenen

Schieberegler und Textfelder direkt auf die Komponenten des Circuits zugreifen und die Neuberechnungen anschliessend ausgelöst werden.

Alle Events, die in der View entstehen, leiten ihre Anfrage zum **Controller** weiter. Ist eine Neuberechnung der Einfügungsverluste nötig, werden die entsprechenden Funktionen des Models aufgerufen. Bewirkt der Event explizit eine Änderung der Darstellung, so wird nur die View aktualisiert.

Im **Model** befinden sich Funktionen und Variablen, die zur Berechnung der Einfügungsverluste nötig sind. Das Model besitzt ebenfalls eine Referenz auf das Circuit, so dass es auf dessen momentanen Werte zugreifen kann. Nach erfolgreicher Berechnung benachrichtigt das Model die bei ihm als Observer registrierte View via **notifyObservers()**, damit diese die neu berechneten Einfügungsverluste holen und darstellen kann. Das Zusammenspiel von Model, View und Controller ist in Abbildung 9 schematisch dargestellt.

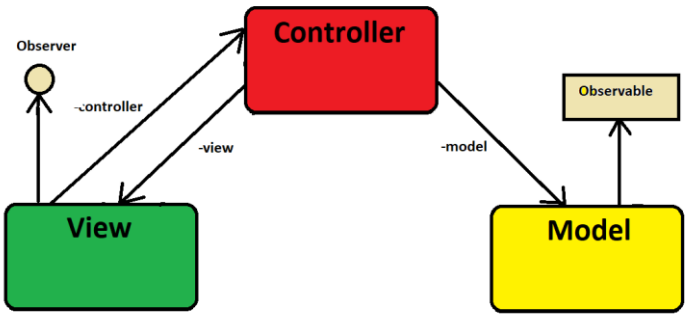


Abbildung 9: MVC Schema

3.2 View

In diesem Abschnitt wird beschrieben, wie die Bedienoberfläche aufgebaut ist.

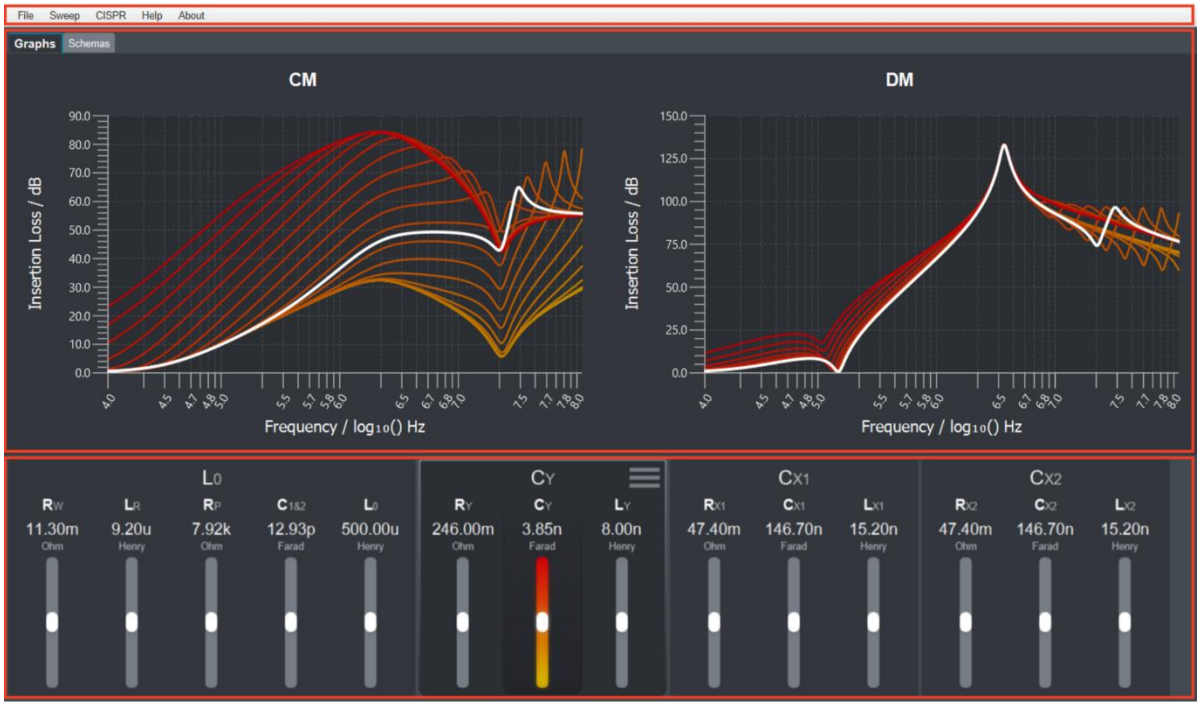


Abbildung 10: Gliederung des Programms

Die oberste Instanz bezüglich des Layouts der Applikation ist die Klasse View (extends Borderpane). Als Borderpane beinhaltet sie die Bereiche Top, Center und Bottom, die in Abbildung 10 rot umrahmt und in genannter Reihenfolge von oben nach unten nummeriert zu sehen sind. Diese werden nun genauer beschrieben.

3.2.1 Top

Abbildung 11: CustomMenuBar

Im Top Bereich befindet sich die **CustomMenuBar,** gezeigt in Abbildung 11. Diese enthält die Menus File, Sweep, CISPR, Help und About.

File: Speichern oder Laden des gesamten Zustands der Applikation als JSON-File.

Sweep: Setzen des Schiebereglerverhaltens: linear ±30% oder logarithmisch 10^{-3} bis 10^{3}

CISPR: Auswahl der Referenzwiderstände R1/R2: 50Ω/50Ω, 100Ω/0.1Ω oder 0.1Ω/100Ω

Help: Ein- und Ausschalten der ToolTips und oder Öffnen des Getting-Started Fensters

About: Öffnen des About Fensters

3.2.2 Center

Der mittlere Bereich der Applikation dient der Darstellung sowohl der Einfügungsverluste wie auch der Schemadiagramme. Um Platz zu sparen, wurde dieser Bereich als TabPane gestaltet. Durch Anwählen des Reiters wird das entsprechende Element (GraphPane oder SchemaPane) angezeigt und das andere in den Hintergrund gerückt.

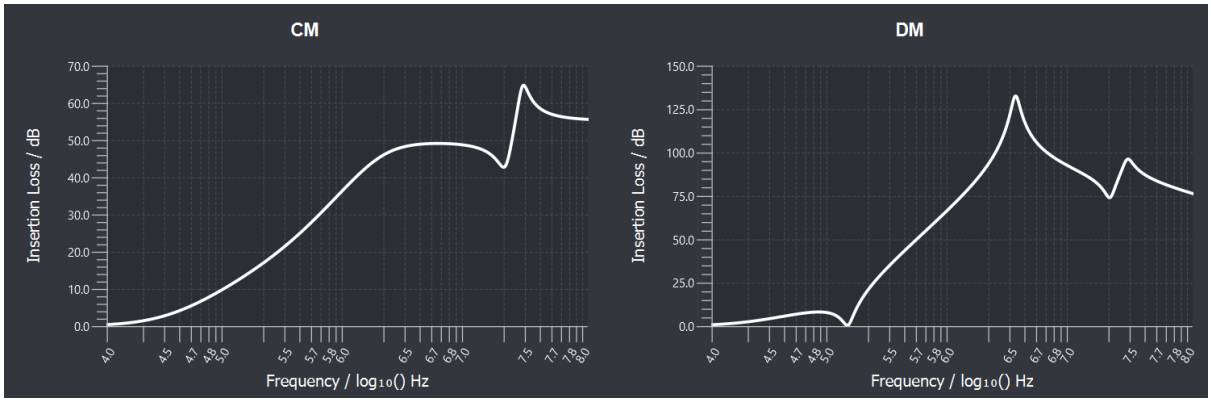


Abbildung 12: GraphPane

Die GraphPane, die in Abbildung 12 zu sehen ist, beinhaltet zwei Instanzen der Klasse GraphBox und legt diese horizontal aus. Sie beinhalten je eine Instanz der Klasse XYChartPane, die die logarithmische Darstellung von Kurven mit unterschiedlichen Farben und Formen erlaubt.

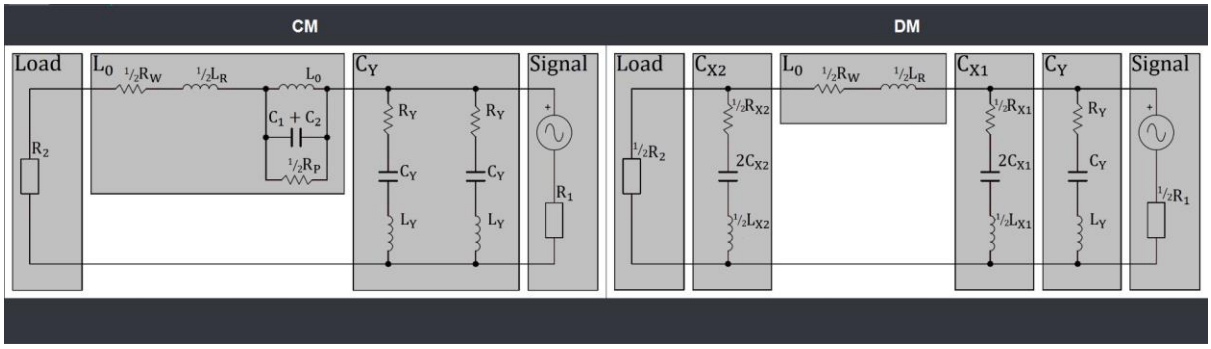


Abbildung 13: SchemaPane

Die SchemaPane, gezeigt in Abbildung 13, ist gleich aufgebaut wie die GraphPane, nur enthält sie zwei Instanzen der Klasse SchemaBox, wo die Schemata der vereinfachten Schaltungen eingesehen werden können.

3.2.3 Bottom

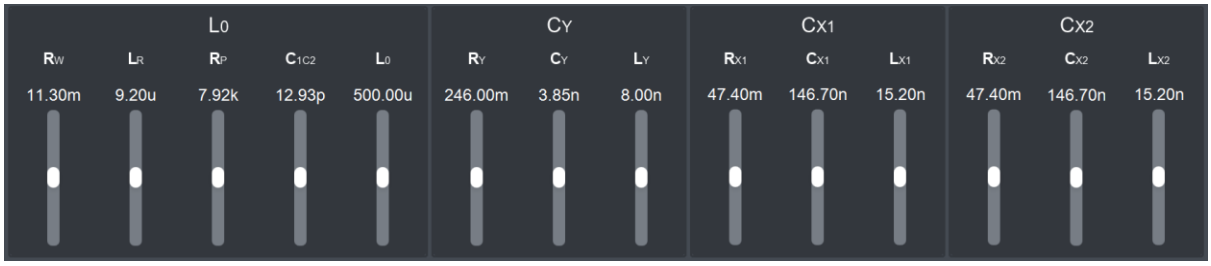


Abbildung 14: Bottom

Dieser Bereich besteht aus einer Instanz der Klasse CircuitPane (extends HBox), die in sich drin alle Komponenten der Schaltung als Instanzen der Klasse CompBox enthält und diese horizontal auslegt. Dies ist in der Abbildung 14 zu sehen.

Abbildung 15: Die beiden Gestalten der CompBox

Die CompBox (extends VBox) ist eines der zentralsten Elemente der Bedienoberfläche, da sie als Hauptschnittstelle zwischen Nutzenden und Applikation fungiert. Die beiden Gestalten, die sie annehmen kann, sind in Abbildung 15 gezeigt.

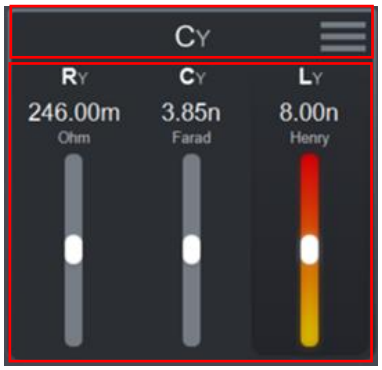


Abbildung 16: compBoxHeader und compBoxBody

Die CompBox beinhaltet 2 Elemente fürs Layout und ordnet diese vertikal, also untereinander an: compBoxHeader und compBoxBody. Diese sind in der

Abbildung 16 rot umrahmt dargestellt.

CompBoxHeader beinhaltet in der Mitte ein Label für den Namen der Komponente und einen JFXHamburger auf der rechten Seite, mit dessen die Darstellung des unteren Bereichs verändert werden kann.

Beim Aufstarten des Programms befindet sich im compBoxBody eine HBox, die alle parasitären Parameter in der Slidervariante, gezeigt in Abbildung 17, horizontal auflistet.

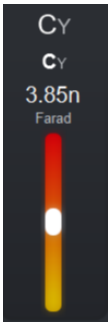


Abbildung 18: SubCompSliderBox

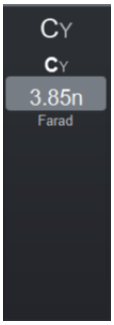


Abbildung 18: SubCompSettingBox

Mit Click auf den JFXHamburger wird diese HBox nun mit einer anderen HBox ausgetauscht, die die parasitären Parameter nun in der Textfeldvariante, zu sehen in Abbildung 18, horizontal auflistet.

3.3 Controller

In diesem Abschnitt wird beschrieben, wie der Controller als Schnittstelle zwischen der View und dem Model fungiert.

Die View besitzt eine Referenz auf den Controller. Ausserdem sind alle Bedienelemente der View (Schieberegler, Textfelder, MenuItems etc.) mit einem geeigneten Listener ausgestattet, um bei Veränderung die entsprechende Funktion des Controllers aufrufen zu können. Die wichtigsten dieser Funktionen werden nun nach Zweck geordnet in Tabelle 1 aufgelistet. Wann diese Funktionen aufgerufen werden und was anschliessend passiert, kann man den Spuren im Anhang C entnehmen.

Zweck	Methode	Beschreibung	Spur Nr. (siehe Anhang C)
Neuberechnung & Darstellung	*setMainLine()*	Wenn ein Schieberegler bewegt wird und sich der Wert des parasitären Parameters ändert, muss nur die Hauptlinie (weiss) neu berechnet und gezeichnet werden.	2
	showSweepLines (Component subComponent)	Sollen die Auswirkungen eines parasitären Parameters angezeigt werden, müssen alle farbigen Kurven (gelb-rot) neu berechnet und gezeichnet werden.	3
Nur Darstellung	*changeAppearanceOf (CompBox compBox)*	Wird auf den Hamburger innerhalb der CompBox geklickt, soll die Darstellung der CompBox zwischen Slider- und Textfeldvariante wechseln.	5
	toggleSizeOf(VBox vbox)	Bei Doppelklick auf GraphBox oder SchemaBox, wird die Breite des Elements von halber auf ganze Fensterbreite umgeschaltet und umgekehrt.	6
	createCompBoxesFromCircuit (Circuit cicuit)	Je nach Zusammenstellung des Circuits generiert der Controller die CompBox Objekte, die dann der CircuitPane der View übergeben und dort dargestellt werden.	1 und 8
Andere	*writeCircuitToFile(File file)* *readFile(File file)*	Mit diesen Beiden Funktionen kann die Schaltung zusammen mit allen Einstellungen der CustomMenuBar, Schiebereglern und Textfeldern in einem JSON-File abgespeichert oder geladen werden.	7 und 8

Tabelle 1: Erklärung der Methoden im Controller

3.4 Model

Das Model besitzt Funktionen zur Berechnung der Einfügungsverluste und speichert die Resultate in privaten Variablen ab. Das Model verfügt auch über eine Referenz auf das Circuit. So kann das Model auf die zur Berechnung nötigen Impedanzen der einzelnen Komponenten und die Matrizen der Zweitore einfach via ***Circuit.TwoPort.Component.getImpedance()*** und ***Circuit.TwoPort.getAMatrix()*** zugreifen.

Die Funktionen ***calculateCM()*** und ***calculateDM()*** bilden das Herzstück der Berechnung. Diese liefern mit Hilfe von ***getImpedanceArray()*** und ***getInsertionLoss()*** die Einfügungsverluste als Array.

SetMainLineData() und ***addSweepLinesData()*** lösen die Berechnungsfunktionen aus und speichern die Resultate in den privaten Variablen **allLineDataCM/DM** ab. Im ersten Fall nur ein Mal (für die Berechnung der schwarzen Linie), im zweiten zwanzig Mal (für die Berechnung der grünblauen Linien).

Damit die View die privaten Variablen **allLineDataCM/DM** zum Zeichnen verwenden kann, stehen die Funktionen ***getListCM()*** und ***getListDM()*** innerhalb des Models zur Verfügung.

Der genaue Ablauf der Berechnungsfunktionen ist in der Spur 9 ersichtlich.

3.5 Lizenzierung:

Das Programm wurde mit dem Framework JavaFX geschrieben, welches man auf der offiziellen Webseite www.oracle.com herunterladen kann. Sowohl JavaFX als auch Java SE unterliegen, nebst der Oracle Binary Code License, der GNU GPL (GNU General Public License). Diese besagt, dass Änderungen oder Ableitungen von GPL-lizenzierten Werken nur unter diesen gleichen Lizenzbedingungen vertrieben werden dürfen. Des Weiteren darf GPL-lizenzierte Software sowohl für kommerzielle als auch private Zwecke ausgeführt werden. Bei Vertrieb oder Weitergabe muss, im Gegensatz zur privaten oder internen Nutzung, der Quellcode veröffentlicht werden. Nebst JavaFX kommt im Programm auch noch die Library JFoenix zur Anwendung. [3], [4], [5]

3.6 Validierung der Software

Damit sich das Projektteam vergewissern kann, ob alle Ziele erreicht wurden, wurde eine Validierung durchgeführt. Im Pflichtenheft wird beschrieben, dass die Software alle im Auftrag beschriebenen Anforderungen erfüllen soll, dazu ist regelmässiger Kontakt mit dem Auftraggeber gepflegt worden. In Absprache mit dem Auftraggeber funktioniert die Software nun wie im Auftrag beschrieben.

Das Verhalten der Software bei fehlerhaften Eingaben wurden intern im Team geprüft. Dafür wurden Fehleingaben gemacht und getestet wie die Software darauf reagiert. Daraufhin wurde die Software so geändert, dass Fehleingaben zu keinen Komplikationen führen werden.

Für die Benutzerfreundlichkeit schien eine Prüfung durch Dritte die sinnvollste Validierung. Um dies zu ermöglichen, wurde die Software mehreren Aussenstehenden vorgeführt. Dabei handelte es sich sowohl um Ingenieure als auch um Laien. Um die Zufriedenheit der Testenden auszuwerten wurden sie nach der Vorführung befragt und gebeten eine Umfrage auszufüllen. Dabei zeichnete sich ab das Laien Mühe mit den Abkürzungen hatten. Laien und Ingenieure fänden eine Einführung hilfreich. Aufgrund dieses Feedbacks wurde eine Einführung im Hilfe-Menü erstellt.

Zusätzlich sollte das Programm in der Lage sein die Betriebssystemen MacOS und Windows zu unterstützen. Dies wurde Team intern geprüft, da im Team Computern beider Betriebssysteme vertreten sind und das Programm funktioniert auf beiden.

4 Elektrotechnik

Für die Berechnungen sind die Schaltungen mit den parasitären Elementen von Interesse.

Nachfolgend hat man für das Grundschema des Filters in Abbildung 19, die parasitären Parameter gemäss Ersatzschema in Abbildung 20 für die CM-Schaltung eingesetzt. Interessant dabei ist der 1MΩ Widerstand, welcher parallel zu CX2 liegt. Dieser wird aus Sicherheitsgründen benötigt, um den CX2 Kondensator zu entladen. In der Funktion des Filters hat er jedoch keinen Einfluss, weshalb er nicht in die Simulationen und Berechnungen einfliesst.

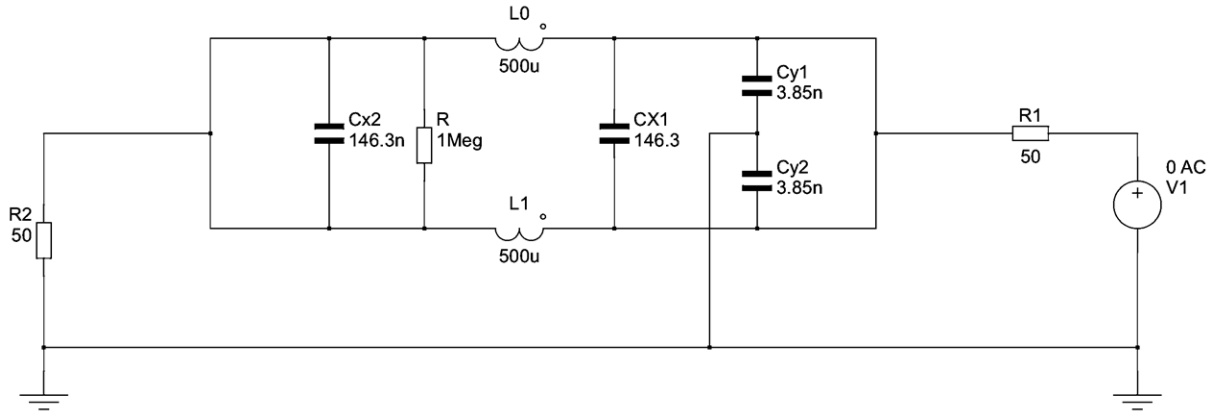


Abbildung 19: CM-Ersatzschaltung ohne parasitäre Parameter

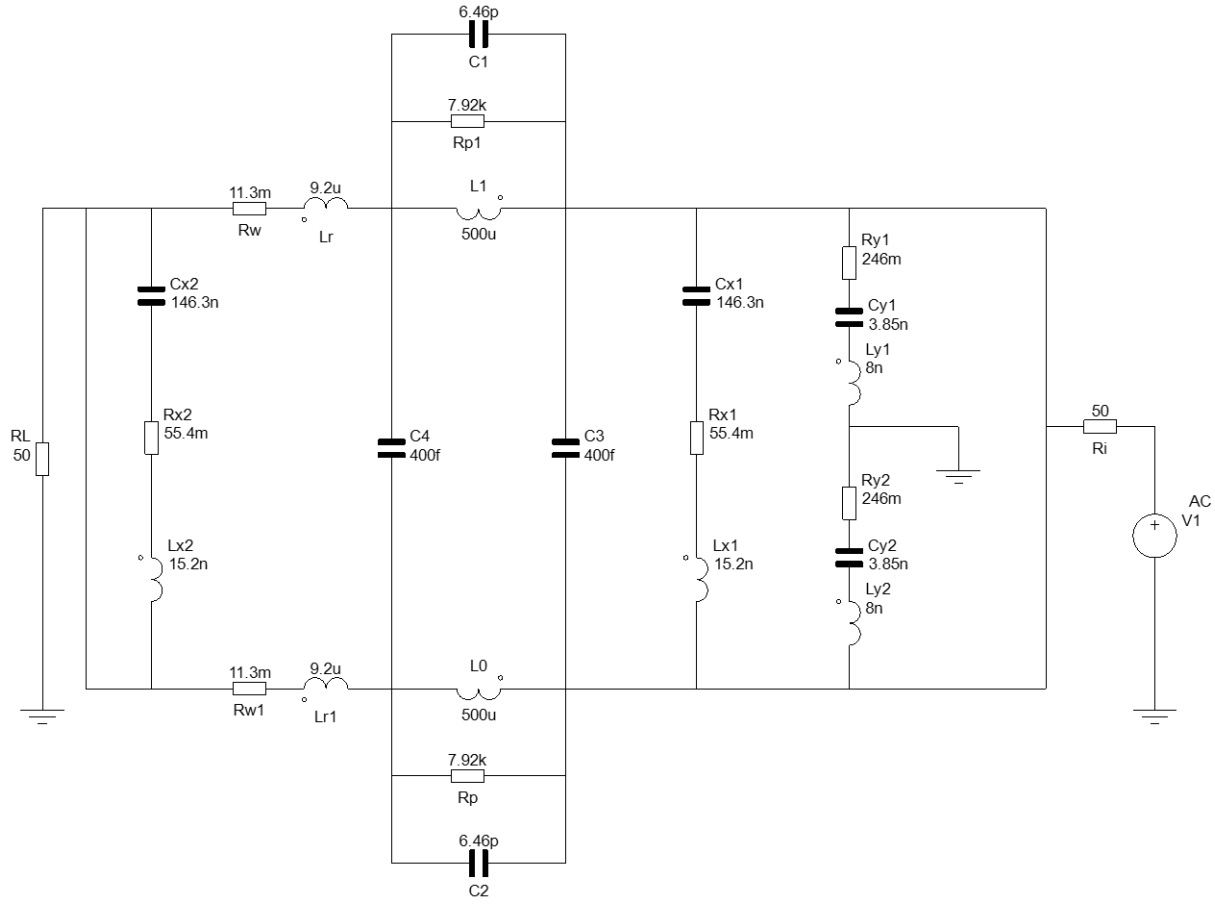


Abbildung 20: CM-Ersatzschaltung mit parasitären Parametern

Die DM-Grundschaltung in Abbildung 21 hat man ebenfalls mit parasitären Parametern ergänzt. Der 1MΩ Widerstand fällt ebenso wie bei der CM-Schaltung weg. Diese ergänzte Schaltung ist in der Abbildung 22 ersichtlich.

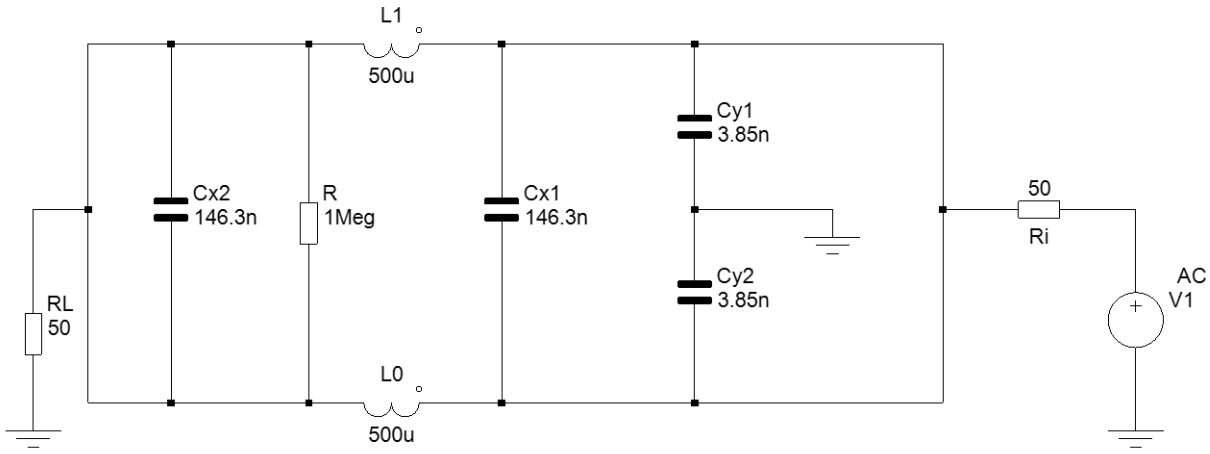


Abbildung 21: DM-Ersatzschaltung ohne parasitäre Parameter

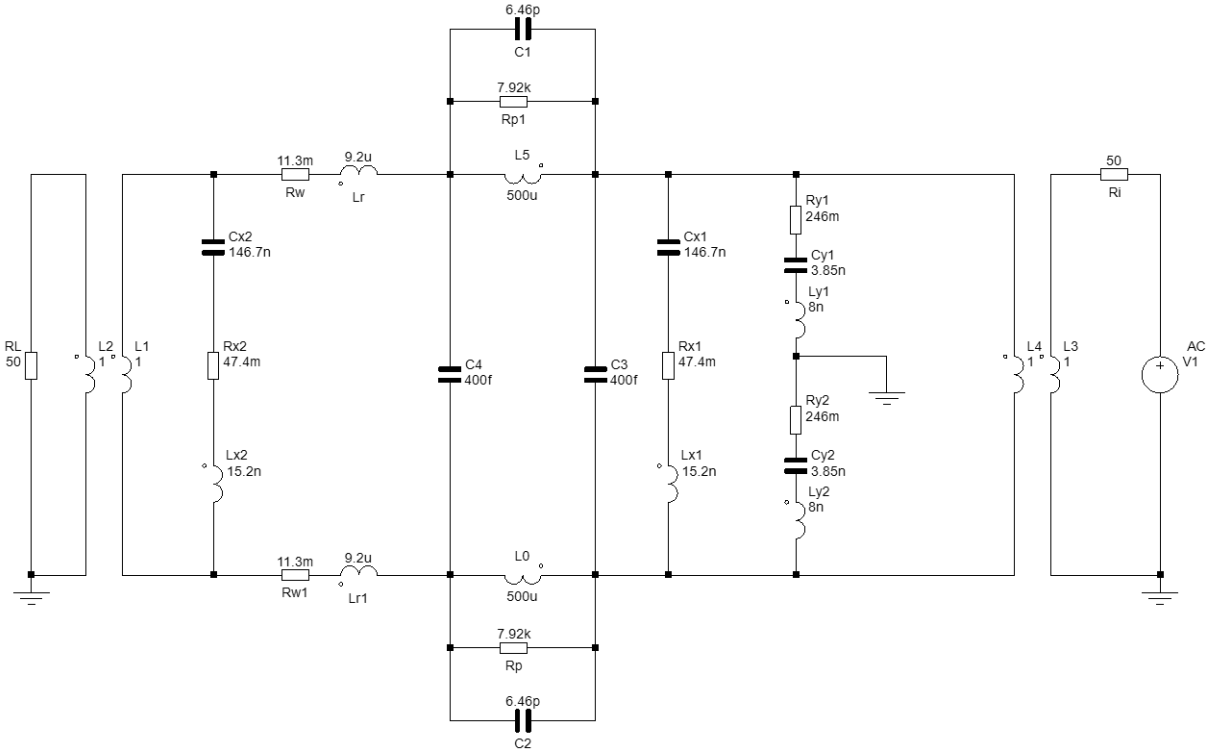


Abbildung 22: DM-Ersatzschaltung mit parasitären Parametern

4.1 Vereinfachung der CM-Schaltung

Die Schaltungen mit den parasitären Elementen wurden danach gemäss den Regeln der Elektrotechnik vereinfacht, so dass Zweitore gebildet werden können. Daraus berechnen sich die S-Parameter. Mit diesen können schliesslich die Einfügungsdämpfung berechnet werden. Nachfolgend werden alle Schritte erklärt.

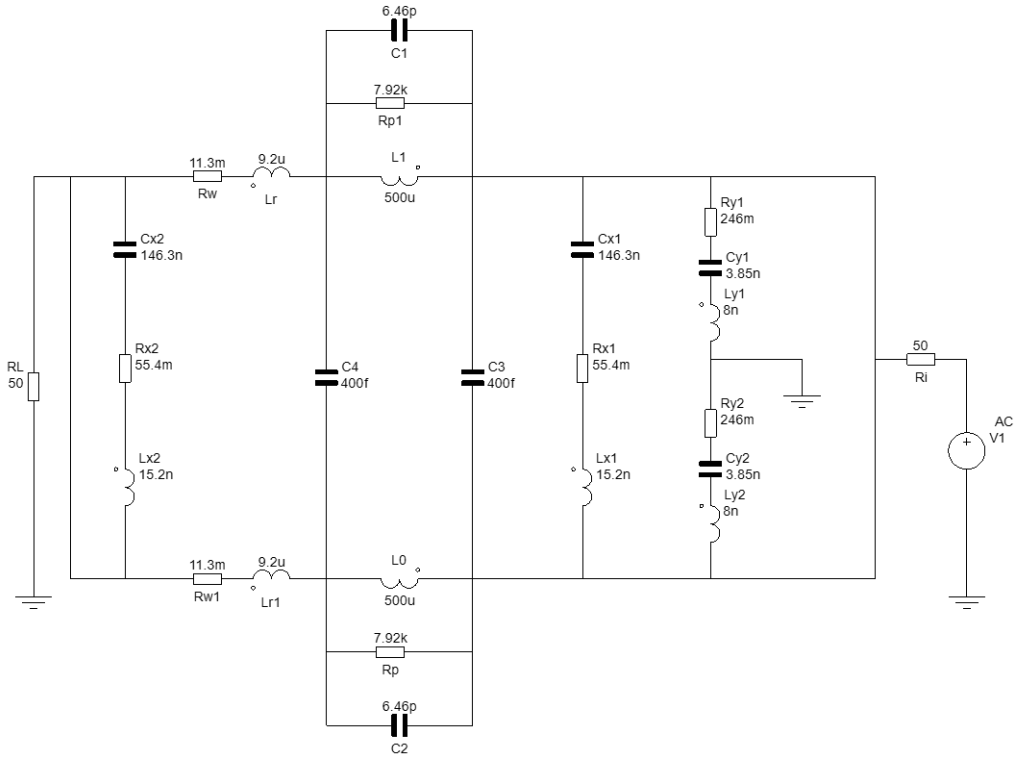


Abbildung 23: CM Grundschaltung

Für die Vereinfachung der Schaltung gehen wir von der Grundschaltung in Abbildung 23 aus. Die CM-Störungen auch symmetrische Störungen genannt, fliessen in beiden Leitern in die gleiche Richtung. Das heisst die beiden Leiter liegen auf demselben Potential, somit haben die Bauteile zwischen den beiden Leitern (CX1, CX2, RX1, RX2, LX1, LX2, C4, C3) keinen Einfluss. Nun können die beiden Leiter zusammengefasst werden, da beide gegenüber Masse symmetrisch sind. Das Zusammenfassen der beiden Leiter entspricht einer Parallelschaltung der beiden Leiter. Deshalb ergeben sich folgende Faktoren für die vereinfachte Schaltung: Der parasitäre Widerstand Rw und die parasitäre Spule Lr halbieren ihre Werte, genau wie der Widerstand Rp1. Durch die Parallelschaltung der Kondensatoren C1 und C2 addieren sich bekanntlich die beiden Kapazitäten zu einer Gesamtkapazität C1 + C2. Die beiden Spulen L0 und L1 sind auf einem Ferritkern aufgewickelt. Da beide Spulen den gleichen Wicklungssinn besitzen, verdoppelt sich die Induktivität L0. Durch das Zusammenfassen der beiden Spulen halbiert sich ihr Wert wieder, woraus der ursprüngliche Wert der Spule resultiert. Die beiden Kondensatoren CY1 und CY2 mit ihren parasitären Elementen bleiben unbeeinflusst, da sie bereits gegen Masse geschalten sind. Nach diesen Vereinfachungen ergibt sich die Schaltung in Abbildung 24.

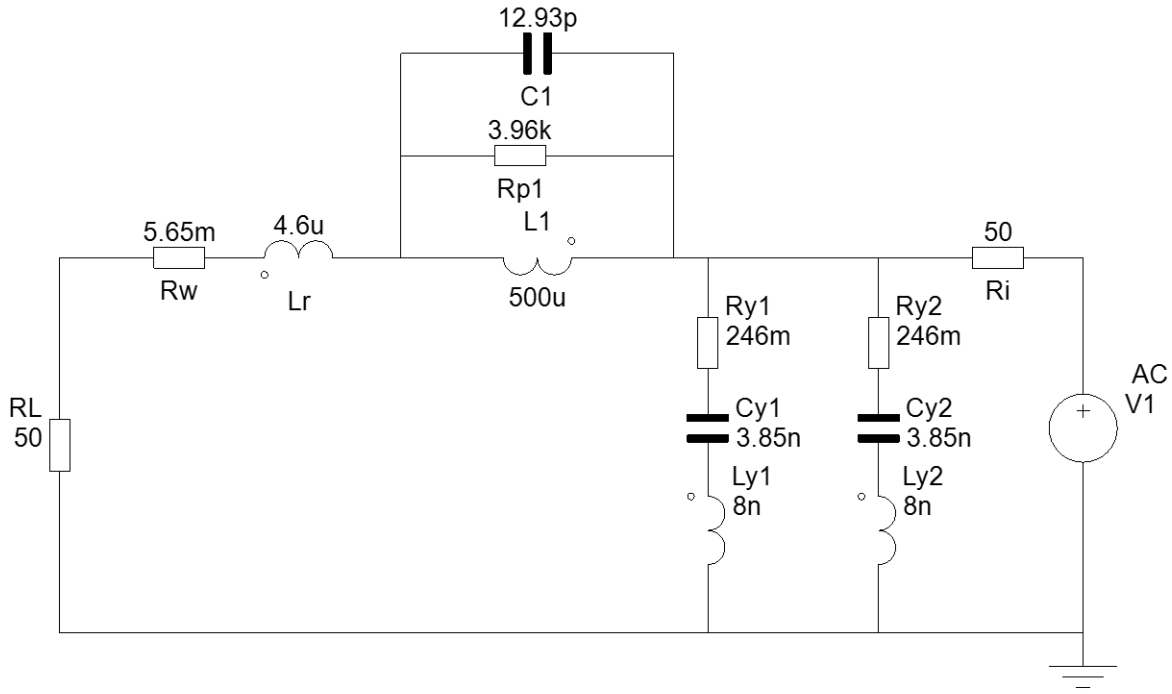


Abbildung 24: Vereinfachte CM Schaltung

Die Validierung der Umformungen wurde mit zwei Simulationen in «MPLAB» bewerkstelligt. Im folgenden Diagramm (Abbildung 25) liegen die beiden Ergebnisse der Simulationen übereinander. Darin ist sichtbar, dass die Vereinfachung zum selben Ergebnis führt wie die Grundschaltung. Die beiden simulierten Schaltungen dazu sind im Anhang 0 hinterlegt.

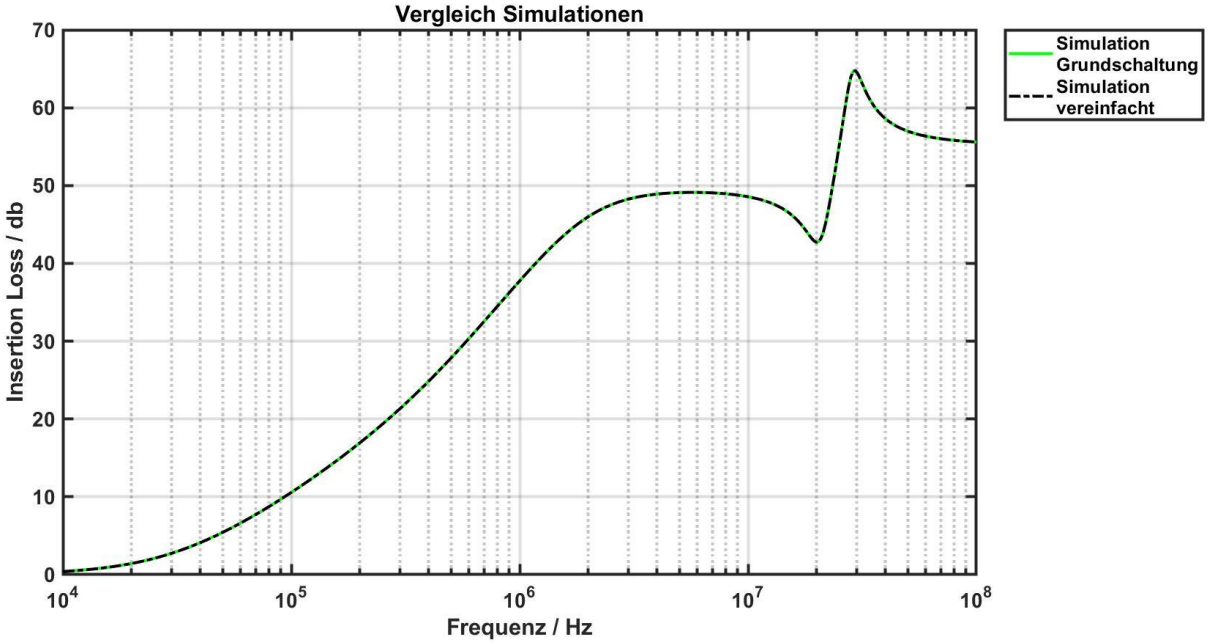


Abbildung 25: Vergleich Grundschaltung / Vereinfachte Schaltung

4.2 Vereinfachung der DM-Schaltung

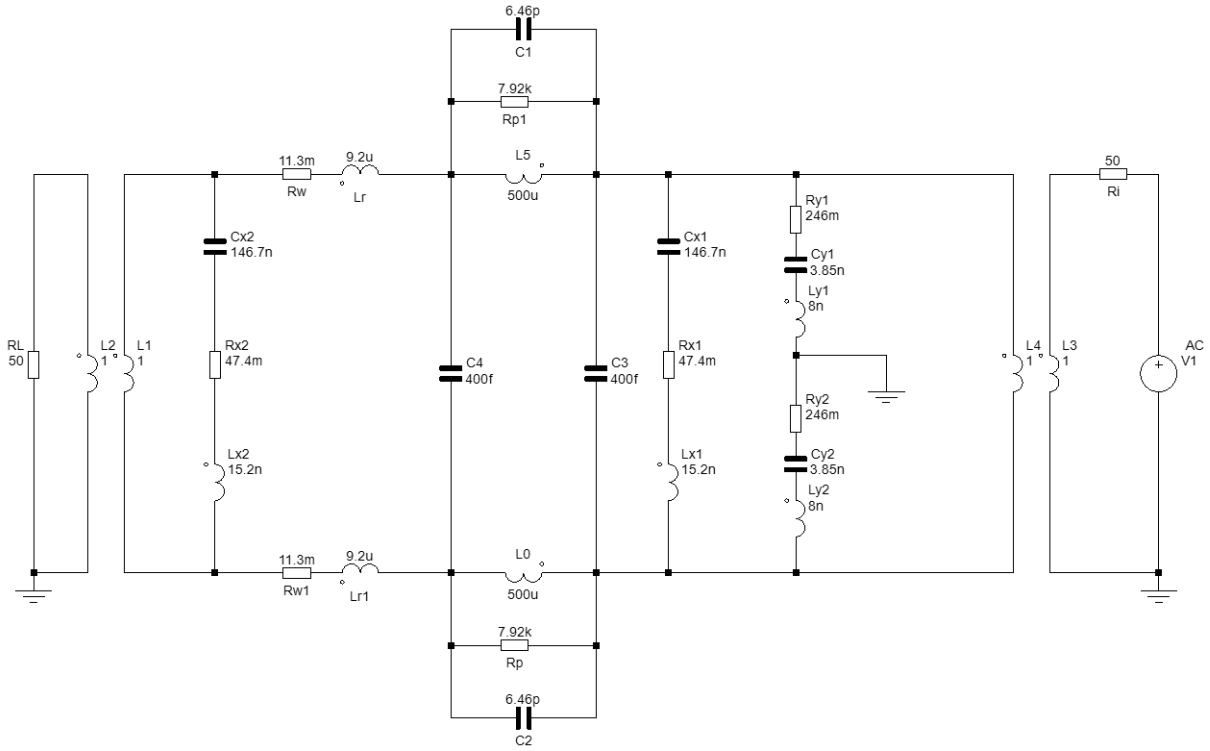


Abbildung 26:Grundschaltung DM

Die Grundschaltung der DM Schaltung ist in der Abbildung 26 sichtbar. Die DM-Störungen, auch asymmetrische Störungen genannt, fliessen in einem Leiter hin und im anderen Leiter wieder zurück. Daher lassen sich die Bauelemente nicht so leicht vernachlässigen wie in der CM Schaltung. Der Ringkern mit den beiden Spulen L0 und L5 gekoppelt sind, hat in der DM-Schaltung keinen Einfluss, da sich die beiden Induktivitäten aufgrund des entgegengesetzten Feldes fast vollständig aufheben. Dadurch heben sich auch die parasitären Elemente (C1, C2, Rp, Rp1) auf. Es bleiben nur noch die die parasitären Parameter Rw und Lr der Spulen übrig. Damit wir wie auch schon in der CM Schaltung die beiden Leiter zusammenfassen können, müssen wir die Bauteile zwischen den beiden Leitern verdoppeln. Dafür werden die Widerstände und Spulen halbiert und die Kondensatoren verdoppelt. Dies betrifft auch den Innen- und den Lastwiderstand. Dabei wird die Funktion der Schaltung nicht verändert. Die 1:1 Trafos werden dabei vernachlässigt, da diese als ideal angenommen werden und im Filter nicht als Bauteil vorkommen. Die Kondensatoren C3 und C4 werden weggelassen, da sie einen vernachlässigbaren Einfluss auf die Schaltung haben. Dies wurde vom Auftraggeber akzeptiert. In Abbildung 27 sieht man die vereinfachte Schaltung.

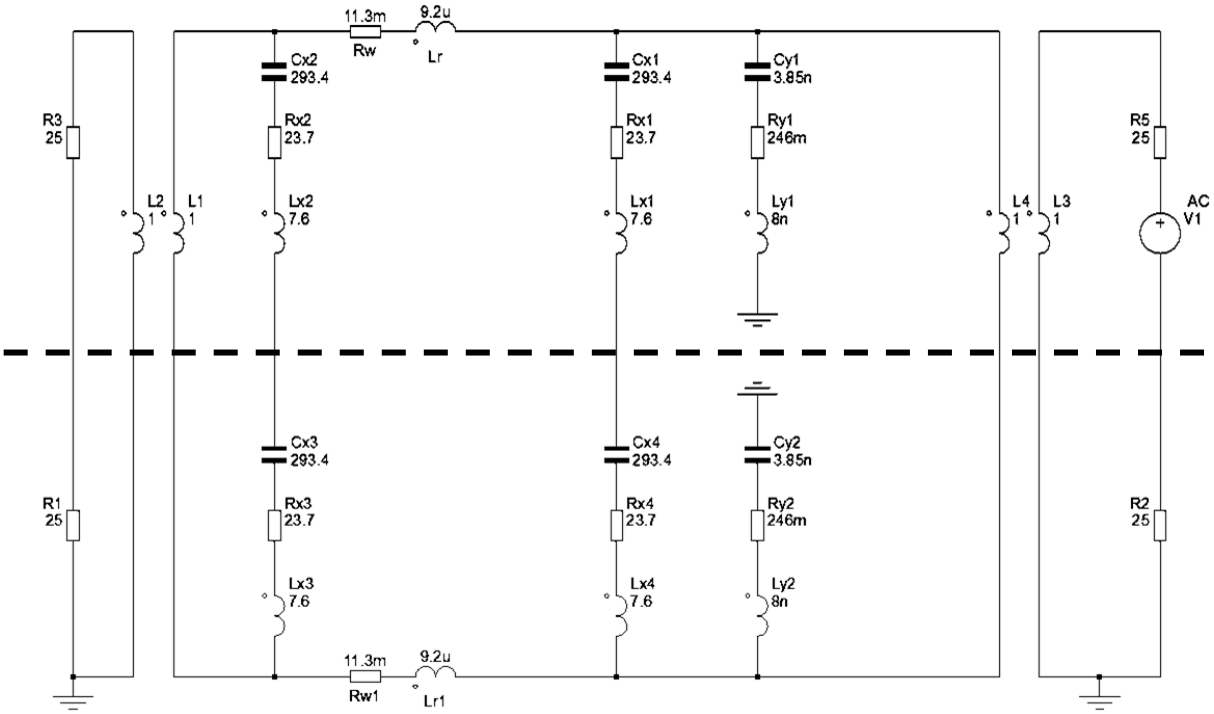


Abbildung 27: DM Schaltung angepasst

Bei der schwarz gestrichelt eingezeichneten Achse, wurde die Schaltung aufgetrennt und gegen Masse gelegt (Abbildung 27). Dies ist möglich, da die gleichen Störungen in Hin- und Rückleiter fliessen und man diese halbieren kann. Die parasitären Bauteile der Spule (Lr und Rw) werden zusammengefasst, das heisst sie werden beide halbiert. Einer der beiden Y-Kondensatoren fällt weg. Dabei spielt es keine Rolle welcher, da die beiden immer gleichgross sind. Deshalb muss man annehmen, dass die parasitären Werte davon gleichgross sind. In der Software kann man deshalb auch nur einen Y-Kondensatorwert verstellen. Die vereinfachte Schaltung, welche zur Berechnung verwendet wurde, ist in Abbildung 28 sichtbar.

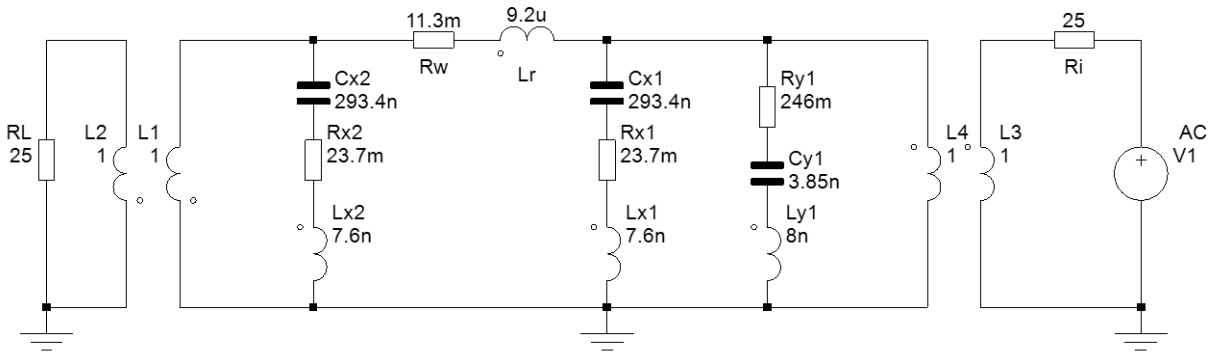


Abbildung 28: Vereinfachte DM Schaltung

Die Validierung der Umformungen wurde mit zwei Simulationen in «MPLAB» bewerkstelligt. Im folgenden Diagramm (Abbildung 29) liegen die beiden Ergebnisse der Simulationen übereinander. Darin ist sichtbar, dass die Vereinfachung zum selben Ergebnis führt wie die Grundschaltung. Die beiden simulierten Schaltungen dazu sind im Anhang 0 hinterlegt.

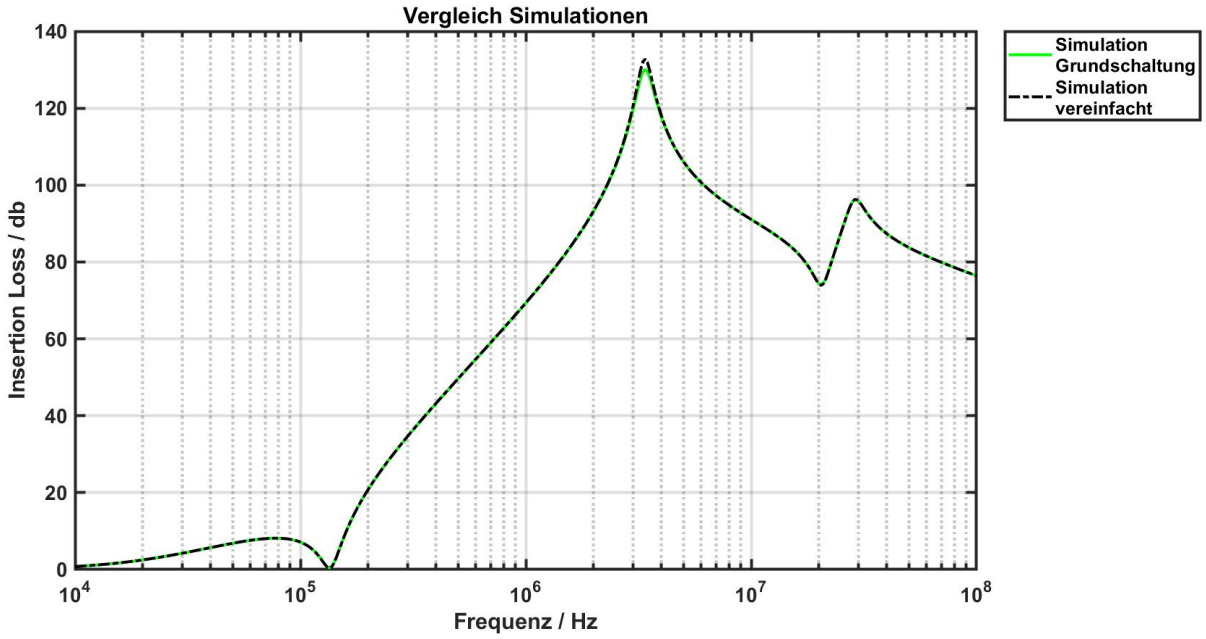


Abbildung 29: Vergleich Simulationen

4.3 Vorgehensweise Berechnung Einfügungsverluste

Wir befassen uns einfachheitshalber mit der CM Schaltung, um das Prinzip der Berechnungen zu erklären. Bei der DM Schaltung wird wiederum genau die gleiche Vorgehensweise angewendet. Jedoch werden die Zweitore anders aufgeteilt. In Abbildung 30 wurde die CM Schaltung in Zweitore unterteilt und beschriftet.

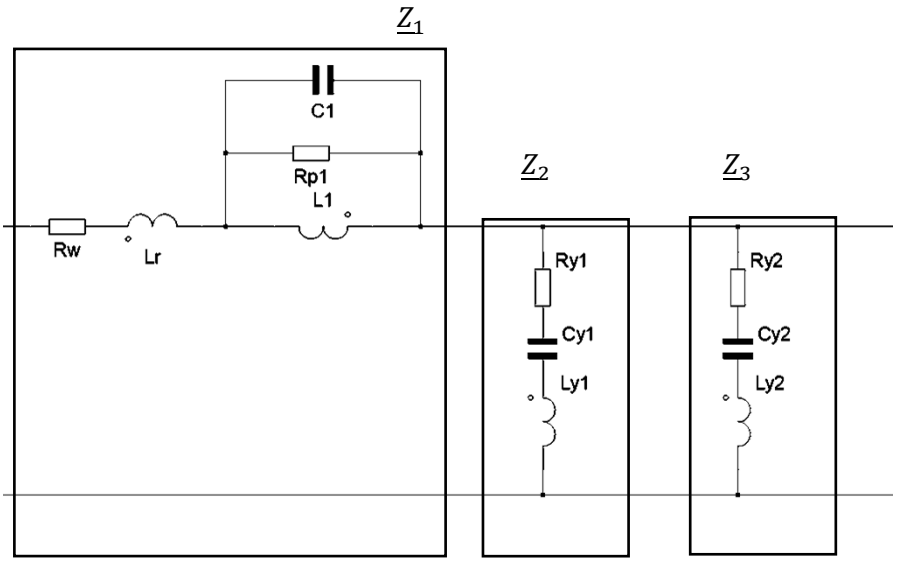
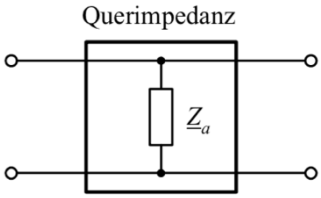
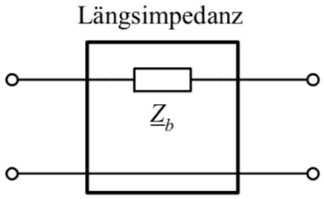


Abbildung 30: Schaltung CM mit Einteilung in Zweitore

Nachdem die Impedanzen in Zweitore zusammengefasst wurden, kann man diese in Kettenmatrizen umwandeln. Dies funktioniert für die unterschiedlichen Zweitore folgendermassen:



$$AQ = \begin{bmatrix} 1 & 0 \\ 1/\underline{Z}_a & 1 \end{bmatrix}$$



$$AL = \begin{bmatrix} 1 & \underline{Z}_b \\ 0 & 1 \end{bmatrix}$$

Man berechnet nun die Kettenmatrix (A1) für die Längsimpedanz ($\underline{Z}1$) sowie (A2) für die Querimpedanz ($\underline{Z}2$) und (A3) für ($\underline{Z}3$).

$$A1 = \begin{bmatrix} 1 & \underline{Z}_1 \\ 0 & 1 \end{bmatrix}$$

$$A2 = \begin{bmatrix} 1 & 0 \\ 1/\underline{Z}_2 & 1 \end{bmatrix}$$

$$A3 = \begin{bmatrix} 1 & 0 \\ 1/\underline{Z}_3 & 1 \end{bmatrix}$$

Für den S21-Parameter benötigen wir die Gesamtmatrix, diese berechnet man mit Hilfe der Kettenschaltung, in dem man die drei Matrizen miteinander multipliziert. Dabei ist es wichtig, dass zuerst A1 mit A2 multipliziert wird und danach mit A3. [6]

$$A = A1\ A2\ A3 = \begin{bmatrix} 1 & \underline{Z}_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1/\underline{Z}_2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1/\underline{Z}_3 & 1 \end{bmatrix}$$

4.4 S-Parameter

Die Streuparameter, abgekürzt S-Parameter genannt, benötigt man um elektrische Komponenten wie Tore genauer zu beschreiben. Sie sind frequenzabhängig und werden als komplexe Zahl angegeben. Dabei sind die benötigten Parameter quadratisch von den Anzahl Toren abhängig. Bei einem Eintor reicht ein Streuparameter, bei einem Zweitor benötigt man vier, bei einem Dreitor sind neun nötig und so weiter.

Die jeweiligen S-Paramter werden mithilfe der Wellengrössen der Tore bestimmt. Genauer, durch die einlaufenden Wellen $\underline{\hat{a}}_v$ und die vom Tor reflektierten Wellen $\underline{\hat{b}}_v$. In der Abbildung 31 sind die Wellen an einem Zweitor eingezeichnet. Nachfolgend wird beschrieben, in welchem Zusammenhang, die einfliessenden- und die reflektierten Wellen mit den vier S-Parametern eines Zweitores stehen.

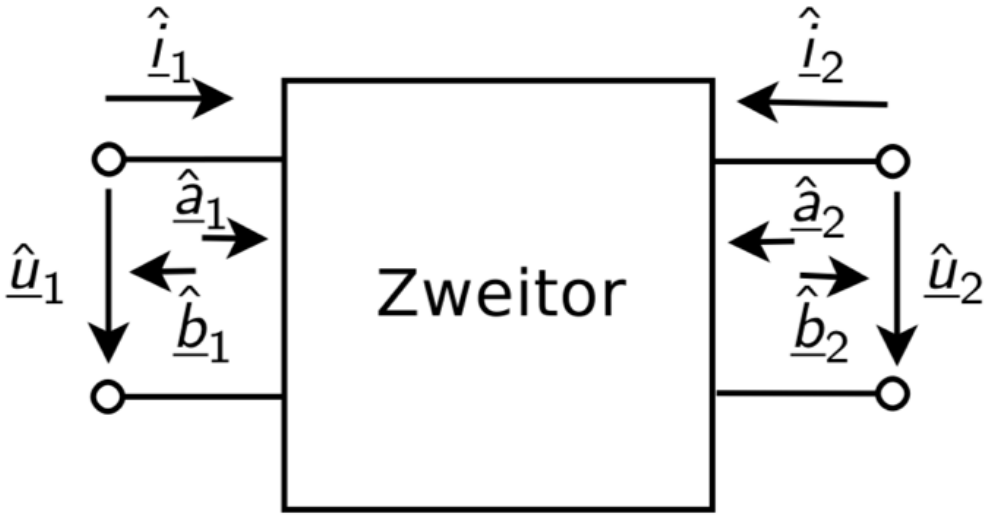


Abbildung 31: Zwei Tor mit Wellengrössen

Wobei $\underline{\hat{a}}_1$ und $\underline{\hat{a}}_2$ die Amplituden der einfliessenden Wellen sind.

Die Grössen $\underline{\hat{b}}_1$ und $\underline{\hat{b}}_2$ sind hingegen die Amplituden der reflektierten Wellen.

Daraus lässt sich die Streumatrix S wie folgt berechnen:

$$\begin{bmatrix} \underline{\hat{b}}_1 \\ \underline{\hat{b}}_2 \end{bmatrix} = \begin{bmatrix} \underline{S}_{11} & \underline{S}_{12} \\ \underline{S}_{21} & \underline{S}_{22} \end{bmatrix} \begin{bmatrix} \underline{\hat{a}}_1 \\ \underline{\hat{a}}_2 \end{bmatrix}$$

Die einzelnen Parameter haben dabei folgende Bedeutung [7]:

$\underline{S}_{11}$: Eingangsreflexionsfaktor

$\underline{S}_{12}$: Rückwärtstransmissionsfaktor

$\underline{S}_{21}$: Vorwärtstransmissionsfaktor

$\underline{S}_{22}$: Ausgangsreflexionsfaktor

Für die Einfügungsverluste benötigt man den Vorwärtstransmissionsfaktor. Dieser ist abhängig von den Bezugswiderständen (R1 und R2), welche am Eingang, sowie am der Last 50 Ω betragen. Daraus berechnet man den Vorwärtstransmissionsfaktor folgendermassen [8]:

$$\underline{S}_{21} = \frac{2}{A_{11}\sqrt{\frac{R_2}{R_1}} + \frac{A_{12}}{\sqrt{R_1 R_2}} + A_{21}\sqrt{R_1 R_2} + A_{22}\sqrt{\frac{R_1}{R_2}}}$$

$A_{(\text{Zeile,Spalte})}$: Wert der Gesamtmatrix an bestimmter Position.

R_1: Eingangswiderstand

R_2: Lastwiderstand

Damit werden schlussendlich die Einfügungsverluste a in dB berechnet:

$$a_I = -20 \log|S_{21}|$$

4.5 Realisierung mit Matlab

Bevor die Berechnungen in Java implementiert wurden, sind die Einfügungsverluste in Matlab berechnet worden. Dies wurde gemäss der Vorgehensweise in 4.3 bewerkstelligt. [9] [10]

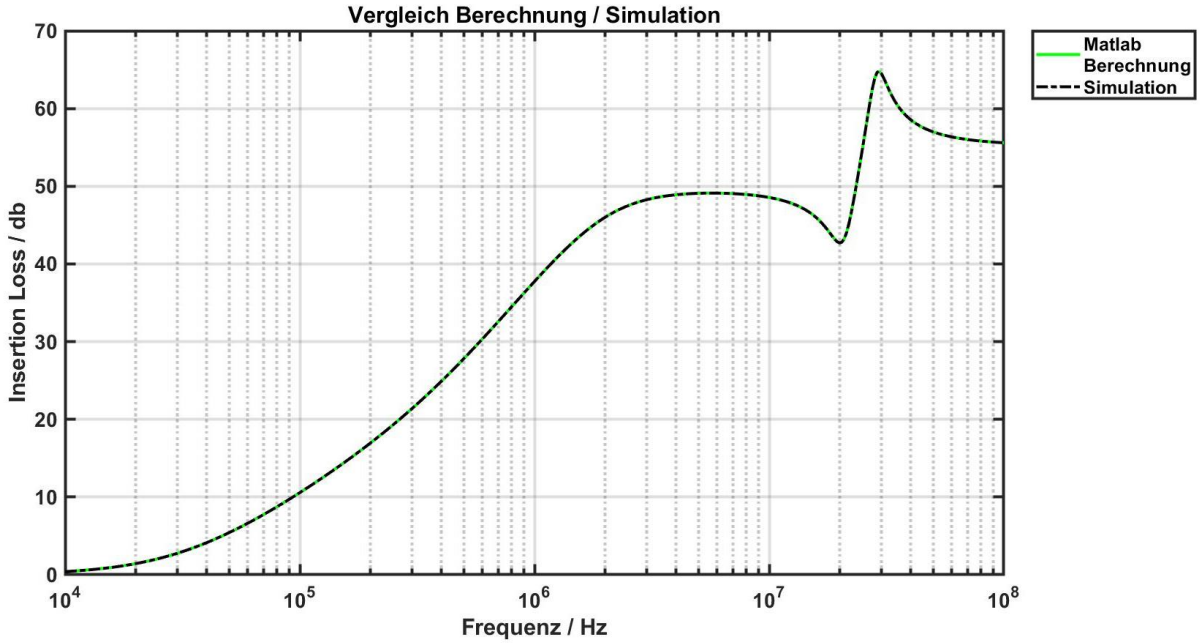


Abbildung 32: Vergleich Berechnung / Simulation

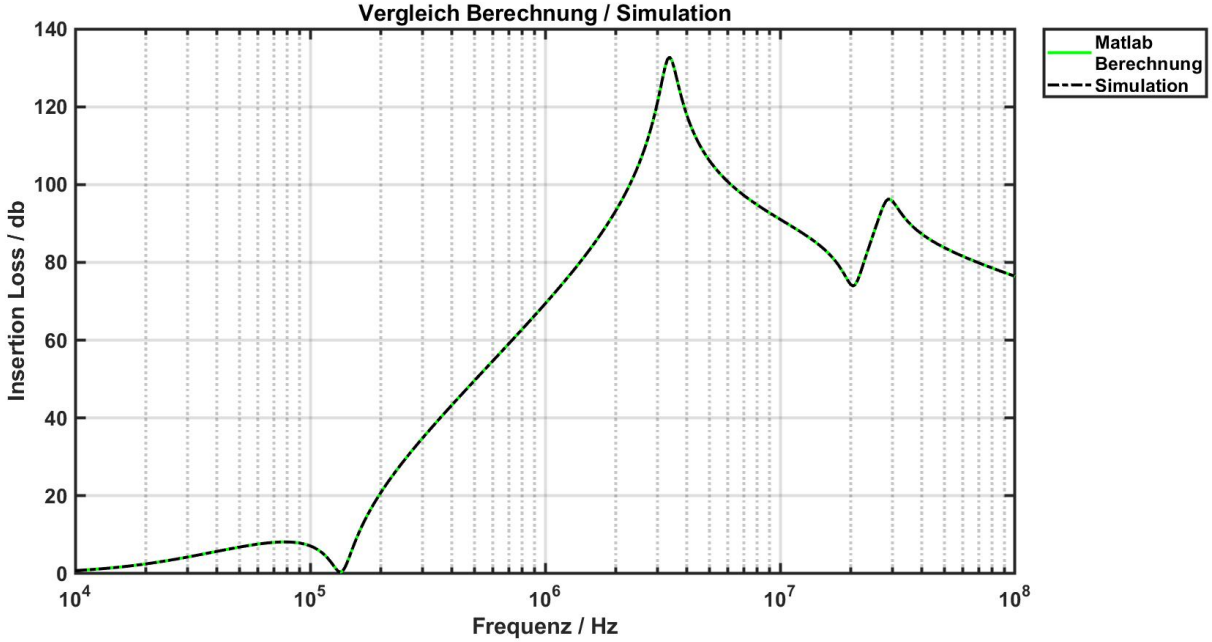


Abbildung 33: Insertion loss Kurve aus MATLAB

Der Vergleich der beiden Kurven (Abbildung 32, Abbildung 33) zeigt, dass die Berechnungen mit der Simulation übereinstimmten. Der Matlab Code für die Berechnungen ist im Anhang B vorhanden.

4.6 Validierung der Elektrotechnik

Die Implementation der Berechnung ist im Softwareteil dokumentiert. Um zu überprüfen, ob die Berechnungen in Java mit den Berechnungen in Matlab übereinstimmen, wurden die beiden Kurven (Abbildung 34, Abbildung 35) wiederum übereinandergelegt und verglichen. Dabei geht hervor, dass die Berechnungen korrekt in die Software implementiert wurde.

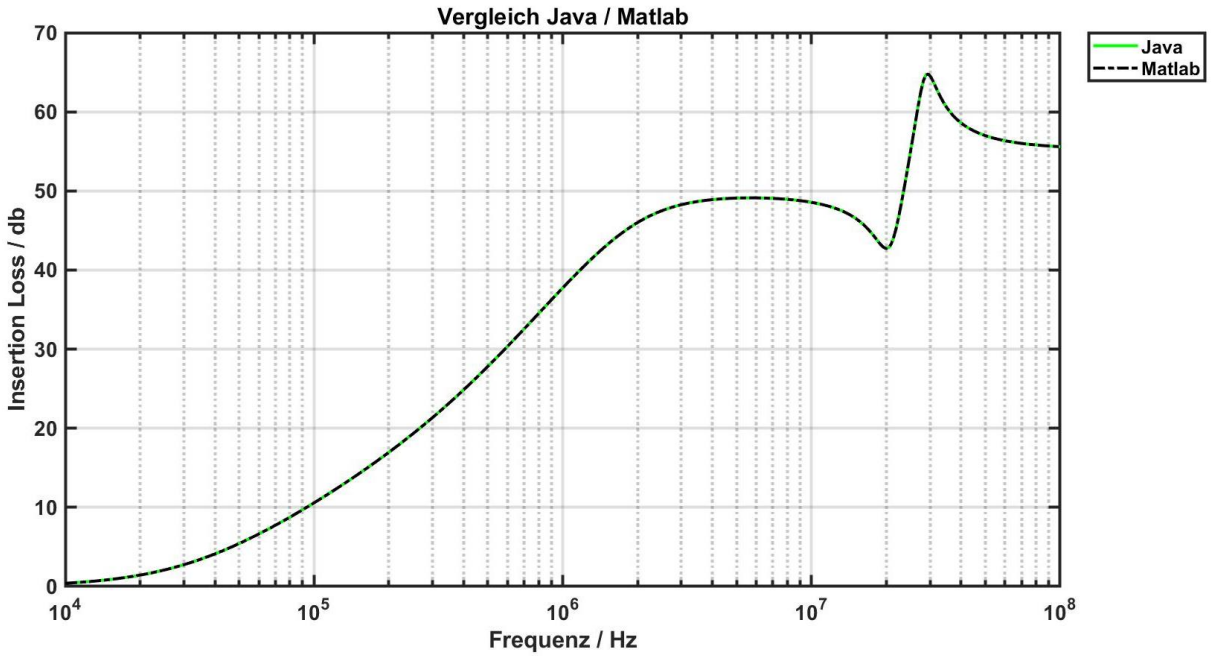


Abbildung 34: Vergleich Java / Matlab CM

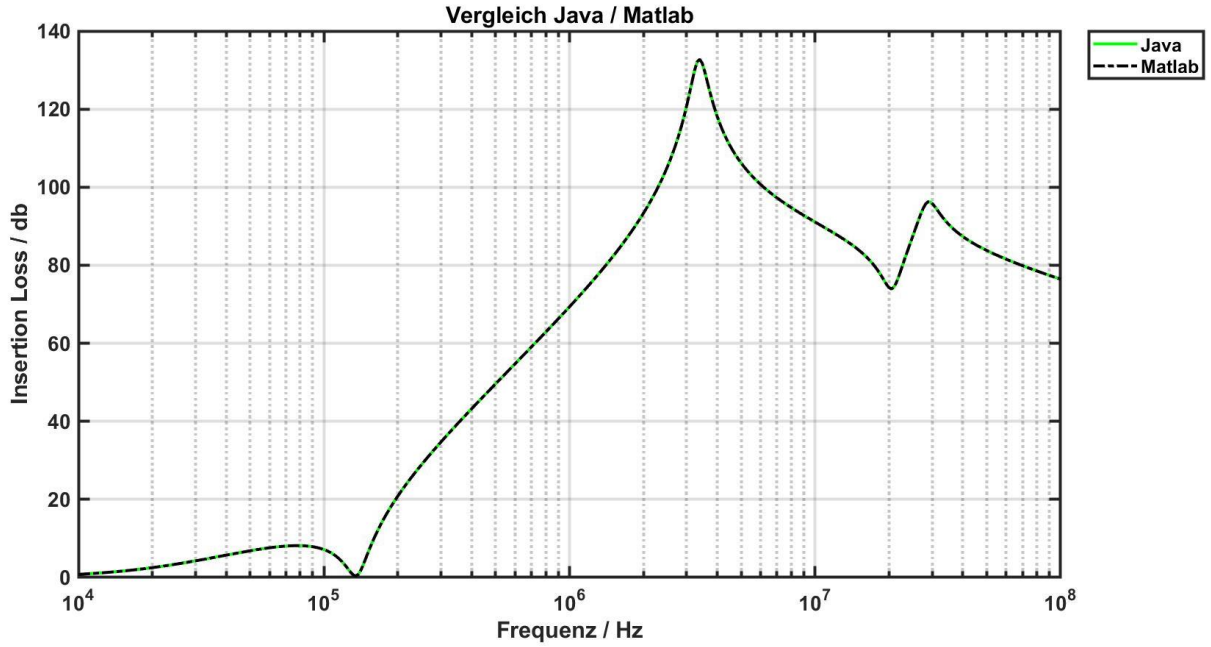


Abbildung 35: Vergleich Java / Matlab DM

5 Schlusswort

Die Software «EMI-DJ» erfüllt alle Anforderungen, welche in der Aufgabenstellung (im Anhang E) verlangt und im Pflichtenheft formuliert sind:

Die Einfügungsverluste eines EMI Filters werden für beide Störungsarten, Gleich- wie auch Gegentakt ausgerechnet und in Graphen dargestellt. Mit Schiebereglern und Textfeldern können die parasitären Parameterwerte dimensioniert werden, worauf in Kürze jeweils Neuberechnung und Neudarstellung der Einfügungsverluste folgen. Die vereinfachten Schaltbilder für die Gleich- und Gegentaktstörungen sind in einem separaten Tab einsehbar, so dass die Bauteilbezeichnung der Schieberegler dem entsprechenden Bauteil zugeordnet werden kann.

Zusätzlich zu den erwähnten Anforderungen bietet die Software folgende besondere Merkmale und Eigenschaften:

Alle Daten und Einstellungen können als JSON-File geladen oder gespeichert werden. Das Menu Sweep erlaubt, die Wertebereiche der Schieberegler von linear ±30% auf logarithmisch (von 10^{-3} bis 10^{+3}) zu verändern. Im Menu CISPR kann aus drei verschiedenen Referenzwiderstandspaaren gewählt werden. Durch die Verwendung des Frameworks JavaFX kann das Design im Falle einer Weiterentwicklung sehr einfach verändert werden.

Aus Zeitgründen konnten nicht alle Wunschziele gemäss Pflichtenheft realisiert werden:

Die Software erlaubt es noch nicht, beliebige Schaltungen zu erstellen und zu simulieren. Dies könnte bei einer Weiterentwicklung der Software noch verwirklicht werden. Zusätzlich könnte die Permeabilität der Spulen mit in die Berechnungen einbezogen werden und so dem Nutzenden noch aussagekräftigere Vorhersagen liefern.

Die zum Kurven zeichnen verwendete ChartPane manipuliert beim Neuzeichnen immer den SceneGraph, was grossen Rechenaufwand mit sich zieht. Dies könnte verbessert werden, indem das Zeichnen der Diagramme mit dem Canvas Objekt realisieren würde.

Das Projektteam konnte sich den Herausforderungen, welche das Projekt zu Beginn mit sich brachte, stellen und das Produkt konnte fristgerecht an den Auftraggeber und an die Fachdozierenden übergeben werden.

6 Quellenverzeichnis

[1] Schaffner, «Single-Stage Filters FN 2020,» Schaffner Group, Luterbach, 2018.

[2] T. Hofer, «EMV Filter Design Theorie und Praxis,» HSI AG, Goldach, 2016.

[3] «Oracle,» [Online].
Available: https://www.oracle.com/technetwork/java/javase/terms/license/index.html. [Zugriff am 1. Juni 2019].

[4] «GNU,» [Online]. Available: https://www.gnu.org/licenses/gpl-3.0.html. [Zugriff am 1. Juni 2019].

[5] «JFOENIX,» [Online]. Available: http://www.jfoenix.com/documentation.html#License. [Zugriff am 1. Juni 2019].

[6] P. Niklaus, «Vorlesungsskript Zweitore (2-Tore),» Hochschule für Technik der Fachhochschule Nordwestschweiz (FHNW), Brugg-Windisch, 2018.

[7] A. Schaum, «Zweitore - Zusammenfassung,» Lehrstuhl für Regelungstechnik, Universität zu Kiel, 2017.

[8] Albrecht, «CISPR 17 Measurements, The truth or Everything you wanted to know about attenuation curves validity but were afraid to ask,» Schaffner EMV AG, Luterbach, 1996.

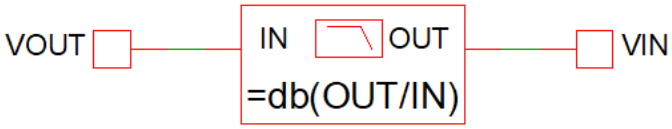
[9] A. Maier und S. Rupp, «Hochfrequenztechnik, Teil 2 - Anwendungen,» DHBW, Stuttgart, 2016.

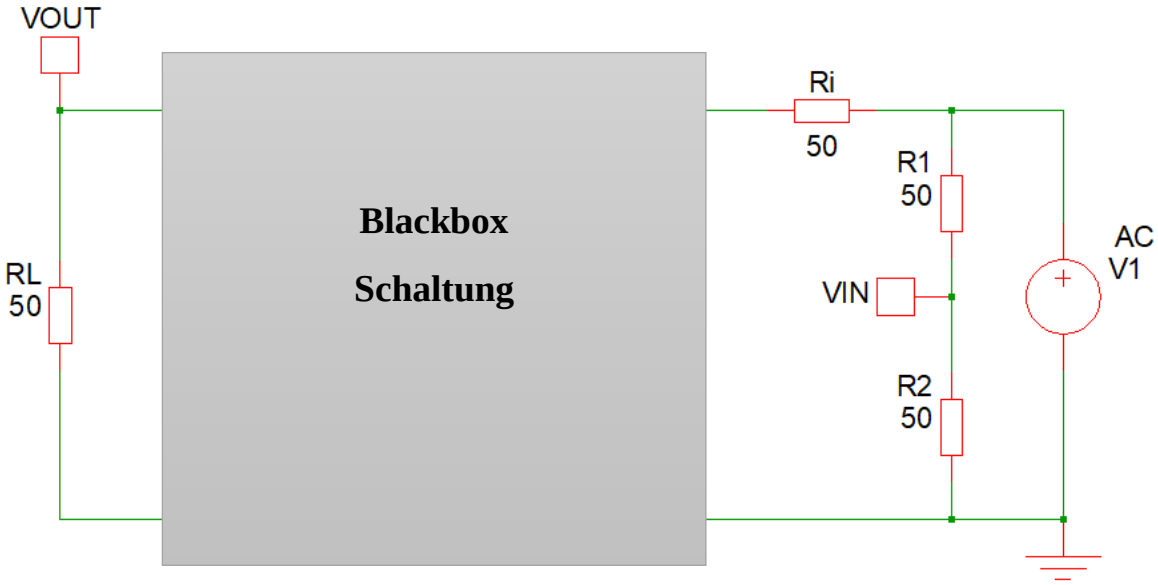
[10] B. Mößlang, «EMC Filter Insertion Loss Simulation,» OMICRON Lab, Reutlingen, 2016.

Anhang

A. Schaltungssimulationen

Für die Simulationen in MPLAB wurde immer der gleiche Schaltungsaufbau verwendet. Dabei wurde mit einer Wechselspannungsquelle (V1) eine Wechselspannung mit verschiedenen Frequenzen angelegt. In der Blackbox wurden jeweils die entsprechenden Schaltungen eingefügt. Zur Auswertung wurde ein Bodesimulator verwendet. Dieser misst am Eingang die Spannung, welche nach dem Filter über dem Lastwiderstand abfällt und am Ausgang die Spannung, welche nur über dem einen Widerstand des Spannungsteilers abfällt. Somit kann man das Ergebnis mit den Einfügungsverlusten vergleichen, welche gleich definiert sind.





B. Matlab Code

Hauptprogramm CM_DM.m

```
%*************************************************************************
% Projekt: Plot CM and DM mode
% Autor:   Luca Krummenacher
% Datum:   6. Mai 2019
% Ort :    FHNW Brugg-Windisch
% Version: 2.0
%*************************************************************************
% used functions: get_s21_dm_4.m, get_s21_CM.m,
%*************************************************************************
clear;
clc;
clf;   %clear figures
f = logspace(4,8,1000);  %Frequenzvektor von 10^4 - 10^8 Hz
for k=1 : length(f)
    H(k)  = -20 * log10(abs(get_s21_dm_4(f(1,k))));
    H2(k) = -20 * log10(abs(get_s21_CM(f(1,k))));
end

%*************************************************************************
% Plot DM
subplot(2,1,1)
semilogx(f,H,'b')   %Logartihmische X-Achse verwenden
grid on
title('Insertion Loss DM')
xlabel('Frequency / Hz')
ylabel('Insertion Loss / db')
legend("DM",'Location','bestoutside')

%*************************************************************************
% Plot CM
subplot(2,1,2)
semilogx(f,H2,'r')
grid on
title('Insertion Loss CM')
xlabel('Frequency / Hz')
ylabel('Insertion Loss / db')
legend("CM",'Location','bestoutside')
adjust_fig

%*************************************************************************
```

Funktion get_s21_dm_4.m

```
function [result] = get_s21_dm_4(frequency)
%**************************************************************************
% Projekt: s21 Parameter berechnen
%          fuer die DM-Schaltung eines EMI-Filters
% Autor:   Luca Krummenacher
% Datum:   18. April
% Ort :    FHNW Brugg-Windisch
% Version: 4.0
%**************************************************************************
% Gibt den s21 parameter fuer eine bestimmte Frequenz zurück
% Die Bauteilparameter sind in dieser Funktion bestimmt
%**************************************************************************
Rref = 50; %50 Ohm Eingangs und Lastwiderstand (Bezugswiderstand)
Rref1 = Rref/2;
Rref2 = Rref/2;

%für L0
Lr = 9.2e-6;

C1 = 6.46e-12;
C2 = C1;

Rw = 11.3e-3;

Ry1 = 246e-3;
Cy1 = 3.85e-9;
Ly1 = 8e-9;

Rx1 = 47.4e-3;
Lx1 = 15.2e-9;
Cx1 = 146.7e-9;

Rx2 = Rx1;
Lx2 = Lx1;
Cx2 = Cx1;

f = frequency;
w = 2*pi*f;

% ZY2, Widerstand und Spule hablieren, Kondensator verdoppeln
Rx2 = 0.5 * Rx2;
Lx2 = 0.5 * Lx2;
Cx2 = 2.0 * Cx2;

ZY2 = Rx2 + Lx2*j*w + 1/(Cx2*j*w);

Z =  Lr*w*j + Rw;

% ZCX1, Rx1 und Lx1 halbieren, Cx1 verdoppeln
Cx1 = 2.0 * Cx1;
Rx1 = 0.5 * Rx1;
Lx1 = 0.5 * Lx1;
ZCX1 = Rx1 + Lx1*j*w + 1/(Cx1*j*w);

% ZCY2 faellt weg
ZCY1 = Ry1 + Ly1*w*j + (1/(j*w*Cy1));
```

```
ZY1 = rpara(ZCY1,ZCX1); %Parallelschaltung der X und Y Kondensatoren

%A Matrixen aus Impedanzen berechnen:
%Querimpedanz A1:
A1 = [1 0; (1/ZY2) 1];      % Y2 in Schema

%Längsimpedanz A2           % Z in Schema
A2 = [1 Z; 0 1];

%Querimpedanz A3
A3 = [1 0; (1/ZY1) 1];      % Y1 in Schema

%Gesamtmatrix berechnen
A = A1 * A2;
A = A * A3;

%s21 berechnen
result = 2 / ( (A(1,1)*sqrt(Rref2/Rref1)) + (A(1,2)/sqrt(Rref1*Rref2)) +
(A(2,1)*sqrt(Rref2*Rref1)) + (A(2,2)*sqrt(Rref1/Rref2)));

end
```

Funktion get_s21_CM.m

```
function [result] = get_s21_CM(frequency)
%************************************************************************
% Projekt: s21 Parameter berechnen
%          fuer die CM-Schaltung eines EMI-Filters
% Autor:   Luca Krummenacher
% Datum:   30. April 2019
% Ort :    FHNW Brugg-Windisch
% Version: 3.0
% Erneuerungen: Matlabcode stimmt nun mit Softwarecode überein.
%************************************************************************
% Gibt den s21 parameter fuer eine bestimmte Frequenz zurück
% Die Bauteilparameter sind in dieser Funktion bestimmt
%************************************************************************
% used functions: par.m
%************************************************************************
Rref1 = 50; % Rref1 = Eingangswiderstand
Rref2 = 50; % Rref2 = Lastwiderstand
Rw = 11.3e-3;
Lr = 9.2e-6;
C1C2 = 12.93e-12;
Rp = 7.92e3;
L0 = 0.5e-3;
f = frequency;
w = 2*pi*f;

ZZtot = 0.5*Rw + 0.5*Lr*w*j + par(L0*w*j,par(0.5*Rp,(1/(j*w*C1C2))));
%ZY aus vereinfachter Schaltung:
Ry1 = 246e-3;
Cy1 = 3.85e-9;
Ly1 = 8e-9;
%beide CY Kondensatoren gleich setzten
Ry2 = Ry1;
```

```
Cy2 = Cy1;
Ly2 = Ly1;

ZY1 = Ry1 + Ly1*w*j + (1/(j*w*Cy1));

ZY2 = Ry2 + Ly2*w*j + (1/(j*w*Cy2));

%A Matrizen aus Impedanzen berechnen:
%Laengsimpedanz A1:
A1 = [1 ZZtot; 0 1];

%Querimpedanz A2:
A2 = [1 0; (1/ZY1) 1];

%Querimpedanz A3:
A3 = [1 0; (1/ZY2) 1];

%Gesamtmatrix berechnen
A = A1 * A2;
A = A * A3;

%s21 berechnen
result = 2 / ( (A(1,1)*sqrt(Rref2/Rref1)) + (A(1,2)/sqrt(Rref1*Rref2)) +
(A(2,1)*sqrt(Rref2*Rref1)) + (A(2,2)*sqrt(Rref1/Rref2)));

end
```

C. Funktionsspuren (Traces)

Spur Nr. 1: Aufstarten

Attribute von **Main**@1939887983 werden initialisiert ...
Start via Main.main(String args[])
Attribute von **Model**@1359557015 werden initialisiert ...
Konstruktor Model():**Model**@1359557015 wird ausgeführt ...
Attribute von **Controller**@2002526006 werden initialisiert ...
Konstruktor Controller():**Controller**@2002526006 wird ausgeführt ...
Methode **Controller**@2002526006.**createCircuit()** wird ausgeführt ...
Attribute von **Circuit**@1743494606 werden initialisiert ...
Konstruktor Circuit():**Circuit**@1743494606 wird ausgeführt ...
Attribute von **View**@199833255 werden initialisiert ...
Konstruktor View():**View**@199833255 wird ausgeführt ...
Attribute von **CustomMenuBar**@817456115 werden initialisiert ...
Konstruktor CustomMenuBar():**CustomMenuBar**@817456115 wird ausgeführt ...
Attribute von **GraphPane**@1178765742 werden initialisiert ...
Konstruktor GraphPane():**GraphPane**@1178765742 wird ausgeführt ...
Attribute von **GraphBox**@539817776 werden initialisiert ...
Attribute von **GraphBox**@539817776 werden initialisiert ...
Konstruktor GraphBox():**GraphBox**@539817776 wird ausgeführt ...
Attribute von **GraphBox**@1366584642 werden initialisiert ...
Attribute von **GraphBox**@1366584642 werden initialisiert ...
Konstruktor GraphBox():**GraphBox**@1366584642 wird ausgeführt ...
Attribute von **SchemaPane**@1253084837 werden initialisiert ...
Konstruktor SchemaPane():**SchemaPane**@1253084837 wird ausgeführt ...
Attribute von **SchemaBox**@2046380526 werden initialisiert ...
Konstruktor SchemaBox():**SchemaBox**@2046380526 wird ausgeführt ...
Attribute von **SchemaBox**@2040218795 werden initialisiert ...
Konstruktor SchemaBox():**SchemaBox**@2040218795 wird ausgeführt ...
Attribute von **CircuitPane**@79762501 werden initialisiert ...
Konstruktor CircuitPane():**CircuitPane**@79762501 wird ausgeführt ...
Methode **Controller**@2002526006.**createCompBoxesFromCircuit()** wird ausgeführt ...
Methode **Controller**@2002526006.**setView()** wird ausgeführt ...
Objekt **View**@199833255 wird als Observer von Objekt **Model**@1359557015 registriert!
Methode **Controller**@2002526006.**setMainLine()** wird ausgeführt ...
Methode **Model**@1359557015.**calculateCM()** wird ausgeführt ...
Methode **Model**@1359557015.**calculateDM()** wird ausgeführt ...
Methode **Model**@1359557015.**notifyObservers()** wird ausgeführt ...
Methode **View**@199833255.**update()** wird ausgeführt ...
Methode **Model**@1359557015.**getFrequency()** wird ausgeführt ...
Methode **Model**@1359557015.**getListCM()** wird ausgeführt ...
Methode **GraphBox**@539817776.**drawMainLine()** wird ausgeführt ...
Methode **Model**@1359557015.**getFrequency()** wird ausgeführt ...
Methode **Model**@1359557015.**getListDM()** wird ausgeführt ...
Methode **GraphBox**@1366584642.**drawMainLine()** wird ausgeführt ...

Spur Nr. 2: Schieberegler verschieben

Event:
Methode **SubCompSliderBox**@1423836390.**slider.valueProperty.anonymousListener()** wird durch Ereignis ausgelöst ...
Methode **Controller**@2002526006.**setMainLine()** wird ausgeführt ...
Methode **Model**@1359557015.**calculateCM()** wird ausgeführt ...
Methode **Model**@1359557015.**calculateDM()** wird ausgeführt ...
Methode **Model**@1359557015.**notifyObservers()** wird ausgeführt ...
Methode **View**@199833255.**update()** wird ausgeführt ...
Methode **Model**@1359557015.**getFrequency()** wird ausgeführt ...
Methode **Model**@1359557015.**getListCM()** wird ausgeführt ...
Methode **GraphBox**@539817776.**drawMainLine()** wird ausgeführt ...
Methode **Model**@1359557015.**getFrequency()** wird ausgeführt ...
Methode **Model**@1359557015.**getListDM()** wird ausgeführt ...
Methode **GraphBox**@1366584642.**drawMainLine()** wird ausgeführt ...

Spur Nr. 3: Hover über der SubCompSliderBox

Event:
Methode **SubCompSliderBox**@1137935276.**hoverProperty.anonymousListener()** wird durch Ereignis ausgelöst ...
Methode **Controller**@929884794.**showSweepLines()** wird ausgeführt ...
Methode **Model**@1359557015.**calculateCM()** wird ausgeführt ...
Methode **Model**@1359557015.**calculateDM()** wird ausgeführt ...
Methode **Model**@1359557015.**notifyObservers()** wird ausgeführt ...
Methode **View**@199833255.**update()** wird ausgeführt ...
Methode **Model**@1359557015.**getFrequency()** wird ausgeführt ...
Methode **Model**@1359557015.**getListCM()** wird ausgeführt ...
Methode **GraphBox**@539817776.**removeSweepLines()** wird ausgeführt ...
Methode **GraphBox**@539817776.**drawSweepLines()** wird ausgeführt ...
Methode **Model**@1359557015.**getFrequency()** wird ausgeführt ...
Methode **Model**@1359557015.**getListDM()** wird ausgeführt ...
Methode **GraphBox**@1366584642.**removeSweepLines()** wird ausgeführt ...
Methode **GraphBox**@1366584642.**drawSweepLines()** wird ausgeführt ...

Spur Nr. 4: Textfeld wenn Enter gedrückt

Event:
Methode **SubCompSettingBox**@1186140299.**slider.valueProperty.anonymousListener()** wird durch Ereignis ausgelöst ...
Methode **Controller**@2002526006.**setMainLine()** wird ausgeführt ...
Methode **Model**@1359557015.**calculateCM()** wird ausgeführt ...
Methode **Model**@1359557015.**calculateDM()** wird ausgeführt ...
Methode **Model**@1359557015.**notifyObservers()** wird ausgeführt ...
Methode **View**@199833255.**update()** wird ausgeführt ...
Methode **Model**@1359557015.**getFrequency()** wird ausgeführt ...
Methode **Model**@1359557015.**getListCM()** wird ausgeführt ...
Methode **GraphBox**@539817776.**drawMainLine()** wird ausgeführt ...
Methode **Model**@1359557015.**getFrequency()** wird ausgeführt ...
Methode **Model**@1359557015.**getListDM()** wird ausgeführt ...
Methode **GraphBox**@1366584642.**drawMainLine()** wird ausgeführt ...

Spur Nr. 5: Click auf Hamburger in CompBox

Event:
Methode **CompBox**@452408347.**hamburger.setOnMouseClicked.anonymousEventHandler()** wird durch Ereignis ausgelöst ...
Methode **Controller**@2002526006.**changeAppearanceOf()** wird ausgeführt ...

Spur Nr. 6: Doppelclick auf GraphBox

Event:
Methode **GraphBox**@539817776.**setOnMouseClicked.anonymousEventHandler()** wird durch Ereignis ausgelöst ...
Methode **Controller**@2002526006.**toggleSizeOfGraph()** wird ausgeführt ...

Spur Nr. 7: File->Save As

Event:
Methode **CustomMenuBar**@667630039.**menuItemSaveAs.setOnAction.anonymousEventHandler()** wird durch Ereignis ausgelöst ...
Methode **Controller**@263196231.**writeCircuitToFile()** wird ausgeführt ...
Methode **Controller**@263196231.**circuitToJson()** wird ausgeführt ...

Spur Nr. 8: File->Open

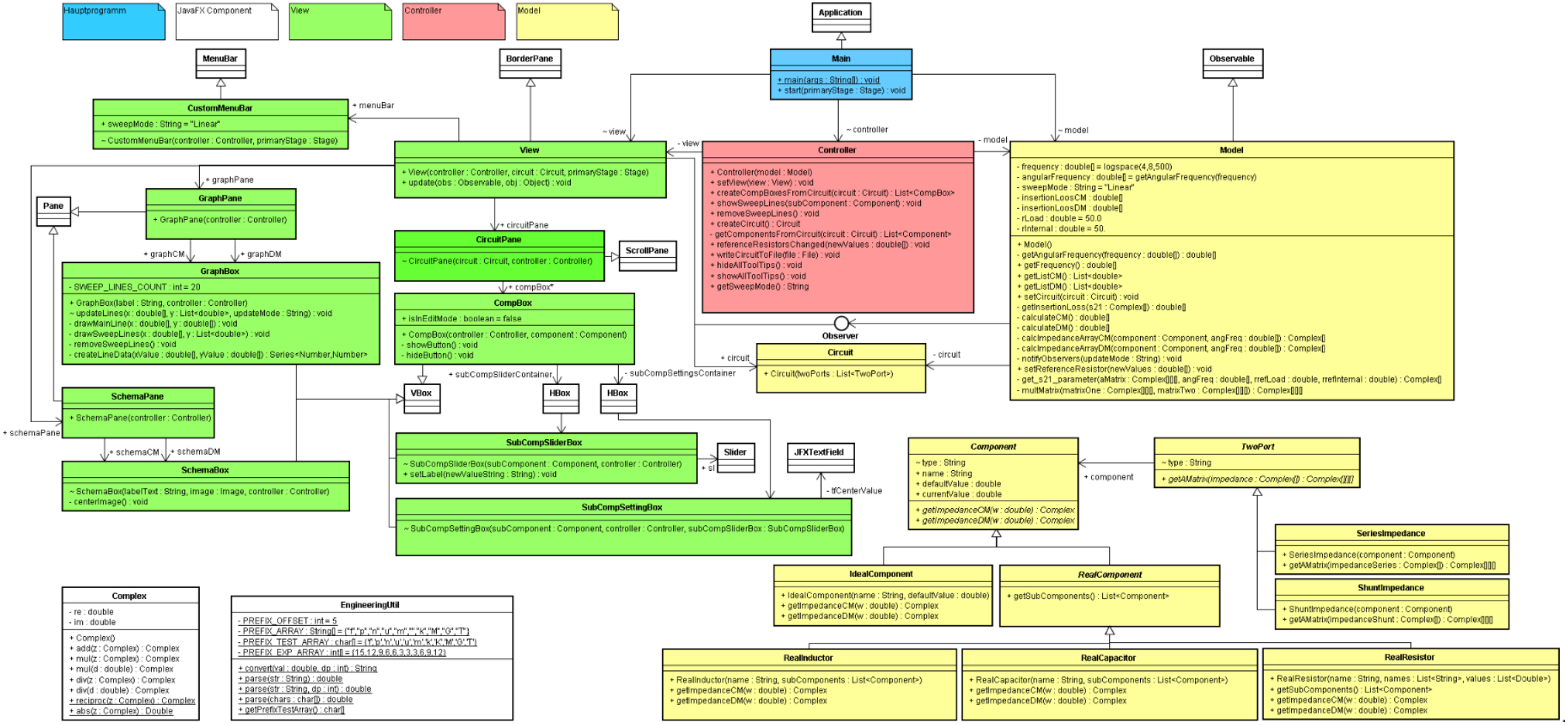
Methode **CustomMenuBar**@667630039.**meuItemOpen.setOnAction.anonymousEventHandler()** wird durch Ereignis ausgelöst ...
Methode **Controller**@263196231.**readFile()** wird ausgeführt ...
Methode **Controller**@263196231.**createCompBoxesFromCircuit()** wird ausgeführt ...

Spur Nr. 9: Berechnungen von CM und DM

Methode **Model**@1359557015.**calculateCM()** wird ausgeführt ...
Methode **Model**@1359557015.**getImpedanceArrayCM()** wird ausgeführt ...
Methode **Model**@1359557015.**getImpedanceArrayCM()** wird ausgeführt ...
Methode **Model**@1359557015.**getImpedanceArrayCM()** wird ausgeführt ...
Methode **SeriesImpedance**@109048629.**getAMatrix()** wird ausgeführt ...
Methode **ShuntImpedance**@162020321.**getAMatrix()** wird ausgeführt ...
Methode **ShuntImpedance**@162020321.**getAMatrix()** wird ausgeführt ...
Methode **Circuit**@1334002592.**get_s21_parameter()** wird ausgeführt ...
Methode **Model**@1359557015.**getInsertionLoss()** wird ausgeführt ...

Methode **Model**@1359557015.**calculateDM()** wird ausgeführt ...
Methode **Model**@1359557015.**getImpedanceArrayDM()** wird ausgeführt ...
Methode **Model**@1359557015.**getImpedanceArrayDM()** wird ausgeführt ...
Methode **Model**@1359557015.**getImpedanceArrayDM()** wird ausgeführt ...
Methode **Model**@1359557015.**getImpedanceArrayCM()** wird ausgeführt ...
Methode **ShuntImpedance**@2092478316.**getAMatrix()** wird ausgeführt ...
Methode **SeriesImpedance**@109048629.**getAMatrix()** wird ausgeführt ...
Methode **ShuntImpedance**@2063007665.**getAMatrix()** wird ausgeführt ...
Methode **ShuntImpedance**@162020321.**getAMatrix()** wird ausgeführt ...
Methode **Circuit**@1334002592.**get_s21_parameter()** wird ausgeführt ...
Methode **Model**@1359557015.**getInsertionLoss()** wird ausgeführt ...
Methode **Model**@1359557015.**notifyObservers()** wird ausgeführt ...

D. Klassendiagramm



E. Aufgabenstellung

Die Aufgabestellung ist als beiliegendes Dokument ab der nächsten Seite zu finden.