Hochschule Magdeburg-Stendal
Fachbereich Ingenieurwissenschaften und Industriedesign (IWID)
Institut für Elektrotechnik

Bachelorarbeit

zur Erlangung des Grades eines „Bachelor of Engineering“ im Studiengang Elektrotechnik

Thema: **„Entwicklung einer benutzerfreundlichen Software zur Berechnung von Rundfunk-Sendeantennen“**

Eingereicht von: **Christine Fuhrmann**

Angefertigt für: NDR Norddeutscher Rundfunk, Hamburg

Matrikel: E 2010

Ausgabetermin: 10.06.2015

Abgabetermin: 31.07.2015

Schulischer Betreuer: Herr Prof. Dr-Ing. Dieter Schwarzenau

Betrieblicher Betreuer: Herr Dr. Nils Eulig

..... 1. Prüfer

..... 2. Prüfer

Inhalt

Abbildungsverzeichnis

Tabellenverzeichnis

Listingverzeichnis

Abkürzungsverzeichnis

Abb.	Abbildung
bspw.	beispielsweise
bzw.	beziehungsweise
d. h.	das heißt
dt.	deutsch
Gl.	Gleichung
GUI	Graphical User Interface, dt.: Grafische Benutzeroberfläche
i. d. R.	in der Regel
s.	siehe
S.	Seite
s. u.	siehe unten
usw.	und so weiter
z. B.	zum Beispiel

1 Einführung

1.1 Stand der Technik

Die Aufgaben der Abteilung Sendertechnik, in der die Arbeit entstanden ist, bestehen in der Planung und im Betreiben von Sendern, der Versorgung der Rundfunknutzer mit Hörfunk und Fernsehen über die Verbreitungswege UKW, DAB+ und DVB-T sowie in der Planung neuer Sendeantennen und der Optimierung bestehender Antennen mit Rücksicht auf Versorgungsanforderungen und Einschränkungen durch geometrische und koordinatorische Auflagen. Diese Bachelorarbeit setzt an den beiden letzteren Punkten an.

Grundlage für die bestehende Antennensoftware ist ein Programmcode, der vor einiger Zeit in der Programmiersprache Quick Basic geschrieben wurde. Zu dieser Software existiert eine ausführliche Analyse in Form einer Bachelorarbeit (s. (Sengelmann 2010)). Basierend auf dieser Analyse und den daraus resultierenden Optimierungsvorschlägen wurde eine neuere Software mit dem grafischen Programmiersystem LabVIEW erstellt, die aktuell genutzt wird. Die Benutzeroberfläche dieses Programms enthält veraltete Funktionen, welche zur Nachbildung der Elementdaten anhand mathematischer Funktionen erforderlich waren, als die Hersteller noch keine Angaben über die Abstrahlungseigenschaften der jeweiligen Elemente lieferten. Da die Antennenhersteller heute meistens die zur Berechnung von Gruppendiagrammen erforderlichen Primärdaten der Antennenelemente angeben, sind die veralteten Funktionen nicht mehr erforderlich. Sie können daher wegfallen, was die Programmoberfläche übersichtlicher gestaltet.

1.2 Aufgabenstellung

Das Ziel der vorliegenden Arbeit war die Entwicklung einer benutzerfreundlichen Software zur Berechnung von Rundfunksendeantennen, welche die bestehende, vom NDR selbst entwickelte LabVIEW-Software ablösen und optimieren sollte.

Die Aufgabe bestand darin, die Rechenalgorithmen der alten Software in eine neue und aktuelle Programmumgebung einzubinden und auf dieser Grundlage eine verbesserte und benutzerfreundlichere Oberfläche zu entwickeln. Die Antennenparameter sollten einfach vom Benutzer eingegeben und geändert werden können und es sollte möglich sein, diese

Änderungen in einem berechneten Horizontaldiagramm zu beobachten sowie die Ergebnisse zu dokumentieren.

Im Folgenden sollen die Anforderungen an die Software im Detail erläutert werden. Das Endergebnis ist ein Horizontaldiagramm einer Rundfunkantenne, das einem bestimmten Vorgabediagramm mit den Soll-Werten des Horizontaldiagramms so weit wie möglich angenähert ist. Die Daten für das Horizontaldiagramm können als Textdatei abgespeichert und somit in eine andere Software zum Weiterverarbeiten eingelesen werden. Zusätzlich lässt sich eine PDF-Datei erzeugen, die das Horizontaldiagramm und die wichtigsten Antenneneigenschaften enthält.

Um das Horizontaldiagramm zu berechnen, sollten mehrere Daten abgefragt werden, die über die grafische Benutzeroberfläche (GUI) in das Programm eingelesen werden. Dazu gehören die in Tabelle 1 aufgeführten Daten.

Tabelle 1: Daten zur Berechnung des Horizontaldiagramms

Grundlagendaten:	• Vorgabedaten • Primärdaten
Antennenparameter:	• Nennfrequenz • Rechenfrequenz • Anzahl der Spalten
Elementspezifische Parameter:	• Leistung • Speisephase • Strahlrichtung • Neigungswinkel • Elemententfernung • Elementrichtung • Vertikalabstand • konstante Phase

Die Definition der Begriffe ist in Tabelle 2 erläutert. Nachdem alle Daten eingegeben wurden, soll die Software das entsprechende Horizontaldiagramm berechnen, darstellen und bei jeder Parameteränderung aktualisieren. So kann mit der schrittweisen Änderung der Parameter ein Diagramm erzeugt werden, das dem Vorgabediagramm nahe kommt.

Zu diesen Anforderungen an das Endergebnis entstanden in den Vorbesprechungen zu den Funktionen der Software viele Ideen und weitere Zielsetzungen. Nicht alle konnten in die Software integriert werden, der Programmcode ist jedoch darauf ausgelegt, dass er beliebig erweitert werden kann. Die Liste dazu befindet sich im Anhang B.

2 Grundlagen

2.1 Begriffsdefinitionen

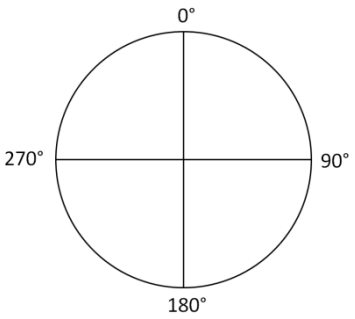
2.1.1 Antennenbegriffe

Für das Verständnis sowohl der theoretischen Grundlagen als auch des Programmaufbaus sind einige Begriffe wichtig, die mit Antennen und deren Darstellung des Strahlungsdiagramms zusammenhängen. In Tabelle 2 sind diese Begriffe in alphabetischer Ordnung aufgelistet und kurz definiert.

Tabelle 2: Begriffsdefinitionen Grundlagen

Begriff	**Definition**
Antennendiagramm	Grafische Darstellung der Richtcharakteristik einer Antenne; in dieser Arbeit als Ortskurve in einem linearen Polardiagramm, auf 1 normiert (Gotthard 1989, S. 25; Sengelmann 2010, S. VI); Horizontaldiagramm: siehe Kapitel 2.2.2.
Antennenelement	Sendeantenne; hier als Antennenelement bezeichnet, da sich das Antennensystem aus mehreren Antennen(-elementen) zusammensetzt (s. u.)
Antennensystem	Anordnung mehrerer Antennenelemente
Azimutwinkel φ	Horizontalwinkel $0 \dots 360\ °$ (s. Abbildung 3 auf S. 14); $0\ ° =$ Norden im Horizontaldiagramm
c	Lichtgeschwindigkeit im Vakuum, bzw. Ausbreitungsgeschwindigkeit elektromagnetischer Wellen im Vakuum
Differenzwinkel δ_h	Differenzwinkel zwischen Azimutwinkel und Strahlrichtung, $\delta_h = \varphi - \beta$
Elemententfernung	Horizontale Entfernung des Antennenelements vom Bezugspunkt bzw. Antennenmastmittelpunkt
Elementrichtung	Horizontale Ausrichtung des Antennenelements im Azimutwinkel

Fortsetzung Tabelle 2: Begriffsdefinitionen Grundlagen

Begriff	**Definition**
Erhebungswinkel ϑ	Vertikalwinkel $0 \ldots 180\ °$ (s. Abbildung 3 auf S. 14); in der Darstellung im Vertkaldiagramm um $90°$ gedreht: $-90 \ldots +90\ °$, so dass $-90\ °$ = Norden, $0\ °$ = Osten; diese Definition ist in Anlehnung an die Herstellerangaben festgelegt
Ebenen	eine Ebene besteht aus einer Reihe horizontal nebeneinander angeordneter Elemente in gleicher Höhe; eine Spalte mit drei Ebenen entspricht drei senkrecht übereinander angeordneten Elementen
Höhenphase	Teil der Phase eines Antennenelements; beeinflusst vom Vertikalabstand des Elements vom Schwerpunkt der Antenne und vom Erhebungswinkel
Horizontaldiagramm	Draufsicht auf das Antennendiagramm; Amplitude aufgetragen auf den Azimutwinkel in Polarkoordinaten; mehr dazu s. Kapitel 2.2.2.  *Abbildung 1: Ausrichtung Horizontaldiagramm*
konstante Phase	Teil der Phase eines Antennenelements; die Phase, die einen frequenzunabhängigen Betrag am Speisepunkt des Antennenelements liefert.
Leistung	In den Raum abgestrahlte Leistung eines Antennenelements als auf 1 normierter Wert: $\text{Leistung} = 1$ entspricht einer Leistung von $100\ \%$. In dieser Arbeit wird mit der Amplitude A gearbeitet, sie entspricht der Quadratwurzel der Leistung: $$A = \sqrt{Leistung} \quad (1)$$
Neigungswinkel	Mechanische Neigung eines Antennenelements bezogen auf $0\ °$ bzw. auf den Horizont

Fortsetzung Tabelle 2: Begriffsdefinitionen Grundlagen

Begriff	**Definition**
Nennfrequenz	die Mittenfrequenz der geplanten Betriebsfrequenzen; Begründung: Die Längen aller Kabel können nur für eine Frequenz ausgelegt werden, bei anderen Frequenzen treten Phasenabweichungen auf. Um diese Abweichungen zu minimieren; wird die Mittenfrequenz gebildet.
Primärdaten	eine Textdatei, die Amplitude und Phase pro Winkel $0 \dots 360\ °$ des eingesetzten Antennenelements enthält, also die Richtcharakteristik des Elements; für jede Antenne werden vom Hersteller die Daten für das Horizontaldiagramm sowie für das Vertikaldiagramm geliefert.
Rechenfrequenz	die Frequenz, für die das Horizontaldiagramm exakt berechnet wird; durch die Rechenfrequenz werden bei konstanten Kabellängen in Bezug auf die Nennfrequenz Abweichungen der Kabelphasen hervorgerufen, diese werden in der Berechnung berücksichtigt
Speisekabelphase	Teil der Phase eines Antennenelements; abhängig von der Rechenfrequenz und der Speisephase des Kabels bei der Nennfrequenz
Speisephase	die Phase, mit der das Antennenelement bei der Rechenfrequenz gespeist wird; sie ist abhängig von der Frequenz und dem Speisenetzwerk
Strahlrichtung β	Richtung in der Horizontalen, in die das Antennenelement am Befestigungspunkt ausgerichtet ist
Spalten	eine Spalte besteht aus vertikal übereinander angeordneten Elementen; vier Spalten in zwei Ebenen entsprechen vier mal nebeneinander jeweils zwei Elementen übereinander
Vertikalabstand	Vertikaler Abstand des Antennenelements von der Schwerpunkthöhe der gesamten Antenne

Fortsetzung Tabelle 2: Begriffsdefinitionen Grundlagen

Vertikaldiagramm	Seitenansicht des Antennendiagramms; Amplitude aufgetragen auf den Erhebungswinkel in Polarkoordinaten
Vorgabedaten	eine Textdatei, die Amplitudenwerte pro Winkel $0 \ldots 360$ ° des Wunschdiagramms enthält, also die Einzüge, die durch umliegende Sender eingehalten werden müssen, um gegenseitige Störungen zu vermeiden

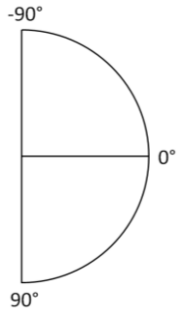


Abbildung 2: Ausrichtung Vertikaldiagramm

Abbildung 3 zeigt das dreidimensionale Kugelkoordinatensystem mit dem Erhebungswinkel ϑ und dem Azimutwinkel φ des Vektors r, der zum Aufpunkt P zeigt.

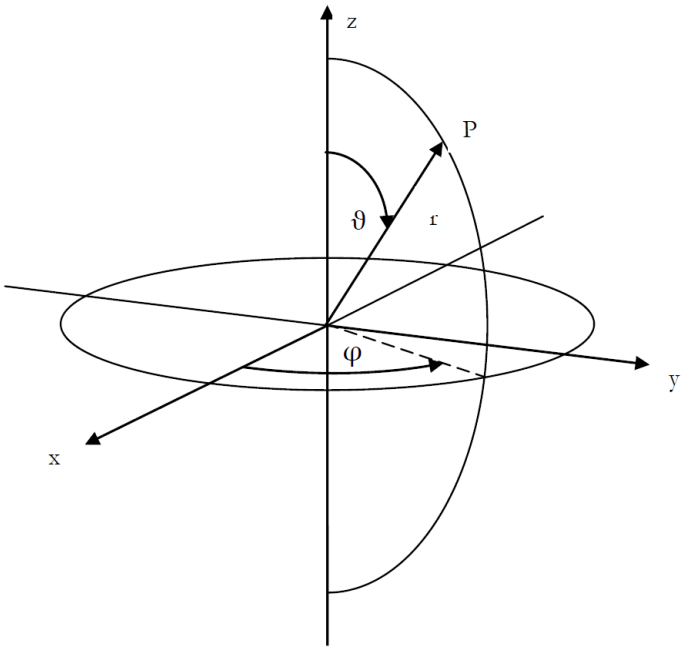


Abbildung 3: Kugelkoordinaten (Sengelmann 2010, S. VII)

In Abbildung 4 wird dargestellt, wie die Winkelbezeichnungen zusammenhängen, die sich auf die geometrische Ausrichtung eines Antennenelements beziehen. Jedes Antennenelement ist vom gemeinsamen Bezugspunkt in einem bestimmten Winkel, der Elementrichtung, am Antennenmast befestigt. Das Antennenelement strahlt seine maximale Leistung in die Strahlrichtung β ab. Der Aufpunkt P befindet sich in Richtung des Azimutwinkels φ. Daraus ergibt sich der Differenzwinkel δ.

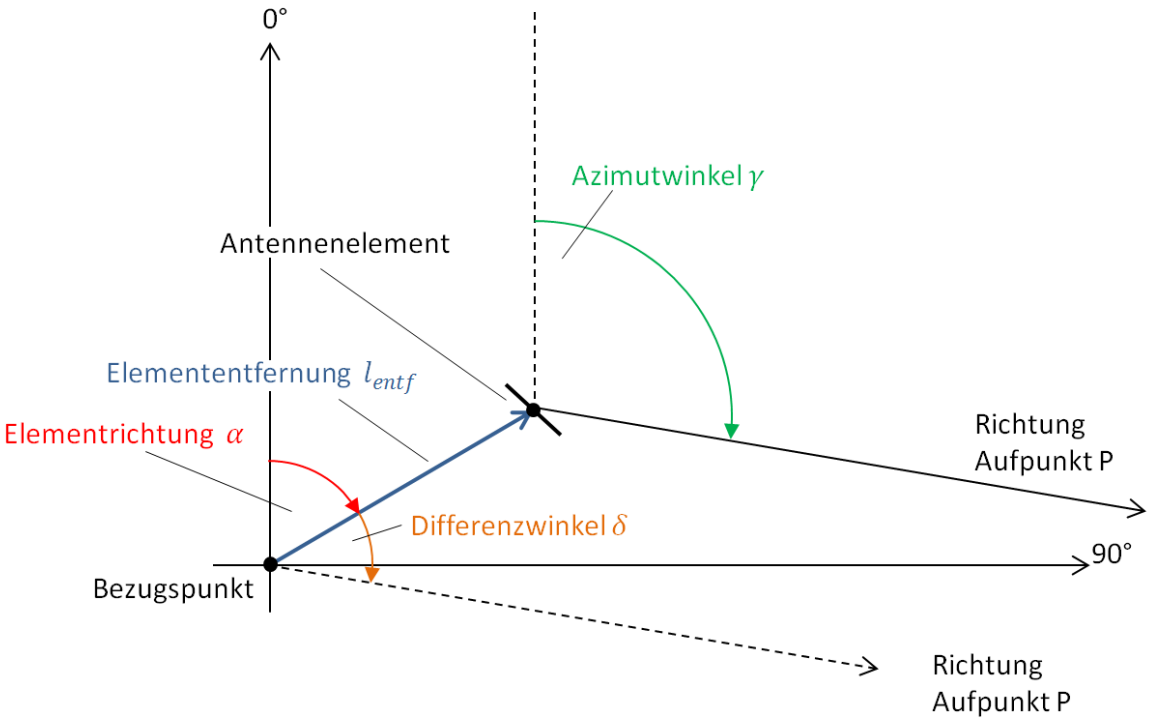


Abbildung 4: Winkelbezeichnungen der geometrischen Ausrichtung eines Antennenelements

Bei der Berechnung des Horizontaldiagramms läuft der Azimutwinkel in $1\,°$-Schritten einmal den gesamten Kreis von $0\,°$ bis $360\,°$ ab. Für jeden Winkel wird der an dieser Stelle ankommende Amplituden- und Phasenwert des Antennenelements berechnet, das mit der Elementrichtung und -entfernung am Bezugspunkt, z. B. dem Mastmittelpunkt, befestigt ist. Der Differenzwinkel zwischen der Elementrichtung und dem Azimutwinkel gibt an, für welchen Winkel die Daten aus den Primärdaten ausgelesen werden müssen. Damit werden für jeden Azimutwinkel und jedes Element die Amplitude und die Phase berechnet, welche u. A. von der Elemententfernung und -richtung abhängig sind.

2.1.2 Programmierbegriffe

Tabelle 3 enthält grundlegende Begriffe zum Verständnis des Entstehungsprozesses und der Struktur der Software in Kapitel 3. Die Begriffe sind alphabetisch geordnet.

Tabelle 3: Begriffsdefinitionen Programmieren

Begriff	**Definition**
Array	„eine Tabelle gleichartiger Elemente wie […] Zahlen, Zeichen oder Zeichenketten“ (Mössenböck 2014, S. 91)
Attribut	Variable einer Klasse (Mössenböck 2014, S. 130)
Double	Datentyp einer Gleitkommazahl
Drag-and-Drop	„Ziehen und Ablegen“ (dt.); Bewegen von grafischen Elementen von einem Ort zum anderen mit dem Mauszeiger
for-Schleife	Ausführung einer Anweisung für (for) eine definierte Anzahl von Malen
GUI	Graphical User Interface (Grafische Benutzeroberfläche)
if-else-Anweisung	Ausführung einer Anweisung, wenn eine Bedingung erfüllt ist (if), sonst Ausführung einer anderen Anweisung (else)
Klasse	„selbst definierte Datentypen, die es erlauben, mehrere Elemente zu einem neuen Objekt zusammenzufassen“ (Mössenböck 2014, S. 129); (Erläuterung des Zusammenhangs s. u.)
Matrix	mehrdimensionales Array
Methode	Anweisungsfolge, die eine bestimmte Teilaufgabe erledigt (Mössenböck 2014, S. 73); (Erläuterung des Zusammenhangs s. u.)
Objekt	wird mit den Attributen einer Klasse erzeugt und kann mit ihren Methoden verändert werden (Erläuterung des Zusammenhangs s. u.)
Package bzw. Paket	„Sammlung[…] zusammengehöriger Klassen“ (Mössenböck 2014, S. 237)
Rückgabewert	der Wert, den eine Methode, je nach Definition, zurückgibt
String	Datentyp einer Zeichenkette
while-Schleife	Ausführung einer Anweisung, solange (while) ein Zustand wahr ist

Der Zusammenhang von Klassen, Attributen, Methoden und Objekten wirkt ohne Vorkenntnisse in objektorientierter Programmierung sehr abstrakt, daher soll er hier mit einigen Sätzen erläutert werden.

In einer Klasse werden Attribute, also Variablen, und Methoden definiert. Wird ein neues Objekt erstellt, erbt es die in der Klasse festgelegten Attribute. Mit den Methoden aus der Klasse können diese Attribute verändert werden. Die Klasse ist also der Datentyp der Objekte, die aus ihr erstellt werden.

2.2 Rundfunk-Sendeantennen

2.2.1 Aufbau einer Rundfunk-Sendeantenne

Eine Rundfunk-Sendeantenne strahlt mit Information modulierte elektromagnetische Wellen ab, die vom Nutzer über geeignete Empfangsgeräte empfangen werden können. Das Antennensystem, das diese Wellen ausstrahlt, besteht i. d. R. aus mehreren Antennenelementen, die in bestimmten geometrischen Anordnungen auf dem Antennenträger montiert sind (Gotthard S. 13). Abbildung 5a zeigt die vertikale, Abbildung 5b die horizontale Anordnung von jeweils vier Antennenelementen.

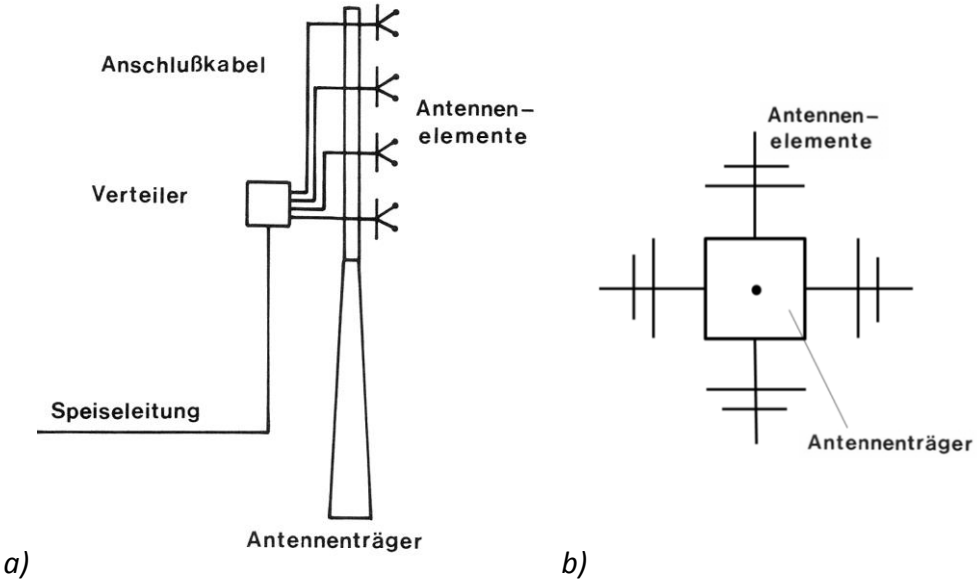
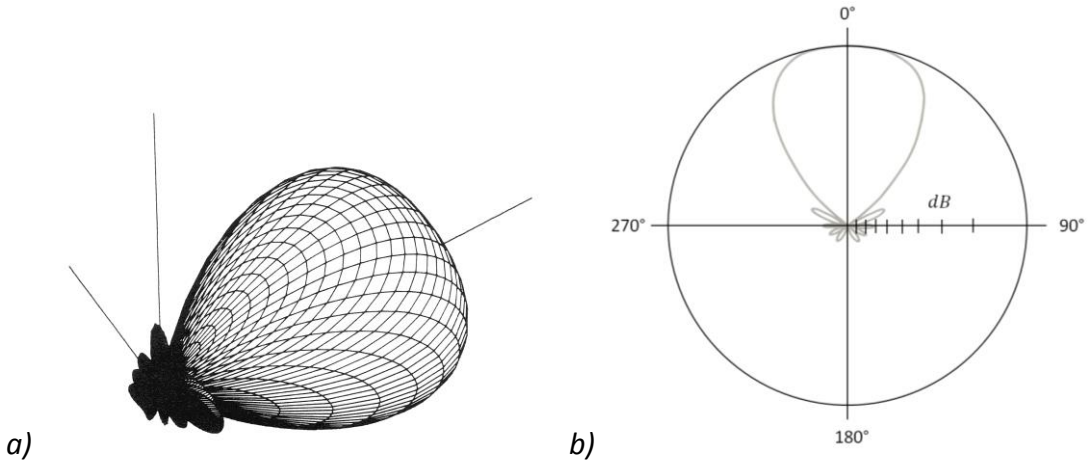


Abbildung 5: Aufbau eines Rundfunksendeantennensystems a) vertikale Anordnung (Gotthard S. 13) b) horizontale Anordnung

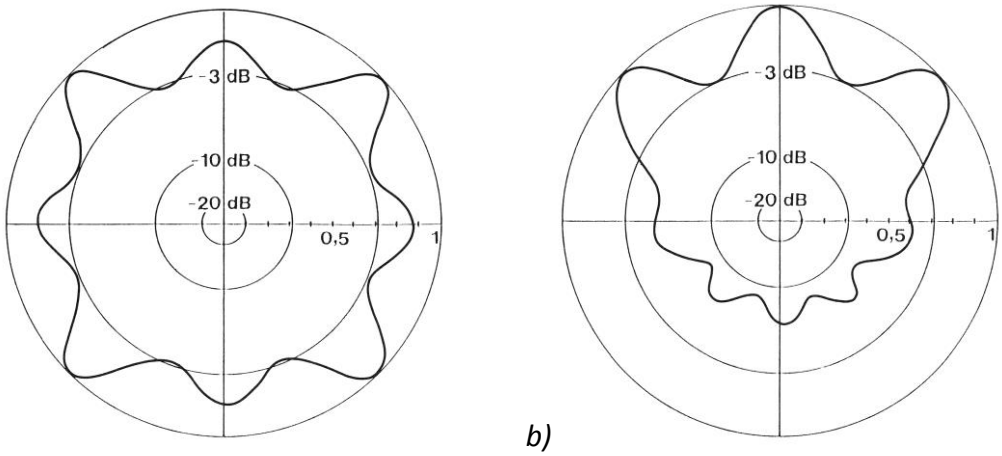
Um die Richtcharakteristik eines einzelnen Antennenelements zu beschreiben, betrachtet man die Amplitude in Abhängigkeit vom Abstrahlungswinkel. Sie beschreibt die abgestrahlte Leistung in den Raum, in Abbildung 6a dreidimensional dargestellt. Am Koordinatenursprung befindet sich das Antennenelement. Diese Darstellung ist jedoch aufwändig herzustellen und erschwert das Ablesen einzelner Werte, daher wird nur das Horizontaldiagramm, d. h. die Horizontalebene für einen konstanten Erhebungswinkel betrachtet (Abbildung 6b). Siehe dazu Kapitel 2.2.2.



a) *b)*

Abbildung 6: Richtcharakteristik eines Antennenelements a) in dreidimensionaler Darstellung (Gotthard S. 25) b) als Horizontaldiagramm

Werden mehrere Antennenelemente in vertikaler und horizontaler Anordnung am Antennenträger angebracht, summieren sich ihre Amplitudenwerte. Durch Anpassung der Elementausrichtung oder der anliegenden Leistung kann die Form des Horizontaldiagramms gezielt verändert werden. (Gotthard S. 25) Siehe dazu Abbildung 7. Das Runddiagramm (a) ist aus vier Antennenelementen mit identischen Eigenschaften zusammengesetzt, wobei die Strahlrichtung der einzelnen Elemente jeweils um $90\,°$ gedreht ist. Die Elemente sind z. B. jeweils an einer Seite eines viereckigen Antennenmasts befestigt. Dagegen wurden für das Richtdiagramm (b) drei Antennenelemente so angeordnet, dass die Hauptstrahlrichtung bei $0\,°$, $120\,°$ und $240\,°$ liegt. Durch Leistungsüberlagerung strahlt die Antenne gerichtet nach Norden.



a) *b)*

Abbildung 7: Kombination mehrerer Antennenelemente als a) Runddiagramm b) Richtdiagramm (Gotthard S. 69)

2.2.2 Horizontaldiagramm

Das Horizontaldiagramm ist in dieser Arbeit definiert als „Richtdiagramm in polarer Darstellung mit linearem Maßstab“ (Gotthard 1989, S. 25). Dabei wird die Amplitude A abhängig vom Azimutwinkel φ in linearem Maßstab in der Einheit dB in einem Polardiagramm aufgetragen. Norden entspricht dabei dem Azimutwinkel $\varphi = 0°$, Osten entspricht dem Azimutwinkel $\varphi = +90°$. Der Wertebereich der Achsen liegt bei $\{0, 1\}$ bzw. $\{-\infty, 0\}\, dB$. Das bedeutet, dass die Amplitudenwerte, aus denen sich das Diagramm zusammensetzt, zueinander ins Verhältnis gestellt werden. Eine Amplitude von $0{,}7$ entspricht dann $70\ \%$ der Maximalamplitude. Die Werte werden linear, die Skala jedoch in dB-Werten aufgetragen. Dadurch entsteht eine logarithmische Skaleneinteilung, wie in Abbildung 7a zu sehen ist. In dieser Arbeit werden die Horizontaldiagramme stets für den Erhebungswinkel $\vartheta = 0°$ berechnet.

2.2.3 Planung einer Antenne

Bei der Planung einer Rundfunksendeantenne wird für einen bestimmten Standort im Versorgungsgebiet ein Antennensystem gesucht, das je nach geforderter Reichweite und Strahlungsleistung die Empfänger in der Umgebung versorgen soll. Standortabhängig gibt es Abstrahlungsbeschränkungen zur Vermeidung von Gleich- oder Nachbarkanalstörungen, z. B. durch benachbarte Sendegebiete anderer Rundfunkanstalten. Es gilt, „das für die Versorgung optimale Strahlungsdiagramm“ zu ermitteln (Gotthard S. 46).

Dafür wird das Antennensystem als Mittelpunkt eines Horizontaldiagramms betrachtet. Die Beschränkungen aus den Himmelsrichtungen sind, wie in Abbildung 8 zu sehen, durch Einzüge dargestellt. Durch das Gruppieren von mehreren Einzelantennen und deren Anbringung am Antennenträger kann die Strahlungscharakteristik des Antennensystems dem Wunschdiagramm angepasst werden.

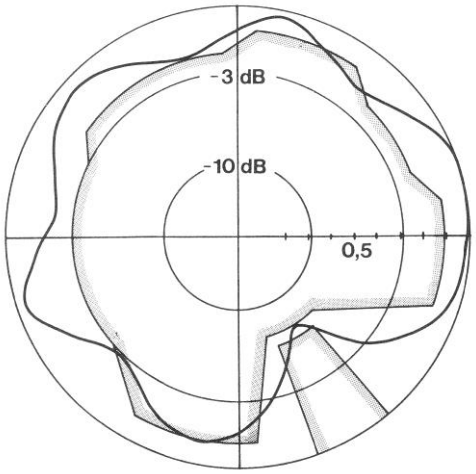


Abbildung 8: Beispiel für die Annäherung einer Antennensystems (schwarz) an ein Wunschdiagramm (grau) (Gotthard S. 46)

2.3 Antennenberechnung

2.3.1 Berechnung der Amplitude im Horizontaldiagramm

Jedes Antennenelement hat für jeden Abstrahlwinkel eine bestimmte Amplitude, die vom Hersteller angegeben wird. Kombiniert man mehrere Antennenelemente, überlagern sich die Strahlungsfelder und es müssen die Amplituden und Phasen für jeden Azimutwinkel φ addiert werden. Daraus ergibt sich für jeden Winkel $\varphi = 0 \dots 360°$ eine Gesamtamplitude.
Um die Werte in das Horizontaldiagramm aufzutragen, müssen die Ergebnisse auf 1 normiert werden. Dafür wird der höchste Wert $\mathrm{A_{max}}$ bestimmt und alle anderen Werte A durch diesen geteilt. Daraus ergibt sich für jede Amplitude A eine auf 1 normierte Amplitude $A_{normiert}$:

$$A_{normiert} = \frac{A}{A_{max}} \tag{2}$$

In der Regel wird ein Antennendiagramm mit einer logarithmischen Achseneinteilung, also in dB, betrachtet, die Darstellung erfolgt wegen der besseren Anschaulichkeit jedoch linear. Für den Export der Antennendaten erfolgt eine Umrechnung der linearen in dB-Werte, um sie besser mit Werten aus Programmen zur Erstellung von Versorgungsprognosen vergleichen zu können. Mit der Gleichung (3) werden die linearen Werte ($Wert_{lin}$) für den Export als Textdatei in dB-Werte ($Wert_{dB}$) umgerechnet:

$$Wert_{dB} = 20 \cdot log_{10}(Wert_{lin}) \quad (3)$$

2.3.2 Berechnung der Phase im Horizontaldiagramm

Die horizontale Phase beschreibt die Phasendifferenz am Aufpunkt P, die durch die horizontale und vertikale Ausrichtung der Antennenelemente entsteht, sowie die elementspezifische Phase aus den Primärdaten des Herstellers. Die horizontale Phase ist also eine Addition der geometrisch erzeugten Phasenverschiebung und der Elementphase. Die hier aufgeführten Berechnungen sind nur für das Fernfeld einer Antenne gültig, das Nahfeld wird nicht betrachtet.

Die Gleichung zur Berechnung der horizontalen Phase lautet unter der Berücksichtigung, dass das Horizontaldiagramm in dieser Arbeit immer für einen Erhebungswinkel von $\vartheta = 0\,°$ berechnet wird:

$$P_{hor} = l_{entf} \cdot cos\ (\varphi - \alpha) \cdot \frac{360°}{c} \cdot f_{rechen} + \varphi_{primär} \quad (4)$$

Tabelle 4 enthält die Definitionen der verwendeten Variablen.

Tabelle 4: Horizontale Phase Definitionen

Symbol	**Definition**
φ_{hor}	Horizontale Phase
$\varphi_{primär}$	interpolierte horizontale Phase aus den Primärdaten
l_{entf}	Elemententfernung
$l_{\varphi,diff}$	Phasendifferenz
f_{rechen}	Rechenfrequenz
α	Elementrichtung
φ	Azimutwinkel
δ	Differenzwinkel

Zuerst soll auf den ersten Teil der Gleichung (4) eingegangen werden:

$$l_{entf} \cdot cos\ (\varphi - \alpha) \tag{5}$$

Abbildung 9 veranschaulicht die Teilgleichung (5), welche die Phasendifferenz berechnet, welche durch die Laufzeitunterschiede durch die mechanische Anordnung der Elemente entsteht. Das Antennenelement ist vom Bezugspunkt aus gesehen in der Elemententfernung und der Elementrichtung am Mast angebracht. Die dadurch entstehende Phasendifferenz wird für jedes Antennenelement relativ zum Bezugspunkt in Richtung des Aufpunkts P berechnet. Der Azimutwinkel ist der Winkel, für den die Berechnung stattfindet.

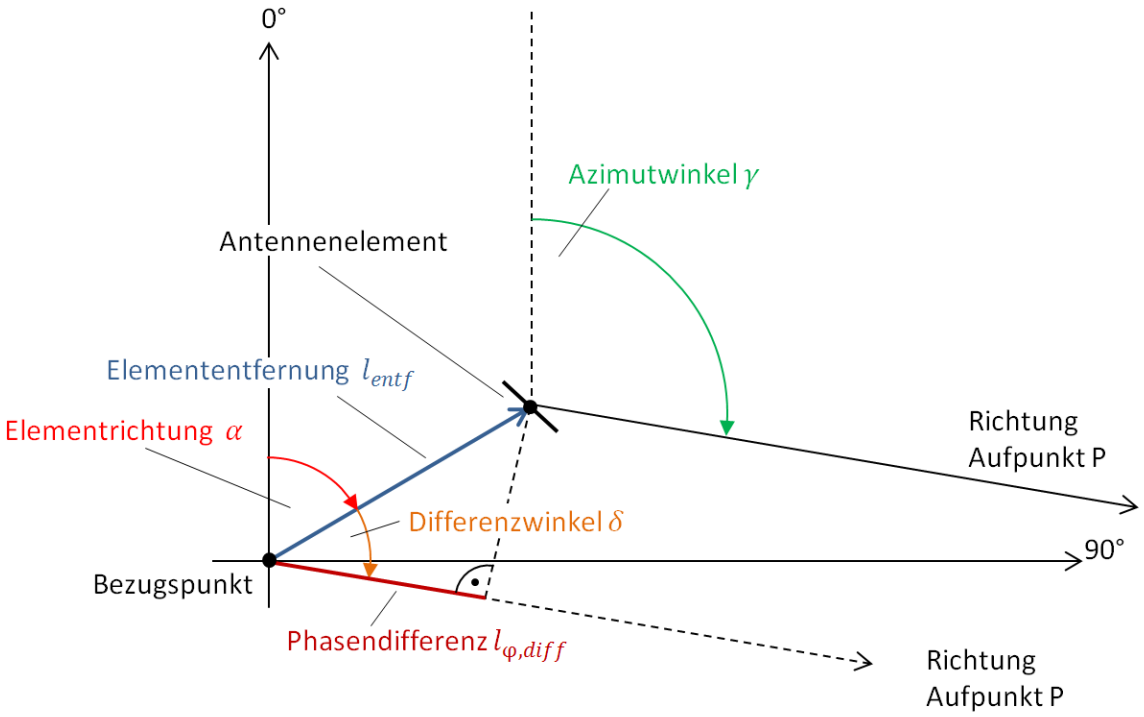


Abbildung 9: Horizontale Phasendifferenz

Die durch den Wegunterschied entstandene Phasendifferenz erhält man, wenn man den Differenzwinkel von Azimutwinkel und Elemententfernung und mit Hilfe der Elemententfernung die Ankathete dieses Winkels berechnet. Siehe dazu Gleichungen (6), (7) und (8).

$$cos\ (\delta) = \frac{l_{\varphi,diff}}{l_{entf}} \tag{6}$$

mit

$$\delta = \varphi - \alpha \tag{7}$$

Umgestellt nach der Phasendifferenz ergibt sich:

$$l_{\varphi,diff} = l_{entf} \cdot cos\ (\delta) \tag{8}$$

Der zweite Teil der Gleichung (4) lautet

$$\cdot \frac{360\ °}{c} \cdot f_{rechen} \tag{9}$$

Das Ergebnis der Teilgleichung 1 ist die Phasendifferenz als Strecke. Da sich elektromagnetische Wellen mit Lichtgeschwindigkeit ausbreiten und die Rechenfrequenz bekannt ist, kann durch Gleichung (9) die Strecke in eine Phase umgerechnet werden.
Der dritte Teil der Gleichung (4) ist die Addition des Phasenanteils, siehe Gleichung (10), der in den Primärdaten für das Antennenelement für den betrachteten Azimutwinkel angegeben ist.

$$+ P_{primär} \tag{10}$$

Im Kapitel 3.7 wird beschrieben, wie diese Berechnungen im Programmcode eingegliedert sind. Auf eine detailliertere Analyse des Rechenalgorithmus soll hier nicht weiter eingegangen werden, er wurde aus dem ursprünglichen Antennenprogramm übernommen und von Sabine Sengelmann in einer Bachelorarbeit eingehend beschrieben, siehe dazu (Sengelmann 2010).

2.3.3 Interpolation

Bei der Berechnung der Gesamtamplitude für einen bestimmten Winkel im Antennendiagramm kann es zu gebrochenen Winkeln (z. B. $2{,}14\ °$) kommen, da die Strahlrichtung des Elements eine Dezimalzahl sein kann. Die Winkel in den Primärdaten sind immer in 1°-Schritten angegeben. Für den Differenzwinkel δ, der die Differenz

zwischen Azimutwinkel und Strahlrichtung beschreibt, kann sich jedoch eine Dezimalzahl ergeben. Um die Amplitude für einen solchen gebrochenen Winkel zu berechnen, müssen der Amplituden- und der Phasenwert linear interpoliert werden.

Bei der Interpolation wird ein unbekannter Wert mit Hilfe bekannter, benachbarter Werte näherungsweise ermittelt. Bei der linearen Interpolation wird angenommen, dass sich die Verteilung der Werte im betrachteten Intervall linear verhält. Für den beschriebenen Anwendungsfall bedeutet dies, dass die gesuchte Amplitude am gebrochenen Winkel im gleichen Verhältnis zu den Amplitudenwerten der benachbarten ganzzahligen Winkel liegt, wie die Winkel zueinander. Abbildung 10 zeigt die beiden ganzzahligen Winkel δ_{ab} und δ_{auf}. Für einen gebrochenen Winkel von bspw. $\delta = 2{,}14\,°$ wäre $\delta_{ab} = 2\,°$ und $\delta_{auf} = 3\,°$.

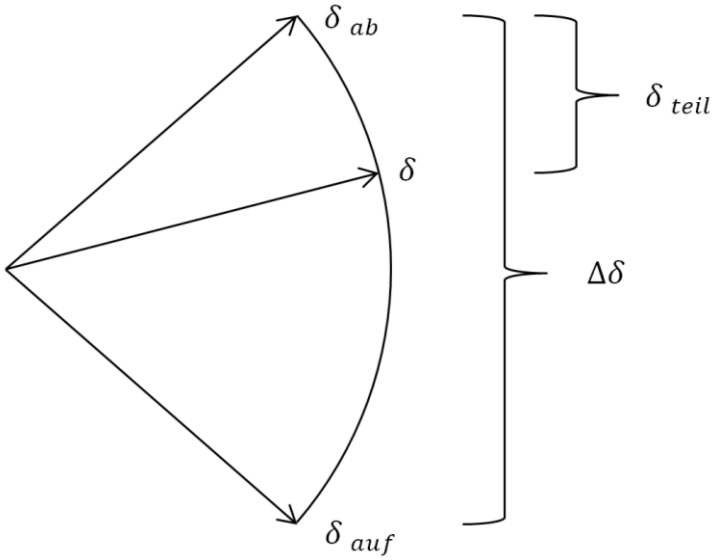


Abbildung 10: Lineare Interpolation (Abb. qualitativ)

enthält die Definitionen für die folgenden Gleichungen.

Tabelle 5: Interpolation Definitionen

Symbol	Definition
δ	gebrochener Winkel
δ_{auf}	aufgerundeter Winkel
δ_{ab}	abgerundeter Winkel
$\Delta\delta$	$\delta_{auf} - \delta_{ab}$
δ_{teil}	Teilwinkel $\delta - \delta_{ab}$
A	gebrochene Amplitude
A_{auf}	Amplitude bei δ_{auf}
A_{ab}	Amplitude bei δ_{ab}
ΔA	$A_{auf} - A_{ab}$
A_{teil}	Teilamplitude $A - A_{ab}$

Das Verhältnis der Teilamplitude A_{teil} zu ΔA ist gleich dem Verhältnis des Teilwinkels δ_{teil} zu $\Delta\delta$ (siehe Gleichung (11)(13)).

$$\frac{A_{teil}}{\Delta A} = \frac{\delta_{teil}}{\Delta\delta} \tag{11}$$

Für die gesamte gebrochene Amplitude gilt

$$A = A_{ab} - A_{teil} \tag{12}$$

Der gebrochene Winkel befindet sich immer zwischen zwei Winkeln, die eine Differenz von $1°$ haben. Daraus folgt, dass $\Delta\delta = 1$ ist.

Nach Umstellen der Gleichung (11) nach $\mathrm{A_{teil}}$ und dem Einsetzen in (12) ergibt sich

$$A = A_{ab} + (\delta - \delta_{ab}) \cdot (A_{auf} - A_{ab}) \tag{13}$$

Die Amplitude am abgerundeten Winkel, also am nächst kleineren ganzzahligen Winkel, wird mit der linear interpolierten Teilamplitude zwischen dem kleineren zum nächst größeren ganzzahligen Winkel addiert. Dieser interpolierte Wert wird im Antennendiagramm für den Horizontalwinkel φ aufgetragen.

2.3.4 Umrechnung von Polar- in Kartesische Koordinaten

Um in Java Werte zu addieren, die in Polarkoordinaten, also in Amplitude (bzw. Betrag) und Phase angegeben sind, müssen sie vorab in Kartesische Koordinaten, also Real- und Imaginärteil, umgerechnet werden. Nach der Addition von jeweils Real- und Imaginärteil kann das Ergebnis wieder zurück in Betrag und Phase transformiert werden.

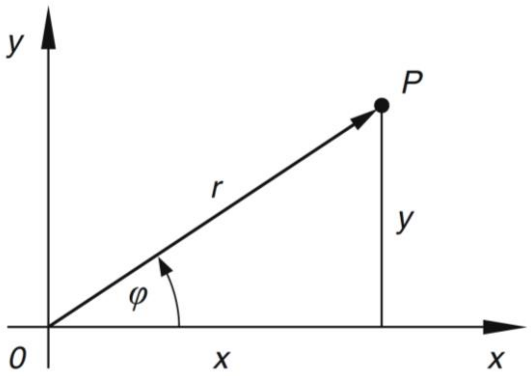


Abbildung 11: Polarkoordinaten $(r;\ \varphi)$ eines Punktes $P = (x, y)$ (Papula 2014a, S. 168)

„Definition: Die Polarkoordinaten $(\mathrm{r};\ \varphi)$ eines Punktes P der Ebene bestehen aus einer Abstandskoordinate r und einer Winkelkoordinate φ“ (Papula 2014a, S. 168) (Abbildung 11). In Tabelle 6 befinden sich die entsprechenden Definitionen.

Tabelle 6: Umrechnung von Polar- in Kartesische Koordinaten: Definitionen

x	Realteil (Kartesische Koordinaten)
y	Imaginärteil (Kartesische Koordinaten)
r	Betrag (Polarkoordinaten): Abstand des Punktes P vom Koordinatenursprung O
φ	Phase (Polarkoordinaten) Winkel zwischen der positiven x-Achse und dem vom Koordinatenursprung O zum Punkt P gerichteten Radiusvektor (Ortsvektor)

Nach (Papula 2014b, S. 169) gelten folgende Gleichungen:

Umrechnung Polarkoordinaten in kartesische Koordinaten

$$x = r \cdot cos(\varphi) \tag{14}$$

$$y = r \cdot sin(\varphi) \tag{15}$$

Umrechnung kartesische Koordinaten in Polarkoordinaten

$$r = \sqrt{x^2 + y^2} \tag{16}$$

$$\varphi = arctan\left(\frac{y}{x}\right) \tag{17}$$

Bei der Berechnung der Phase φ ist zu beachten, dass eine Fallunterscheidung für die einzelnen Quadranten des Koordinatensystems vorgenommen werden muss. Es gilt laut (Papula 2014a, S. 170):

Tabelle 7: Fallunterscheidung bei der Berechnung der Phase

Quadrant	I	II, III	IV
$\varphi =$	$arctan\left(\frac{y}{x}\right)$	$arctan\left(\frac{y}{x}\right) + 180°$	$arctan\left(\frac{y}{x}\right) + 360°$

In der Programmiersprache Java gibt es für diese Fallunterscheidung eine eigene Methode namens „Math.atan2“. Sie gibt ein Ergebnis aus, welches für den Wertebereich $[0, 2\pi]$ gültig ist (Oracle Corporation 2014c).

2.4 Programmieren in Java

2.4.1 Auswahl von Programmiersprache und Entwicklungsumgebung

Die Wahl einer geeigneten Programmiersprache fiel nach einigem Abwägen gegen die Sprache C++ auf Java. Anwendungen, die mit Java programmiert werden, laufen plattformabhängig auf fast jedem Betriebssystem (Abts 2013, S. 1). Java ist einfach und bietet trotzdem ein umfangreiches Angebot von Klassenbibliotheken. (Goll und Heinisch 2014, S. V). Sie unterscheidet sich außerdem von anderen Programmiersprachen durch ihre Objektorientierung und ihre breite Anwendbarkeit (Abts 2013, S. 1). Java ist ein weitverbreitetes Entwicklungswerkzeug für unterschiedlichste Internetangebote und Applikationen und es gibt mehrere professionelle Entwicklungs-umgebungen auf dem

Markt (Niemann 2003, S. 19). Für dieses Projekt wurde die frei verfügbare integrierte Entwicklungsumgebung (IDE) „NetBeans“ verwendet. Siehe dazu Anhang A.

2.4.2 Objektorientierte Programmierung

Im Gegensatz zu Programmiersprachen wie C++ werden bei der objektorientierten Programmierung die Daten und die Funktionen nicht getrennt, sondern zu selbstständigen Einheiten zusammengefasst (Abts 2013, S. 33). Eine Klasse gibt dieser Einheit einen Rahmen, sie enthält Methoden, erzeugt Objekte und legt deren Eigenschaften fest.

Wird ein Objekt aus einer Klasse erzeugt, besitzt es als Datentyp diese Klasse. Es enthält die in der Klasse definierten Eigenschaften in Form von Variablen bzw. Attributen und bestimmte Funktionen in Form von Methoden. Durch die Methoden kann auf dieses Objekt zugegriffen werden, die Objekteigenschaften können so z. B. modifiziert oder ausgegeben werden.

Auf die Objekte kann von außen nur durch Aufrufen der Methoden der dazugehörigen Klasse zugegriffen werden, sie bilden die Schnittstelle zu den Objekten. Während die Methoden einer Klasse meist als öffentlich, also von außerhalb der Klasse aufrufbar, deklariert werden, werden die Variablen einer Klasse als privat, also nach außen hin geschützt, deklariert. Jedes Objekt ist mit den Eigenschaften und Methoden seiner Klasse verknüpft. (Ullenboom 2003, S. 126ff; Abts 2013, S. 33ff; Goll 2014, S. 39ff; Niemann 2003, S. 62ff)

Es wird bspw. die Klasse „Punkt“ definiert, siehe dazu Abbildung 12. Diese Klasse enthält die Eigenschaft „x-Position“ und die Methode „setX“. Es wird ein Objekt der Klasse „Punkt“ mit dem Namen „Punkt1“ erzeugt, wobei ihm die Eigenschaft „x-Position“ zugewiesen wird.

Mithilfe der Methode „setX“ kann der „x-Position“ ein neuer Wert zugewiesen werden, z. B. der Wert „3“. (Goll 2014, S. 91ff)

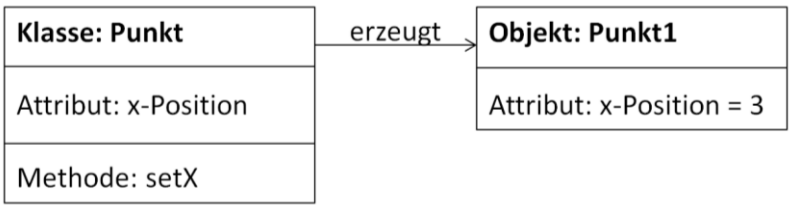


Abbildung 12: Erzeugung eines Objektes

Die Methoden greifen also direkt auf die Eigenschaften der Objekte zu. Werden mehrere Objekte erzeugt, können die Methoden auf jede dieser einzelnen Objekte angewandt werden.

2.4.3 GUI – Graphical User Interface

Die grafische Benutzeroberfläche, auch GUI (Graphical User Interface), ist die Schnittstelle, über die der Benutzer mit der Anwendung kommuniziert. Dies geschieht über Fenster, in denen Elemente zur Ein- und Ausgabe von Informationen positioniert sind. (Balzert und Priemer 2014, S. 8) Java bietet für das Erstellen von GUIs das Paket „Java.awt“ (AWT steht für Abstract Window Toolkit). Es enthält verschiedene „Swing“-Klassen, mit denen grafisch optimierte GUI-Elemente wie Menüs, Textfelder oder Buttons erstellt und angeordnet werden können. (Abts 2013, S. 227)

3 Entwicklung der Software

In diesem Teil der Arbeit wird beschrieben, wie die Software entwickelt wurde, welche Programmteile welche Funktion haben und wie die Aufgabenstellung bearbeitet wurde.

3.1 Java und NetBeans

Um Java als neue Programmiersprache und NetBeans als Entwicklungsumgebung für Java kennenzulernen, gibt es im Internet diverse Tutorials, also Schritt-für-Schritt-Anleitungen als Video oder bebilderten Text. Zusätzlich existieren viele Internet-Foren, in denen sich Java-Nutzer gegenseitig helfen, Tipps geben oder darüber diskutieren, wie die unterschiedlichsten Programmierprobleme gelöst werden könnten. Der Einstieg in Sprache und Entwicklungsumgebung erfolgte über Anfängerbeispiele und kleine Testprogramme. Für das Verständnis der Zusammenhänge und zum Nachschlagen von Definitionen und Begriffsdefinitionen wurde die im Literaturverzeichnis aufgezählte Fachliteratur genutzt. Nützlich zum Erlernen der Sprache waren aber vor allem die Vorkenntnisse in anderen Programmiersprachen, da sich die Syntax und die Struktur der Sprachen in vielen Fällen ähneln.

3.2 Bibliotheken

Einer der vielen Vorteile von Java sind die zahlreichen Bibliotheken, in denen für unterschiedliche Anwendungsbereiche spezielle Klassen und Methoden enthalten sind. Zur Verfügung stehen z. B. die von Oracle entwickelte Klasse „System“ mit der Methode „System.out.print“, die Zeichen und Variablenwerte auf dem Bildschirm ausgibt, oder die Klasse „Math“ mit der Methode „Math.cos“, die einen cosinus-Wert zurückgibt. Zu jeder dieser Bibliotheken gibt es ausführliche Dokumentationen im Internet zu finden: (Oracle Corporation 2014b). Diese Klassen sind in den Entwicklungsumgebungen für Java eingebunden und müssen nicht extra importiert werden.

Externe Bibliotheken, die nicht in Java integriert sind, müssen erst heruntergeladen, in NetBeans importiert und mit den Programmdateien verknüpft werden. Für das Antennenprogramm wurden die in Tabelle 8 aufgelisteten Bibliotheken verwendet.

Tabelle 8: Im Programm verwendete externe Bibliotheken

Bibliothek	Beschreibung
JFreeChart-1.0.19	Erstellen und Darstellung von Diagrammen (Object Refinery Limited 2014b)
JCommon-1.0.23	Modulieren von Diagrammen, genutzt in der Bibliothek JFreeChart (Object Refinery Limited 2014a)
PDFBox-1.8.7	Erstellen und Arbeiten mit PDF-Dokumenten (Apache 2015)

Die Suche nach den passenden Bibliotheken für die unterschiedlichen Zwecke erfolgte über Internetrecherche, erst über eine Stichwortsuche in einer Online-Suchmaschine, z. B. Google, und dann über das Durchforsten der gefundenen Foren und Webseiten. Bevor die Bibliotheken in NetBeans genutzt werden können, müssen sie importiert und in die Anwendung integriert werden. Erst wenn das Programm Zugriff auf alle notwendigen Bibliotheksdateien hat, können deren Funktionen genutzt werden.

3.3 Main class

In der Main-Klasse, der „Haupt“-Klasse, wird gewissermaßen der Startschuss für das Ausführen der Software gegeben. Beim Erstellen der ausführbaren Programmdatei, die aus dem NetBeans-Projekt eine Java-Anwendung macht, wird die Main-Klasse als die Klasse festgelegt, die beim Start der Anwendung als erstes aufgerufen wird. Nur was in dieser Klasse steht, wird ausgeführt.

In den Anfängen der Entwicklung wurden die ersten Methoden noch alle nacheinander in der Mainklasse aufgerufen. Mit steigender Anzahl und Komplexität der Methoden wurden diese dann immer mehr in eigene Klassen ausgelagert, die die Methoden thematisch zusammenfassen. Das Endergebnis ist eine Main-Klasse mit nur einem Methodenaufruf. Dort wird die GUI aufgerufen, von wo aus die komplette Kommunikation von Nutzer und Software stattfindet. Alles Weitere geschieht über die Eingabefelder und damit verknüpfte Aktionen.

3.4 Graphical User Interface

Die Grafische Benutzeroberfläche (GUI) entstand zu Beginn der Entwicklungsphase getrennt vom Quelltext. Mit dem Entstehen von immer mehr Methoden und Klassen im

Quelltext wurden diese parallel dazu in die GUI integriert. Gleichzeitig wurde die GUI laufend dem wachsenden Quelltext angepasst. Beide Programmteile sind miteinander gewachsen.

3.4.1 Design und Entwicklung der GUI

Aus Vorüberlegungen zu einer möglichen Benutzeroberfläche für die neue Antennensoftware existierten schon einige Ideen, z. B. zur Anordnung der Elemente oder zur Verbesserung spezieller Funktionen aus der alten Software. So konnte mit Hilfe von Skizzen ein erster Entwurf entstehen, der dann über die Entwicklungsumgebung NetBeans umgesetzt werden sollte. Dafür war es notwendig, zu recherchieren und kennenzulernen, was es in Java für Möglichkeiten zur grafischen Darstellung von Daten gibt und wie sich diese wie gewünscht einsetzen lassen. Wie für das Erlernen der Sprache dienten dafür vor allem Internetforen, Online-Tutorials und die im Literaturverzeichnis angegeben Fachbücher.

Nach dem Erstellen verschiedener kleiner Übungsanwendungen mit simplen Benutzeroberflächen zum Testen von Buttons, Textfeldern usw. konnte eine erste GUI für die zukünftige Antennensoftware programmiert werden. Anhand dieser Vorlage wurden die Vorstellungen und Zielsetzungen eingebracht, überarbeitet und die Oberfläche immer wieder entsprechend angepasst. In Abbildung 13 wird die Benutzeroberfläche der alten LabVIEW-Software gezeigt.

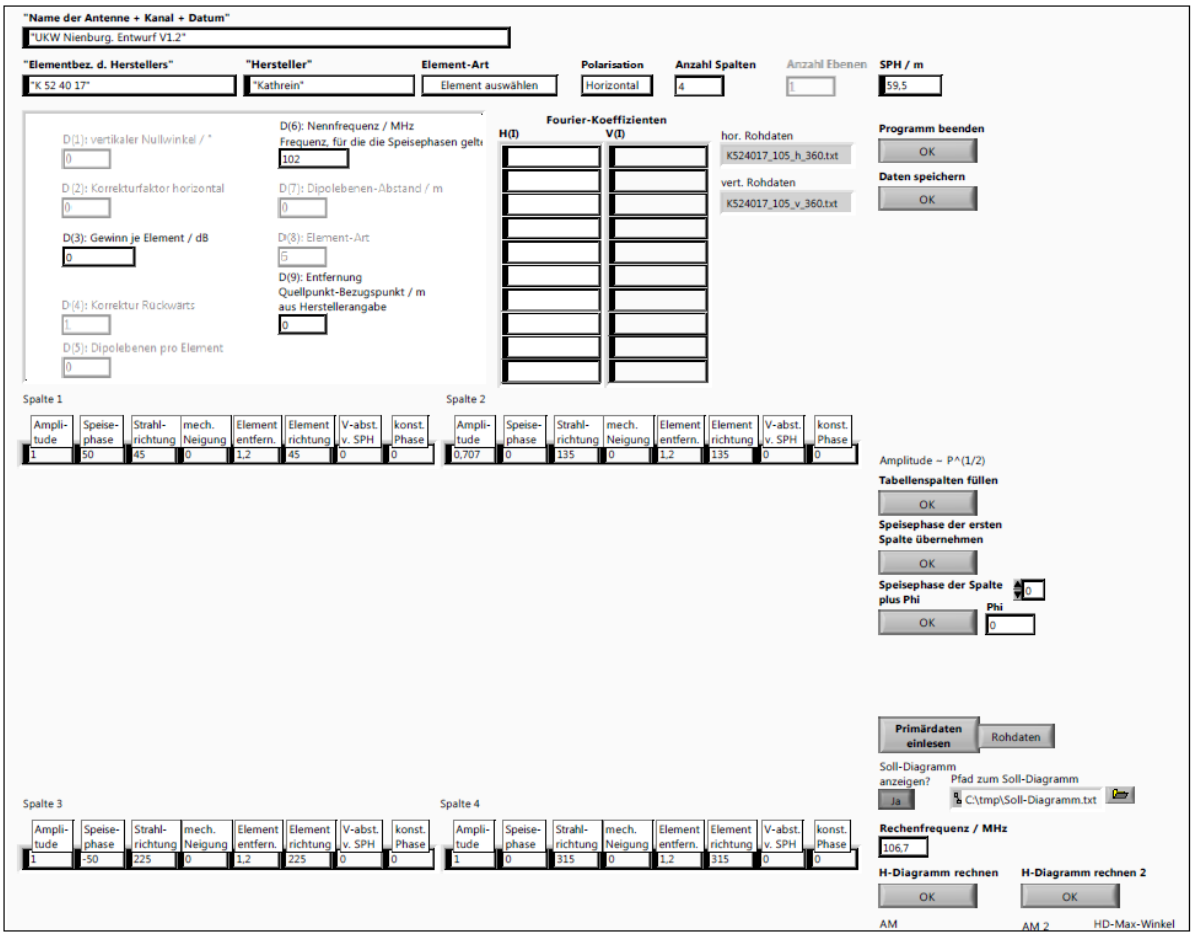


Abbildung 13: Screenshot der alten LabVIEW-Benutzeroberfläche

Im Vergleich dazu ist in Abbildung 14 die neu erstellte Java-GUI nach dem Programmstart und ohne eingegebene Daten zu sehen.

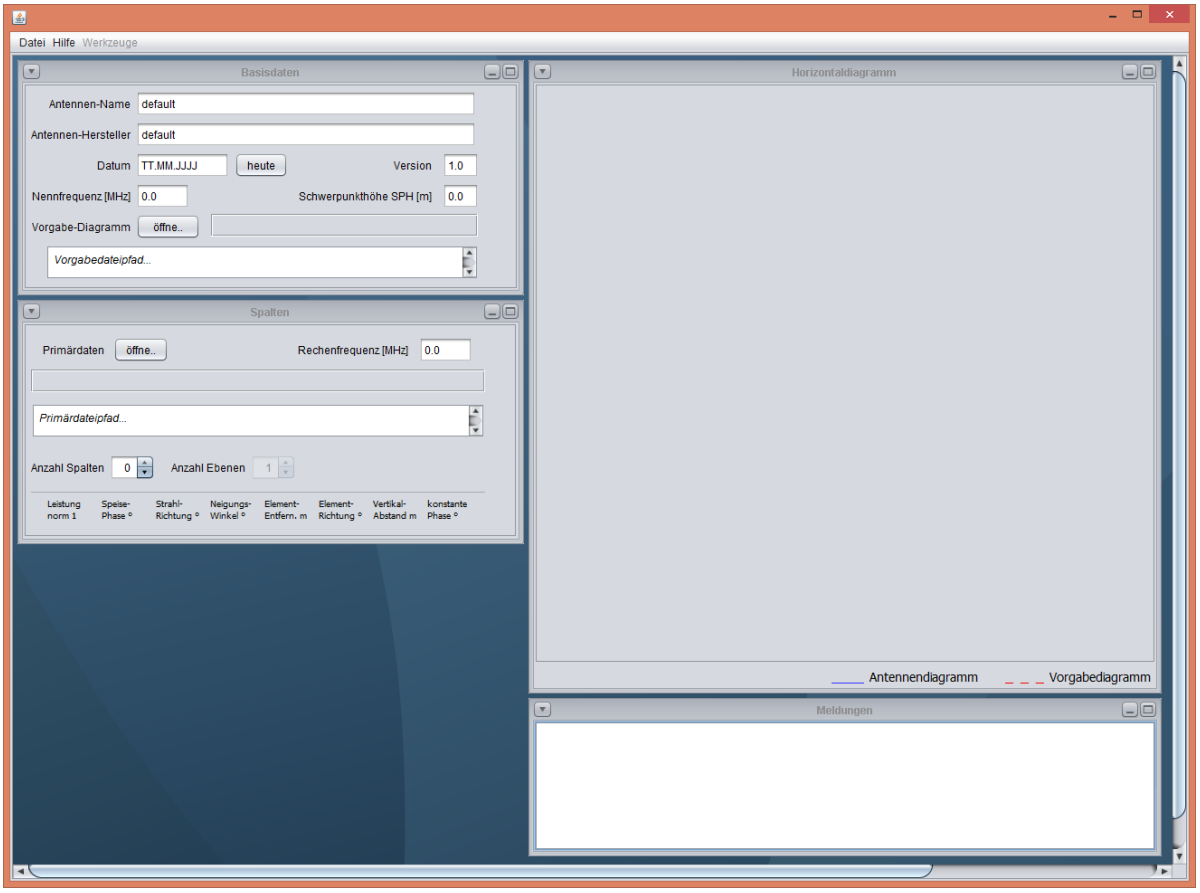


Abbildung 14: Fertige Java-GUI nach der Entwicklungsphase zum Programmstart

Es ist deutlich zu erkennen, dass der Umfang der Eingabefelder auf ein Minimum reduziert wurde. Viele der Funktionen der alten Software sind in der neuen nicht mehr notwendig, da durch detailliertere Antennendaten vom Hersteller z. B. Näherungsrechnungen von Elementdaten mit Hilfe von Fourier-Koeffizienten überflüssig geworden sind. Andererseits sind in der neuen Software manche Funktionen noch nicht eingeflossen, die die Bedienung vereinfachen könnten, wie z. B. das automatische Ausfüllen von Antennenparametern. Im Anhang B befindet sich eine Aufzählung von Ideen für die Weiterentwicklung der Software.

Durch die neue Gestaltung der neuen GUI konnten im Entwicklungsprozess auch die Funktionsweisen und Aufgaben der Software besser nachvollzogen und mit der Bedienung über die Oberfläche in Verbindung gebracht werden. Im Laufe der Entwicklungsphase wurden z. B. immer wieder die Anzahl und die Anordnung von Eingabefeldern angepasst, grafische Elemente wurden entfernt oder hinzugefügt. Daraus entstanden ist die neue Java-GUI in Abbildung 14.

Die Entwicklungsumgebung NetBeans bietet eine intuitive Drag-and-Drop-Funktion, mit der schnell komplette GUIs erstellt werden können. Der dahinterliegende Quelltext wird von NetBeans automatisch erzeugt. Später konnte der Quelltext so weit angepasst werden, dass die Kommunikation zu den manuell erstellten Klassen einwandfrei und im erforderlichen Umfang sichergestellt wurde. Weitere Informationen zu den Elementen der NetBeans-GUI-Funktionen finden sich in (Oracle Corporation 2015).

Die Java-GUI in Abbildung 14, oder in erweiterter Ansicht in Abbildung 15, besteht aus vielen unterschiedlichen Elementen, die weiter unten einzeln erläutert werden. Sie alle entstammen der Bibliothek Swing, siehe dazu (Oracle Corporation 2014a). Den Rahmen, bzw. das Fenster den Hauptprogramms, bildet das JFrame. Es enthält an der oberen Kante die Menüleiste JMenu und ein JScrollPane, ein scrollbares Panel. Die Scrollbalken sind rechts und unten im Fenster zu sehen. Im JScrollPane sind die JInternalFrames eingebettet, unabhängige Fenster für die Basisdaten der Antenne (links oben), die Daten zur Charakterisierung der einzelnen Spalten (links unten), das Diagramm (rechts oben) und das Meldungsfenster (rechts unten). Die JInternalFrames sind mit Textfeldern, Buttons und Eingabefeldern bestückt, die jeweils mit Aktionen belegt wurden, welche den Programmablauf steuern bzw. beeinflussen.

Eine Besonderheit ist der JSpinner, das Zählerfeld im Spalten-Fenster, mit dem man über die Pfeiltasten die Anzahl der Spalten wählen kann. Das Element ist so programmiert, dass beim Klicken die angezeigte Anzahl Felder zur Eingabe der Spalten-Parameter geöffnet oder geschlossen werden. Die Programmierung dieser dynamischen Textfelder ist im Kapitel 3.4.2. beschrieben. In Abbildung 15 sind bspw. drei Spalten geöffnet, die mit ihren Parametern das Horizontaldiagramm (blau) ergeben. Eine Spalte entspricht in der GUI einer Reihe waagerecht nebeneinander angeordneter Eingabefelder. Würden die Spalten senkrecht angeordnet, wie es der Geometrie einer Gruppenantenne entsprechen würde, würde dies zu viel Platz im Fenster einnehmen. Zur Begriffsdefinition siehe Kapitel 2.1. Die gestrichelte rote Linie ist das Vorgabediagramm.

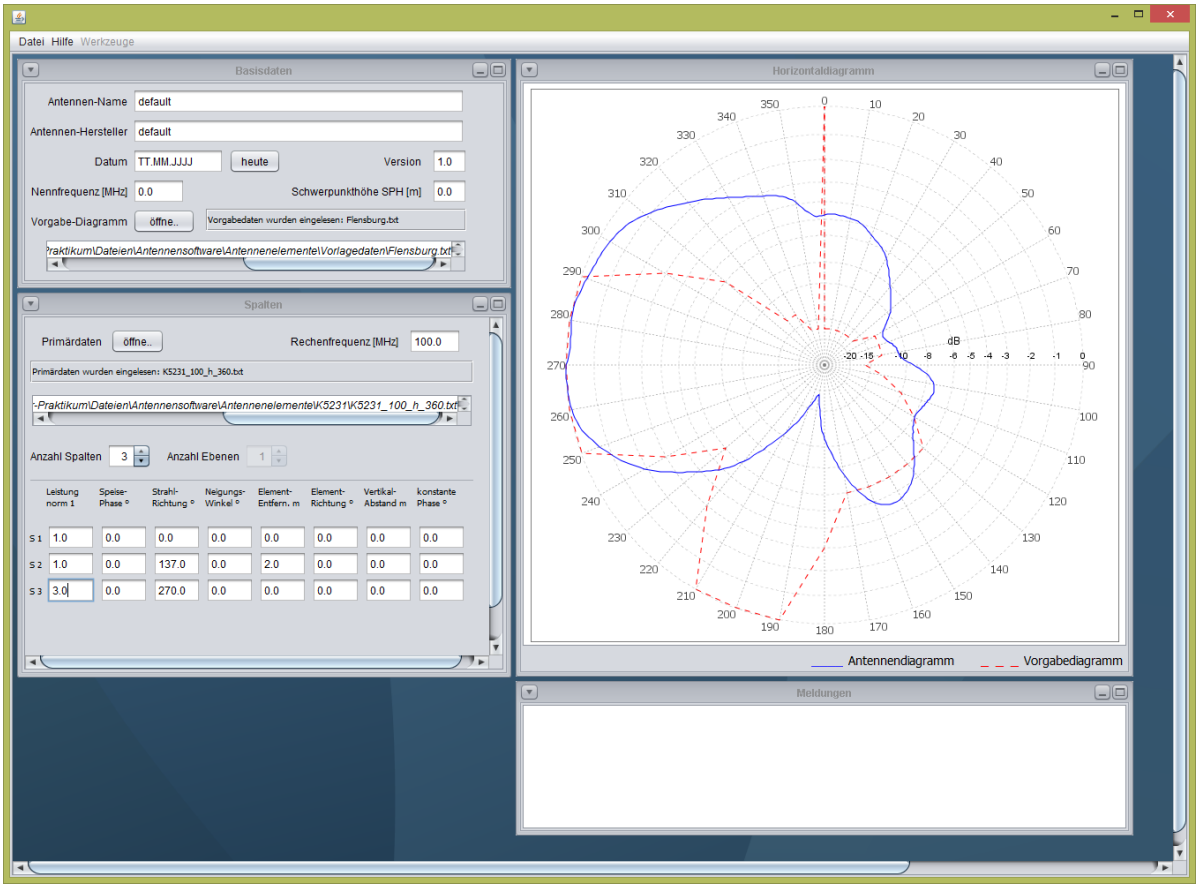


Abbildung 15: GUI mit aufgeklappten Spalten und Diagramm

Alle JInternalFrames können verschoben, in der Größe verändert und an den unteren Bildrand verkleinert werden, falls sie im Hauptfenster stören. Bei Bedarf lassen sich diese wieder vergrößern.

3.4.2 Dynamisches Erstellen von Spalten-Textfeldern

Bei der Auswahl der Spaltenanzahl mit dem JSpinner werden entsprechend viele Zeilen mit Textfeldern zur Eingabe der Spaltenparameter erzeugt bzw. entfernt. In Abbildung 14 sind bspw. drei Spalten ausgewählt, also drei Reihen geöffnet. Zum Erstellen der Textfelder wird mit einer Methode, die beim Klicken der Pfeiltasten aufgerufen wird, ein neues Objekt erzeugt, welches die entsprechenden Textfelder enthält und in der GUI an gewünschter Stelle platziert werden kann. Den dynamisch erzeugten Textfeldern müssen dynamische Variablennamen zugewiesen, bzw. auch wieder entzogen werden. Da Variablennamen nicht über Methoden zurückgegeben werden können, musste ein Umweg über das Erstellen einer Map gewählt werden. Zur näheren Erläuterung von der Verwendung von Maps siehe Kapitel 3.10 *Globale Variablen*.

3.4.3 Verknüpfen der GUI mit dem Rechenalgorithmus

Eine wichtige Aufgabe der Software ist die Zusammenarbeit von GUI und den im Quelltext programmierten Berechnungen. Beide Objekte stehen in ständiger Kommunikation miteinander, welche durch Tastatureingaben und Mausklicks gesteuert wird. Zum Überwachen dieser Aktionen gibt es Methoden der Klasse Swing, sie heißen ActionListener und Erfassen sogenannte ActionEvents.

Es wird bspw. in einem Textfeld die Eingabetaste gedrückt. Der ActionListener detektiert dieses ActionEvent und springt in die Methode, die für dieses bestimmte ActionEvent definiert wurde. Die Aktionen, die im Quelltext der Methode stehen, werden ausgeführt. Das wäre in diesem Fall das Auslesen des Textes, der in dem Feld steht, und das Schreiben auf eine Variable vom Typ String (Zeichenkette). Gleichzeitig springt der Zeiger in der GUI ein Textfeld weiter. Mit diesem ActionEvent ist außerdem eine Methode zum Prüfen des eingegebenen Textes verknüpft, auch sie wird vom ActionListener aufgerufen. So wird ein Wert nur auf die zugewiesene Variable geschrieben, wenn er die Prüfung z. B. des Wertebereichs besteht.

Ein weiteres in NetBeans verfügbares vorprogrammiertes grafisches Element zur Verwendung in der GUI ist der JFileChooser, der ein Dialogfenster öffnet, über das man ein Verzeichnis auswählen kann. In die dazugehörige Methode, die beim Ausführen des JFileChoosers aufgerufen wird, wird programmiert, was mit dem ausgewählten Verzeichnispfad oder dem Namen der ausgewählten Datei geschehen soll. Diese Werte, in diesem Fall Zeichenketten, können in beliebige Variablen geschrieben werden, damit andere Methoden darauf zugreifen können, z. B. zum Weitergeben des Dateinamens an die Methode, die die Datei auslesen soll. Der JFileChooser wird in der Antennensoftware genutzt, um die Vorgabe- und Primärdaten auszuwählen.

Nach gleichem Prinzip funktioniert auch der Befehl zum Speichern oder Öffnen von Projektdaten. Zum Speichern wird der JFileChooser mit der Eigenschaft ShowSaveDialog ausgeführt, zum Öffnen mit ShowOpenDialog. Es erscheint je nach Voreinstellung ein Auswahlfenster mit den entsprechenden Buttons, über die der Benutzer Projekte in der Verzeichnisstruktur des Rechners öffnen oder ablegen kann.

Um die Kommunikation zwischen Benutzer und Anwendung zu erleichtern, wurde ein Meldungsfenster in die GUI eingebaut. Vor allem während der Entwicklungsphase war dies

hilfreich, da dort definierte Fehlermeldungen ausgegeben werden, wenn es z. B. zu Falscheingaben kommt oder Daten fehlen, die zur Berechnung des Horizontaldiagramms nötig sind. In der fertiggestellten Anwendung sollten dort bestenfalls keine Meldungen auftauchen, da die Software fehlerfrei läuft.

3.5 Einlesen der Primärdaten

Die zu Beginn wichtigste erforderliche Funktion der Software war das Importieren von Textdateien. Die Primärdaten liefern die Grundlage für die Berechnung des Horizontaldiagramms und bestehen aus Textdateien. Diese enthalten für eine bestimmte Rechenfrequenz zu jedem Winkel den gemessenen Amplituden- und Phasenwert des Antennenelements, mit dem die Gesamtantenne aufgebaut wird. Die Datei enthält also drei Spalten, die mit einem Tabulator voneinander getrennt sind. Zum Importieren der Daten wurde zu Beginn der Entwicklung eine Klasse zum Einlesen einer Tab-getrennten Textdatei mit einer bestimmten Anzahl von Zeilen und Spalten entwickelt. Die Methoden in dieser Klasse übernahmen das Übertragen der Werte in eine zweidimensionale Matrix sowie das Auslesen und Ausgeben der einzelnen Matrixelemente auf dem Bildschirm. Später wurde die Abfrage ergänzt, ob es sich um horizontale oder vertikale Primärdaten handelt. Die horizontalen Primärdaten beschreiben das horizontale, die vertikalen Primärdaten das vertikale Diagramm. Ein Antennenelement wird durch beide Datensätze eindeutig beschrieben. Da das Horizontaldiagramm einen Bereich von $0 \dots 360\,°$ und das Vertikaldiagramm von $0 \dots 180\,°$ umfasst, und somit die eine Textdatei 360, die andere jedoch nur 180 Zeilen enthält, muss zum Einlesen der Daten eine Differenzierung stattfinden.

Der endgültige Einlesevorgang wird durch den Benutzer über die GUI angestoßen. Wie oben erwähnt, gelten die Primärdaten immer für eine bestimmte Rechenfrequenz. Im Antennenprogramm gibt es ein Feld, in das die geplante Rechenfrequenz eingetragen wird. Dann wird vom Benutzer das Verzeichnis ausgewählt, in dem sich die Primärdateien befinden. Diese enthalten die Rechenfrequenz im Dateinamen, was beim Import der Dateien beachtet werden muss. Um die Dateien fehlerfrei einlesen zu können, muss der Dateiname die folgende Form haben: K5231_100_h_360.txt. Der erste Teil ist die Katalognummer des Antennenelements (vom Hersteller festgelegt), die „100“ steht für eine Rechenfrequenz von 100 MHz, das „h“ steht für horizontal, es handelt sich also um die

Primärdaten des Horizontaldiagramms, und die „360“ steht für den Wertebereich $0 \dots 360\,°$.

Die Anwendung durchläuft das ausgewählte Verzeichnis und sucht von allen Textdateien, die einen Dateinamen im gültigen Format haben, diejenige Datei aus, deren Rechenfrequenz der in der GUI eingetragenen am nächsten ist. Verzeichnispfad und Dateiname werden erzeugt, die Datei wird geöffnet und über eine Methode zeilenweise ausgelesen, zwischengespeichert und in einzelne Elemente geteilt, die in einer Matrix gespeichert werden. Aus dieser Matrix werden die Daten für die Berechnung des Horizontaldiagramms ausgelesen.

3.6 Einlesen und Darstellen der Vorgabedaten

Das Einlesen der Vorgabedaten geschieht auf ähnlichem Weg wie bei den Primärdaten. Ein Unterschied ist, dass die Vorgabedaten in $10°$-Schritten statt in $1°$-Schritten angegeben werden und dass die Textdatei nur eine Spalte für die Gradzahl und eine Spalte für die Amplitude enthält, die Phase entfällt. Um die Vorgabedaten zusammen mit dem berechneten Horizontaldiagramm darzustellen, müssen beide Diagramme im richtigen Maßstab und Wertebereich überlagert werden. Siehe dazu Kapitel 3.8.2 *Darstellung des Diagramms*.

3.7 Rechenalgorithmus

Um den Rechenalgorithmus aus dem ursprünglichen Programm in die neue Software einzubauen, musste er komplett analysiert und in allen Einzelheiten nachvollzogen werden. Auf diese Weise konnte der Rechenalgorithmus optimiert, veraltete Codeelemente weggelassen und seine Funktionalität voll eingearbeitet werden. Die für diese Arbeit relevanten Ergebnisse dieser Analyse wurden im Kapitel 2.3 *Antennenberechnung* mathematisch beschrieben. In diesem Kapitel wird der Algorithmus aus funktionaler Sicht beschrieben.

Um den Programmcode übersichtlicher zu gestalten und ihn besser nachvollziehen zu können, wurden zuerst die Variablennamen neu vergeben bzw. die alten Variablen, die bisher benutzt wurden, umbenannt. Im Anhang C befindet sich eine Variablentabelle, die die alten LabVIEW-Variablen und die neuen Java-Variablen gegenüberstellt. Im Laufe der

Entwicklung der Software wurden auch neue Variablen erstellt, um die überarbeitete Funktionsweise der Software zu ermöglichen.

Der Algorithmus zur Antennenberechnung ist wie im ursprünglichen Programm als Schleifenstruktur aufgebaut. Er liefert als Ergebnis zu jedem Azimutwinkel eine Gesamtamplitude und eine Gesamtphase aller Antennenelemente der Gesamtebene. In Abbildung 16 ist die Anordnung der Schleifen dargestellt. Die fettgedruckten Begriffe werden in den folgenden Unterkapiteln näher erläutert.

Für jeden Azimutwinkel ($aw = 0 \dots 360\ °$)

Für jede Spalte ($s = anzahlSpalten$)

Für jede Ebene (in dieser Arbeit $e = 1$)

- Berechnen des horizontalen **Differenzwinkels** zwischen Azimutwinkel und Strahlrichtung
- **Interpolieren** des horizontalen Amplitudenwerts für den horizontalen Differenzwinkel (s. Gleichung (13) auf S. 25)
- Berechnen des vertikalen **Differenzwinkels** zwischen Erhebungswinkel und Neigungswinkel
- **Interpolieren** des vertikalen Amplitudenwerts für den vertikalen Differenzwinkel (s. Gleichung (13) auf S. 25)
- Berechnen der Amplitude des Antennenelements: **Elementamplitude**
- **Interpolieren** des horizontalen Phasenwerts für den horizontalen Differenzwinkel (s. Gleichung (13) auf S. 25)
- Berechnen der Phase des Antennenelements: **Elementphase**, bestehend aus horizontaler Phase (s. Gleichung (4) auf S. 21), Höhenphase, Speisekabelphase, konstanter Phase
- Umrechnen von Elementamplitude und Elementphase in **Real- und Imaginärteil**
- Aufaddieren jeweils der Real- und Imaginärteile

Umrechnen von aufaddiertem Real- und Imaginärteil in **Gesamtamplitude und -phase**

Abbildung 16: Schleifenstruktur des Rechenalgorithmus

Als Ergebnis der beiden inneren Schleifen erhält man den aufaddierten Gesamtamplituden- und den aufaddierten Gesamtphasenwert aller durchlaufenen Antennenelemente für einen Aufpunkt P am Azimutwinkel φ in Form von Real- und Imaginärteil. Die

übergeordnete Schleife berechnet daraus Gesamtamplitude und Gesamtphase. Diese Berechnung erfolgt nacheinander für alle Azimutwinkel. Nachdem diese Azimutschleife durchgelaufen ist, erhält man jeweils 360 Gesamtamplituden und –phasen, welche ein vollständiges Horizontaldiagramm beschreiben. Diese Werte werden in ein Array geschrieben, auf 1 normiert und dann mit Hilfe weiterer Methoden als Diagramm ausgegeben.

3.7.1 Differenzwinkel

Die Methode zur Berechnung des Differenzwinkels zwischen Azimutwinkel und Strahlrichtung in der Horizontalen, bzw. zwischen Erhebungswinkel und Neigungswinkel in der Vertikalen, wird, wie die Methode zur Interpolation, in der Rechenschleife für jeden Winkel aufgerufen. Der Differenzwinkel wird berechnet und das Ergebnis als Rückgabewert von der Methode zurückgegeben.

3.7.2 Interpolation

Die Methode zur Interpolation des Amplitudenwerts wird in der Rechenschleife für jeden Winkel aufgerufen. Es wird mit einer if-Abfrage geprüft, ob der betroffene Winkel eine gebrochene Zahl ist. Wenn ja, wird die lineare Interpolation durchgeführt und der interpolierte Amplitudenwert zurückgegeben.

3.7.3 Elementamplitude

In der Theorie werden für die Gesamtamplitude die Amplitudenwerte der einzelnen Antennenelemente für einen Azimutwinkel addiert, siehe dazu Kapitel 2.3.1 *Berechnung der Amplitude im Horizontaldiagramm*. Um zu dieser Gesamtamplitude zu gelangen, wird als erstes die Amplitude interpoliert. In Listing 1 ist der Quellcode zur Berechnung aufgeführt.

Listing 1: Java-Quellcode zur Berechnung der interpolierten horizontalen Amplitude

```
interpolierteAmplitudeHorizontal =
interpoliereWert("a", Global.PrimaerdatenHorizontal,
berechneDifferenzwinkel("h", azimutwinkel, Global.strahlrichtung));
```

Der Variable `interpolierteAmplitudeHorizontal` wird der Rückgabewert der Methode `interpoliereWert` zugewiesen. Dieser Methodenaufruf beinhaltet den Parameter `"a"` für

die Interpolation der Amplitude, die Primärdaten `Global.PrimaerdatenHorizontal`, welche eingelesen und als Globalvariable abgelegt sind, und den Differenzwinkel. Dieser wird als Rückgabewert einer weiteren Methode `berechneDifferenzwinkel` aufgerufen. Die Methode zur Berechnung des Differenzwinkels gibt den gebrochenen horizontalen Winkel zurück, das `"h"` steht wieder für horizontal. Zur Berechnung wird die Differenz aus dem Azimutwinkel `azimutwinkel` und der durch den Benutzer in der GUI angegebenen Strahlrichtung `Global.strahlrichtung[s]` gebildet.

Die Berechnung der vertikalen Amplitude in Listing 2 geschieht analog zur horizontalen Amplitude, nur wird hier auf die Primärdaten für das Vertikaldiagramm `Global.PrimaerdatenVertikal` zugegriffen und bei der Berechnung des Differenzwinkels wird ein `"v"` für vertikal angegeben.

Listing 2: Java-Quellcode zur Berechnung der Interpolierten vertikalen Amplitude

```
interpolierteAmplitudeVertikal = interpoliereWert("a",
Global.PrimaerdatenVertikal, berechneDifferenzwinkel("v", erhebungswinkel,
Global.neigungswinkel));
```

Listing 3 zeigt die Berechnung der Elementamplitude.

Listing 3: Java-Quellcode zur Berechnung der Gesamtamplitude

```
elementamplitudeGesamt = interpolierteAmplitudeHorizontal *
interpolierteAmplitudeVertikal * Global.amplitude;
```

Die Amplitude für das Antennenelement, im Quelltext `elementamplitudeGesamt`, erhält man durch Multiplikation der horizontalen Amplitude `interpolierteAmplitudeHorizontal` und der vertikalen Amplitude `interpolierteAmplitudeVertikal` mit der Amplitude, die vom Benutzer als Leistungsfaktor in der GUI eingegeben wird, `Global.amplitude`. Der sich daraus ergebende Wert `elementamplitudeGesamt` wird nun in der for-Schleife aus Abbildung 16 für jedes Antennenelement berechnet und später zusammen mit der Phase über die Umrechnung in kartesische Koordinaten aufaddiert.

3.7.4 Elementphase

Wie bei der Amplitude wird auch die Phase im ersten Schritt interpoliert, siehe Listing 4. Es werden dieselben Methoden aufgerufen, nur diesmal mit dem Parameter `"p"` für Phase.

Listing 4: Java-Quellcode zur Berechnung der interpolierten horizontalen Primärphase

```
interpoliertePhaseHorizontal = interpoliereWert("p",
Global.PrimaerdatenHorizontal, berechneDifferenzwinkel("h", azimutwinkel,
Global.strahlrichtung[s]));
```

Die Berechnung der horizontalen Phase wird in Kapitel 2.3.2 *Berechnung der Phase im Horizontaldiagramm* beschrieben. Der in Listing 5 aufgeführte Quelltext ist eine eins-zu-eins-Übersetzung in Programmiersprache. Daher werden hier nur die Quelltext-Elemente beschrieben, die hinzugefügt wurden, um den Rechenalgorithmus für die horizontale Phase zu implementieren.

Listing 5: Java-Quellcode zur Berechnung der horizontalen Phase

```
horizontalePhase =
Global.gradZuRad * Global.elementenfernung[s] * Global.kRechen *
Math.cos(Global.gradZuRad *(azimutwinkel - Global.elementrichtung[s])) *
Math.cos(Global.gradZuRad * erhebungswinkel) + Global.gradZuRad *
interpoliertePhaseHorizontal * Math.cos(Global.gradZuRad *
erhebungswinkel);
```

Die Variable `Global.gradZuRad` ist der Faktor zur Umrechnung von der Gradzahl ins Bogenmaß. Der Faktor `Global.kRechen` beschreibt die Kreiswellenzahl, in der theoretischen Berechnung in Kapitel 2.3.2 in seinen Einzelteilen $\frac{360°}{c} \cdot Rechenfrequenz$ zu finden. Der Ausdruck `Math.cos` gibt den Cosinuswert des in den Klammern stehenden Parameters zurück.

Listing 6: Java-Quellcode zur Berechnung der horizontalen Gesamtphase

```
elementphaseGesamt = horizontalePhase + hoehenphase + speisekabelphase +
Global.gradZuRad * Global.konstantePhase[s];
```

Die Phase eines Antennenelements, hier `elementphaseGesamt`, beinhaltet die `horizontalePhase`, die `hoehenphase`, welche vom Vertikalabstand des Elements vom Schwerpunkt der Antenne und dem Erhebungswinkel beeinflusst wird, die `speisekabelphase`, welche von der Rechenfrequenz und der Speisephase des Kabels bei der Nennfrequenz abhängig ist, sowie die konstante Phase `Global.konstantePhase[s]`, die vom Benutzer in der GUI eingegeben wird, siehe Listing 6.
Wie bei der Amplitude wird die Phase für jedes Element berechnet und dann aufaddiert.

3.7.5 Rechnen mit Real- und Imaginärteil

In Kapitel 0 wird der mathematische Hintergrund der Umrechnung von kartesischen in Polarkoordinaten und zurück beschrieben. In Listing 7 ist der Quelltext zu sehen, mit dem die Umrechnung von Amplitude und Phase in Real- und Imaginärteil im Programm realisiert wurde.

Listing 7: Java-Quellcode zur Berechnung von Real- und Imaginärteil

```
realteil = elementamplitudeGesamt * Math.cos(elementphaseGesamt);
imaginaerteil = elementamplitudeGesamt * Math.sin(elementphaseGesamt);
```

Die Elementamplitude `elementamplitudeGesamt` und die Elementphase `elementphaseGesamt` werden mit Hilfe der Methoden `Math.cos` und `Math.sin` in kartesische Koordinaten umgerechnet. Die erste Gleichung entspricht dabei der Gl. (14), die zweite Gleichung entspricht der Gl. (15) aus Kapitel 0, beide auf S. 26.
Zum Aufaddieren der Werte wird am Ende jeder inneren Schleife zur Berechnung für ein Element der neu berechnete Wert zu dem vorherigen Wert hinzuaddiert, s. Listing 8.

Listing 8: Java-Quellcode zum Aufaddieren von Real- und Imaginärteilen

```
realteilGesamt = realteilGesamt + realteil;
imaginaerteilGesamt = imaginaerteilGesamt + imaginaerteil;
```

So erhält man am Ende der Schleife die Gesamtwerte für alle berechneten Antennenelemente.

3.7.6 Gesamtamplitude und -phase

Nachdem die Schleifen durchlaufen sind, ergeben sich eine Gesamtamplitude und eine Gesamtphase, die das Antennensystem an einem bestimmten Azimutwinkel beschreiben. Sie werden von kartesischen zurück in Polarkoordinaten umgerechnet. Der Quellcode dazu ist in Listing 9 aufgeführt.

Listing 9: Java-Quellcode zum Zurückrechnen in Amplitude und Phase

```
diagrammamplitude = Math.sqrt(realteilGesamt * realteilGesamt +
imaginaerteilGesamt * imaginaerteilGesamt);

diagrammphase = Math.atan2(imaginaerteilGesamt,realteilGesamt) /
Global.gradZuRad;
```

Die Methode `Math.sqrt` berechnet die Wurzel aus der Summe des quadrierten Real- und Imaginärteils, die Methode `Math.atan2` berechnet den Tangens unter Berücksichtigung der Fallunterscheidung, die in Kapitel 0 auf S. 27 beschrieben wird.

Die erste Gleichung zur Berechnung der Gesamtamplitude, im Quelltext `diagrammamplitude`, entspricht der Gl. (16), die zweite Gleichung zur Berechnung der Gesamtphase, im Quelltext `diagrammphase`, entspricht der Gl. (17) aus Kapitel 0, s. S. 27.

Diese Berechnung wird für jeden einzelnen Azimutwinkel $0 \ldots 360\,°$ durchgeführt, die Ergebnisse werden in eine Matrix geschrieben, mit welcher im nächsten Schritt zur Darstellung des Diagramms weitergearbeitet wird.

3.8 Horizontaldiagramm

3.8.1 Normierung der Amplitude

Zur Normierung der Amplitudenwerte auf 1 wird die Matrix, welche die berechneten Gesamtamplituden- und phasenwerte enthält, mit einer for-Schleife durchlaufen, um den auftretenden Maximalwert festzustellen. Dafür wird jeder Wert mit dem vorherigen verglichen und in eine Variable für den Maximalwert geschrieben, wenn dieser größer ist. Wenn der Maximalwert feststeht, wird die Matrix erneut durchlaufen, jeder Wert durch den Maximalwert geteilt und in eine neue Matrix für die normierten Werte geschrieben. Zur Darstellung werden diese normierten Werte ausgelesen und daraus ein Diagramm generiert.

Das Gleiche wird auch mit den Vorgabedaten gemacht. Nachdem die Werte aus der Textdatei in eine Matrix importiert wurden, wird diese Matrix auf dieselbe Weise normiert und neu abgespeichert.

3.8.2 Darstellung des Diagramms

Für Java gibt es verschiedene Möglichkeiten, Diagramme zu erstellen und auf dem Bildschirm auszugeben. Für die Darstellung kartesischer Koordinatensysteme gibt es dazu viele nützliche Bibliotheken und Methoden, um Kurvendarstellungen zu modellieren und zu individualisieren. Für Polarkoordinaten ist es schon schwieriger, geeignete Vorlagen zu finden, doch letztendlich konnte mit Hilfe der Bibliotheken JFreeChart und JCommon, siehe Kapitel 3.2 *Bibliotheken*, ein geeignetes Diagramm mit den gewünschten Kurveneigenschaften, Achsenbeschriftungen, Hilfslinien und der Überlagerung des Vorgabe- und des Horizontaldiagramms entwickelt werden.

Aus den normierten Amplitudenwerten der Vorgabedaten sowie der berechneten Antennendaten werden zwei Datenreihen erstellt, das Dataset, aus dem das Diagramm generiert wird. In dem Diagramm wird ein Graph erstellt, welchem das Dataset zugewiesen wird. Nun wird das Diagramm in einem Rahmen eingebettet, um es in der GUI sichtbar zu machen. Um den Graphen und das Diagramm in der Darstellung anzupassen, wird ein sogenannter Renderer genutzt, über den Einstellungen wie Liniendicke, Achsenbeschriftung usw. programmiert werden können. All diese Einzelschritte werden jeweils von einer eigenen Methode ausgeführt, die aus der übergeordneten Methode aufgerufen wird. So kann bei Ausführen einer Aktion durch den Benutzer, die das Anzeigen des Diagramms auslöst, mit dem Aufruf einer einzigen Methode, welche alle anderen Methoden aufruft, das Diagramm ausgegeben werden.

3.9 Speichern und Einlesen von Projektdaten

Um ein Projekt zu speichern und es später wieder aufrufen zu können, werden die Daten, die in die Eingabe-Oberfläche eingetragen werden, in einer Textdatei als Tab-getrennte Werte gespeichert. So müssen nicht jedes Mal alle Werte neu eingegeben werden. Ein vorher gespeichertes Projekt kann über „Menü - Öffnen..“ in die Antennensoftware importiert werden. Dabei wird die ausgewählte Textdatei eingelesen und die enthaltenen Werte in die Textfelder der GUI geschrieben. Darunter sind auch die Verzeichnispfade für die Vorgabe- und die Primärdaten. Dann wird jedes Textfeld neu ins Programm eingelesen, als hätte der Benutzer die Eingaben per Hand gemacht und bestätigt. So wird jeder Wert erneut auf seine Gültigkeit geprüft und das Horizontaldiagramm neu berechnet.

Sollen nicht nur die Projektdaten, sondern auch die berechneten Daten des Horizontaldiagramms gespeichert werden, können diese ebenfalls in eine Textdatei exportiert werden. Diese kann später in andere Programme zur weiteren Verarbeitung importiert werden.

Zum Speichern des Diagramms als PDF-Datei kann der Menüpunkt „exportieren - PDF-Diagramm“ genutzt werden. Dabei wird eine Datei erstellt, die Antennennamen, Datum, Version und das Horizontaldiagramm enthält. Dies macht Sinn, wenn zu den erstellten Diagrammdaten eine Übersicht benötigt wird, um den Daten ein Bild zuzuordnen.

3.10 Globale Variablen

Das Programm wurde im Laufe der Entwicklung in immer mehr Klassen und Pakete unterteilt, um übersichtlich zu bleiben. Auf die Variablen, die über die GUI vom Benutzer eingegeben und geändert werden, wird von vielen unterschiedlichen Methoden zugegriffen, um damit Berechnungen durchzuführen oder sie zu modifizieren. Es wurden also globale Variablen benötigt, die aus allen Paketen und Klassen erreichbar waren. Zu diesem Zweck wurde eine Klasse erstellt, die alle Variablen beinhaltet, die im Programmablauf dauerhaft verändert werden. Darunter fallen z. B. alle Variablen, Arrays und Matrizen, in welche die Antennendaten geschrieben werden. In Java können Variablen als „public“ (öffentlich) deklariert werden, damit so ein Zugriff von außen möglich ist. Ein weiterer Vorteil der Globalvariablen ist, dass sie mit Defaultwerten belegt werden können, also mit Standardwerten für den Fall, dass noch kein spezieller Wert zugewiesen wurde.

Im Kapitel 3.4.2 wird beschrieben, dass über einen Zähler mit Pfeiltasten Felder für die Spaltenparameter hinzugefügt und entfernt werden können. Wenn so ein Feld erstellt wird, müssen jedem darin enthaltenen Textfeld Variablennamen zugeordnet werden, um die Daten, die später in die Textfelder eingegeben werden, eindeutig zu identifizieren. Diese Variablennamen werden über eine Methode anhand des Namens des Textfeldes erstellt. Die Schwierigkeit hierbei war, dass Variablen nicht in einer Methode übergeben werden können, denn aus den Textfeldnamen, die als String übergeben werden, können keine Variablen erstellt werden. Um dieses Problem zu lösen, wurde eine Map benutzt.
Eine Map ist eine Liste von Schlüsselworten und ihnen zugeordneten Werten. Ihre Funktionsweise ist an einem Beispiel in Abbildung 17 dargestellt.

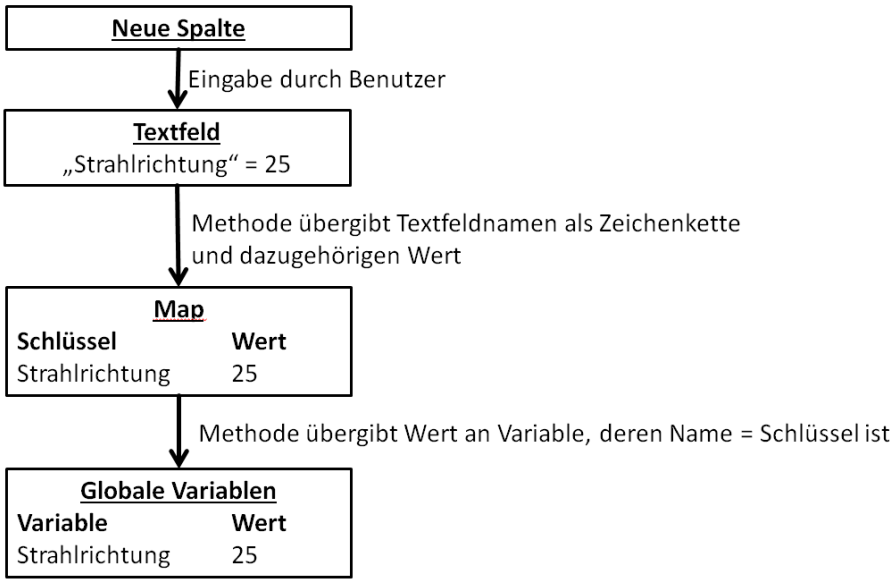


Abbildung 17: Werteübergabe mit Hilfe einer Map

Anstatt beim Erstellen einer neuen Antennenspalte einen neue Globale Variable für jedes Textfeld zu erstellen, wird in der Map ein Schlüsselwort, der Variablenname, mit einem dazugehörigen Wert belegt. Eine Methode sorgt dafür, dass die Werte aus der Map regelmäßig in ein vorher definiertes globales Array mit beliebig vielen Elementen geschrieben werden. So gelangen die Werte aus den Spalten mit Hilfe von Methoden über den Umweg der Map zu den globalen Variablen und können dort von anderen Methoden abgerufen und weiterverarbeitet werden. Dies mag kompliziert wirken, ist jedoch, einmal entwickelt, auf beliebig viele Spalten anwendbar. Die Alternative wäre das manuelle

Erstellen hunderter Platzhalter-Variablen, unabhängig davon, wie viele tatsächlich gebraucht werden.

3.11 Abfangen von Fehlern

Die Software arbeitet mit den Daten, die der Nutzer über die GUI auswählt oder eingibt. Fehlerhafte Eingaben führen dazu, dass Methoden nicht funktionieren und das Programm abstürzt. Nur durch das konsequente Abfangen von allen möglichen Fehleingaben kann das verhindert werden. Der Programmcode wurde deshalb mit einem Abfrage-System gespickt, mit dem jede Eingabe und auch jede Durchführung von Methoden überprüft und eine entsprechende Meldung in Form einer Prüfvariable weitergegeben wird. Um den Benutzer dabei zu unterstützen, Fehleingaben zu korrigieren, werden für bestimmte Fehler informative Meldungen in der GUI ausgegeben oder erscheinen als Dialogfenster auf dem Bildschirm. In Abbildung 18 ist z. B. eine Meldung zu sehen, die erscheint, wenn beim Öffnen einer Projektdatei eine Datei ausgewählt wird, die nicht die Endung .txt hat.

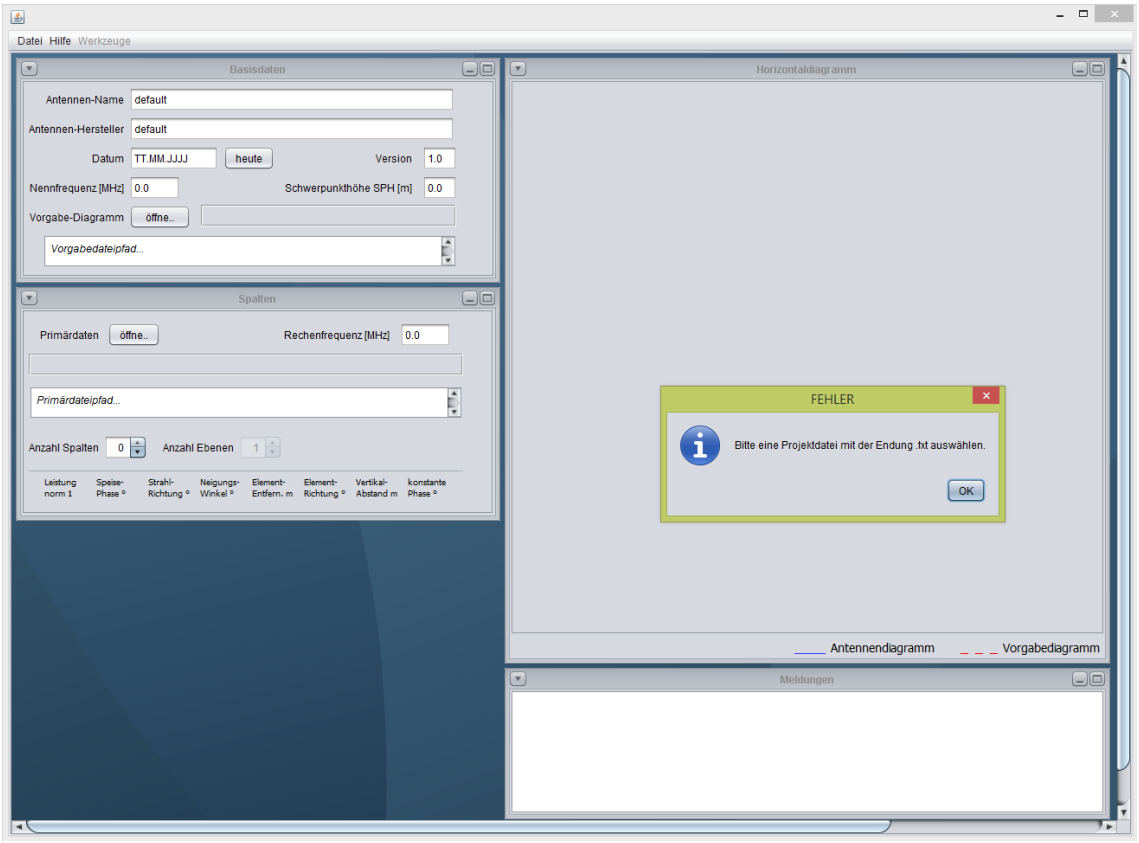


Abbildung 18: Fehlermeldung beim Öffnen einer Projektdatei mit falscher Dateiendung

Das Abfangen von Fehlern ist an vielen Stellen im Rechenalgorithmus implementiert. Durch das Rechnen mit Winkeln und Wertebereichen muss immer kontrolliert werden, ob diese Bereiche im Laufe der Berechnungen überschritten wurden und eventuelle korrigierende Eingrenzungen durchgeführt werden müssen. Übersteigt der Azimutwinkel z. B. durch Addition $360\,°$, muss mit $360\,°$ subtrahiert werden, da sonst Folgefehler und nicht eindeutige Werte auftauchen können. Das Diagramm wird für den Bereich von $0 \ldots 360\,°$ ausgegeben. Ein Winkel von bspw. $460\,°$ entpricht einem Winkel von $100\,°$, muss aber in den richtigen Bereich von $0 \ldots 360\,°$ umgerechnet werden, um korrekt dargestellt werden zu können.

Auch wichtig ist die Abfrage nach dem Vorhandensein aller notwendigen Daten zum Berechnen und Ausgeben des Horizontaldiagramms. Würde das Programm bei unvollständigen Daten die Berechnung durchführen, käme es zu fehlerhaften Ergebnissen und das Diagramm würde, wenn überhaupt, falsch dargestellt werden.

Durch eine konsequente Fehlereingrenzung und die entsprechende Maßnahme zum Abfangen dieses Fehlers bei wiederholtem Auftreten konnten mit der Zeit viele Fehlerquellen eliminiert werden. Es kann trotzdem passieren, dass Benutzer Wege finden, das Programm so zu bedienen, dass Fehler entstehen, die bei der Entwicklung nicht bedacht wurden, und das Programm als Folge zum Absturz gebracht wird. In diesem Fall kann im Quellcode jederzeit an entsprechender Stelle nachgerüstet und Fehlerabfragen ergänzt werden. Wie bei jeder neuen Software braucht es eine gewisse Einarbeitungszeit, in der solche Mängel aufgedeckt und behoben werden können.

3.12 Überarbeiten der Programmstruktur

Bevor der Quellcode seine endgültige Form bekam, entstand durch das Erstellen, Erweitern, Verschieben und Auslagern von Klassen und Methoden während des Entwicklungsprozesses ein unübersichtliches Gerüst ohne einheitliche Struktur. Deshalb wurde, nachdem alle Funktionen der Software fertiggestellt waren, eine thematische Ordnung entwickelt, nach der der Programmcode strukturiert wurde. Es entstanden fünf Pakete, die jeweils mehrere Klassen enthalten, welche wiederum mehrere Methoden enthalten. Die Beschreibung der Pakete ist in Tabelle 9 aufgeführt.

Tabelle 9: Beschreibung der Pakete

Paketname	**Beschreibung**
Main	Start des Programmablaufs, Aufbau der GUI
Variablen	Definition der globalen Variablen, Map
Oberflaeche	Alle GUI-Elemente, Klassen für JFrames, JInternalFrames usw.
Berechnungen	Einlesen der Vorgabe- und Primärdaten, Berechnung und Ausgabe des Horizontaldiagramms
ExportUndSpeichern	Speichern und Einlesen der Projektdaten, Speichern der PDF-Datei

Verschiedene Diagramme der Programmstruktur mit allen Elementen befinden sich im Anhang D.

3.13 Testen und Verifizieren der Software

Das Testen der Software fand teilweise schon während des Entwicklungsprozesses statt, da jede neu integrierte Methode erst dann fertiggestellt war, wenn sie ihren Zweck fehlerfrei erfüllte. Um die Software als Ganzes und die korrekte Berechnung und Darstellung des Horizontaldiagramms zu testen, wurden Diagramme mit definierten Parametern im alten LabVIEW-Programm erstellt und dann mit dem Ergebnis der neuen, fast fertiggestellten Software bei gleichen Parametern verglichen. In Abbildung 19 ist ein solcher Vergleich dargestellt. Hier wurde mit gleichen Werten in der alten Software (a) und in der neuen Software (b) ein Horizontaldiagramm erstellt. Die durchgezogene Linie in Abbildung (b) zeigt das Horizontaldiagramm, die gestrichelte Linie stellt das Vorgabediagramm dar, welches in Abb. a weggelassen wurde und in diesem Kontext keiner Beachtung bedarf. Zusätzlich wurden die Amplitudenwerte beider Diagramme tabellarisch miteinander verglichen. Dabei wurden weder qualitativ noch quantitativ signifikante Unterschiede festgestellt.

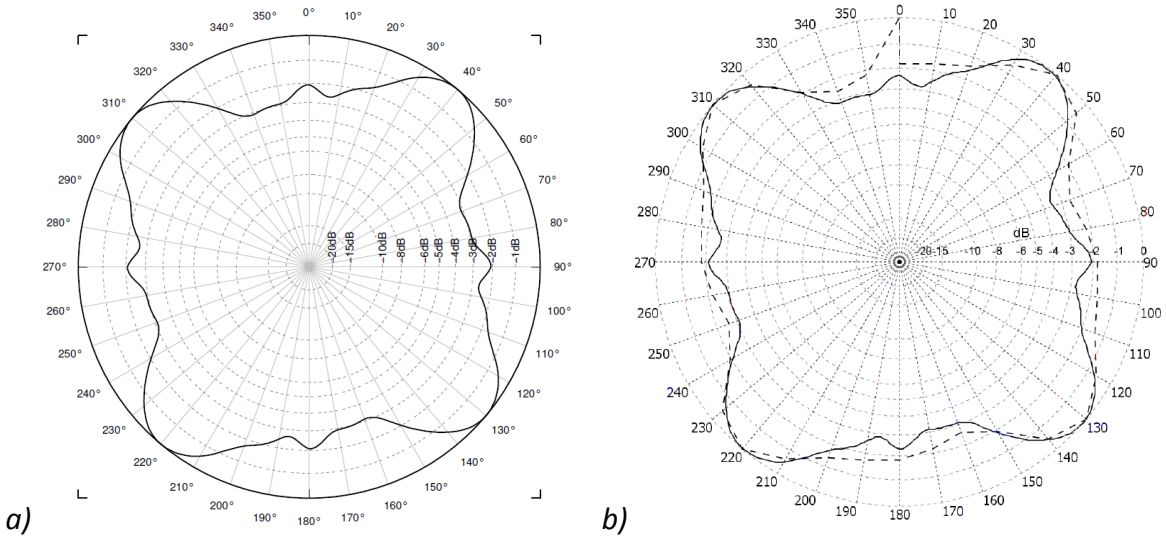


Abbildung 19: Vergleich Horizontaldiagramme a) alte LabVIEW-Software, b) neue Java-Software

Die Berechnung der Antennendaten und die Darstellung im Diagramm wurde also in unterschiedlichen Szenarien getestet, verglichen und für verifiziert erklärt.

Auch das Speichern und Öffnen von Projektdaten wurde ausgiebig getestet und Fehlerszenarien simuliert. Projekte können nun erfolgreich abgespeichert und in einem leeren Projekt geöffnet werden, das Diagramm wird korrekt erzeugt und die Daten richtig eingelesen.

Damit ist die neue Java-Software auf die geforderte Funktion verifiziert und getestet.

4 Zusammenfassung und Ausblick

Das Ziel dieser Arbeit war die Entwicklung einer Software, mit der Horizontaldiagramme von Antennen erstellt, angepasst und dokumentiert werden können. Dafür wurde eine vorhandene, aber veraltete Software in ihren Grundzügen übernommen, in eine neue Programmumgebung übertragen und dort erweitert und optimiert. Es ist eine übersichtliche und stabile Anwendung entstanden, welche die Grundanforderungen, eine benutzerfreundliche grafische Benutzeroberfläche (GUI), und die korrekte Berechnung und Darstellung eines Horizontaldiagramms erfüllt. Alle Funktionalitäten, die in die Anwendung integriert sind, sind ausgiebig getestet und funktionieren verlässlich. Vergleichstests mit den Horizontaldiagrammen der alten Software weisen keine nennenswerten Unterschiede auf.

Der Rechenalgorithmus zur Berechnung von Amplitude und Phase der Antennenelemente, das Herzstück der alten sowie der neuen Software, wurde mit neuen Variablennamen und in schlankerer Form integriert. Durch den objektorientierten Aufbau der Programmstruktur ließ der Rechenalgorithmus sich ideal mit den neu entwickelten Funktionen verknüpfen. Die zur Berechnung des Diagramms notwendigen Daten werden vom Benutzer über die GUI eingegeben und über ein Gerüst von Klassen, Methoden und Variablen zu einem Diagramm verarbeitet, welches bei jeder Änderung neu berechnet und ausgegeben wird.
In Zusammenarbeit mit zukünftigen Anwendern wurde die grafische Oberfläche auf das Wichtigste reduziert und ermöglicht im Vergleich zu der alten Software eine intuitive und einfache Bedienung. Insgesamt ist eine übersichtliche GUI entstanden, die es ermöglicht, Antennenparameter einzugeben und den Einfluss von Veränderungen augenblicklich im ständig neu berechneten Horizontaldiagramm zu verfolgen. Die Darstellung des Diagramms wurde optisch der Ausgabe der alten Software angepasst, neben dem Horizontaldiagramm wird das Vorgabediagramm angezeigt, um die Anpassung des Horizontaldiagramms an die Vorgaben zu erleichtern. Um die Daten zu dokumentieren, wurden die Option zum Speichern und Öffnen der Projektdatei sowie die Ausgabe des Horizontaldiagramms als Textdatei sowie als PDF-Datei programmiert. So können Projekte abgelegt und später

weiter bearbeitet werden, es können so auch Zwischenstände über den Datenexport gespeichert und miteinander verglichen werden.

Zur Erstellung von Antennensystemen können beliebig vielen Spalten erstellt, entfernt und modifiziert werden. Paramater wie die Rechenfrequenz können verändert werden, die Software passt die importierten Primärdaten entsprechend an und berechnet das Diagramm neu.

Mit Hinblick auf die langfristige Nutzung des Programms ist Raum für Weiterentwicklungen. Am wichtigsten ist das Erweitern auf die Berechnung des Vertikaldiagramms, dessen Darstellung und die Berechnung des Antennengewinns. Diese Punkte sind notwendig zur vollständigen Planung einer Antenne. Auch die Vorschläge in der Liste *Zielsetzung und Ideen* im Anhang B bieten Ansätze für die Optimierung der Anwendung.

Literaturverzeichnis

Abts, Dietmar (2013): Grundkurs JAVA. Von den Grundlagen bis zu Datenbank- und Netzanwendungen ; mit … 22 Abbildungen, 167 Programmbeispielen und 133 Aufgaben. 7., aktualisierte Aufl. Wiesbaden: Springer Vieweg.

Apache (2015): Apache PDFBox - A Java PDF Library. Bibliothek. Copyright © 2009-2015 The Apache Software Foundation, Licensed under the Apache License, Version 2.0. Online verfügbar unter https://pdfbox.apache.org/.

Balzert, Helmut; Priemer, Jürgen (2014): Java: Anwendungen programmieren. Von der GUI-Programmierung bis zur Datenbank-Anbindung ; mit Java 8. 3. Aufl. Dortmund: W3L-Verl (Informatik).

Goll, Joachim (2014): Architektur- und Entwurfsmuster der Softwaretechnik. Mit lauffähigen Beispielen in Java. 2., aktualisierte Aufl. 2014. Wiesbaden: Springer Vieweg (SpringerLink : Bücher).

Goll, Joachim; Heinisch, Cornelia (2014): Java als erste Programmiersprache. Ein professioneller Einstieg in die Objektorientierung mit Java. In: *Java als erste Programmiersprache*.

Gotthard, Othmar (1989): FM- und TV- Sendeantennensysteme: Kathrein-Werke KG. Online verfügbar unter http://books.google.de/books?id=kWrONAAACAAJ.

Mössenböck, Hanspeter (2014): Sprechen Sie Java? Eine Einführung in das systematische Programmieren. 5., überarb. u. erw. Aufl. Heidelberg: Dpunkt.

Niemann, Alexander (2003): Das Einsteigerseminar objektorientierte Programmierung in Java. [der methodische und ausführliche Einstieg ; über 350 Seiten Einsteiger-Know-how]. 3., überarb. Aufl. Bonn: Bhv (Das @Einsteigerseminar).

Object Refinery Limited (2014a): JCommon. Bibliothek. © 2007-2014 Object Refinery Limited. Online verfügbar unter http://www.jfree.org/jcommon/.

Object Refinery Limited (2014b): JFreeChart. Bibliothek. © 2005-2014 Object Refinery Limited. Online verfügbar unter http://www.jfree.org/jfreechart/.

Oracle Corporation (2014a): Java Bibliothek javax.swing. Package javax.swing. Copyright © 1993, 2014, Oracle and/or its affiliates. All rights reserved. Online verfügbar unter http://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html.

Oracle Corporation (2014b): Java™ Platform, Standard Edition 7 API Specification. This document is the API specification for the Java™ Platform, Standard Edition. Copyright © 1993, 2014, Oracle and/or its affiliates. All rights reserved. Online verfügbar unter http://docs.oracle.com/javase/7/docs/api/.

Oracle Corporation (2014c): Java™ Platform, Standard Edition 7 API Specification - Class Math. This document is the API specification for the Java™ Platform, Standard Edition. Copyright © 1993, 2014, Oracle and/or its affiliates. All rights reserved. Online verfügbar unter http://docs.oracle.com/javase/7/docs/api/java/lang/Math.html.

Oracle Corporation (2015): Designing a Swing GUI in NetBeans IDE. Tutorial. © 2015, Oracle Corporation and/or its affiliates. Online verfügbar unter https://netbeans.org/kb/docs/java/quickstart-gui.html.

Papula, Lothar (2014a): Mathematik für Ingenieure und Naturwissenschaftler Band 1. Ein Lehr- und Arbeitsbuch für das Grundstudium. 14., überarb. u. erw. Aufl. 2014. Wiesbaden: Springer Vieweg (SpringerLink : Bücher).

Papula, Lothar (2014b): Mathematische Formelsammlung für Ingenieure und Naturwissenschaftler. Mit über 400 Abbildungen, zahlreichen Rechenbeispielen und einer ausführlichen Integraltafel. 11., überarb. Aufl. Wiesbaden: Springer Vieweg (SpringerLink : Bücher).

Sengelmann, Sabine (2010): Entwicklung einer Software zur Berechnung der Strahlungscharakteristik von Gruppenantennen. Bachelorthesis. Hochschule für Angewandte Wissenschaften, Hamburg. Fakultät Technik und Informatik, Department Informations- und Elektrotechnik.

Ullenboom, Christian (2003): Java ist auch eine Insel. [Programmieren für die Java-2-Plattform in der Version 1.4]. 3. Aufl. Bonn: Galileo Press (Galileo computing).

Anhang

Anhang A Pro- und Contraliste zur Auswahl von Sprache und Software

Tabelle 10: Anhang A Pro- und Contraliste zur Auswahl der Programmiersprache

C++ / Qt

Pro	Software	• kostenlos • Drag & Drop • Mit Lizenz: Source Code muss nicht offengelegt werden
	Sprache	• Programmierung auf hohem Abstraktionsniveau • Objektorientiert • Umfangreiche Programmiermöglichkeiten
	Anmerkung	• Schon im Studium kennengelernt • Software getestet: klappt mittelgut, d. h. ist wenig intuitiv, erste Versuche waren mühsam, keine zufriedenstellenden Ergebnisse
Contra	Software	• Drag & Drop für C++ in Visual Studio nicht verfügbar seit 2010, stattdessen in C# • C#: läuft nur unter Windows
	Sprache	• Mehr Fehlerquellen als Java (z.B. Zeiger)
	Anmerkung	• Evtl. zu umfangreich
	Sprache	• Reduzierter Befehlsumfang
	Anmerkung	• Einarbeitungsaufwand minimal höher als für C++

Tabelle 11: Anhang A Pro- und Contraliste zur Auswahl der Entwicklungsumgebung

Java / NetBeans

Pro	Software	• Kostenlos • Drag & Drop • Ideal für GUIs bzw. gängiger als C++ • Läuft auf allen weit verbreiteten Betriebssystemen mit installierter Laufzeitumgebung
	Sprache	• laut Internetrecherche einfacher und sicherer als C++ • Objektorientiert • Keine Mehrfachvererbung von Klassen, keine Zeigerarithmetik, dadurch weniger Fehlerquellen als C++ • Unterstützt nebeneinander ablaufende Programmteile (Threads)
	Anmerkung	• Für den Zweck scheinbar besser geeignet • Software getestet: klappt sehr gut, erste Testanwendungen konnten problemlos erstellt werden • Gute Literatur vorhanden (Hochschulbibliothek)
Contra	Software	• Laufzeitumgebung benötigt

Fazit

Da die Vorteile der Kombination aus der Programmiersprache Java und der Software NetBeans gegenüber der Alternative mit C++ und Qt überwiegen, wurden sie für die Bearbeitung des Projektes ausgewählt.

Anhang B Zielsetzung und Ideen

Tabelle 12: Anhang B Ziele für diese Arbeit

Thema	Beschreibung
Einlesen von Vorgabe- und Primärdaten	• einheitliche Benennung der Dateien, um sie einzulesen und Informationen aus dem Dateinamen zu entnehmen: neue Kathrein Katalognummern nutzen, Format bleibt • Import von Tab-getrennten Textdateien
Darstellung Horizontal-diagramm	• kontinuierliche und automatische Anpassung des Diagramms an die Daten, ohne auf „Berechnen“ klicken zu müssen • Änderung des Diagramms erst, wenn alle notwendigen Daten eingegeben und keine leeren Felder mehr vorhanden sind
Ausgabe der Diagrammdaten	• Exportieren der Horizontal-Diagramme als Textdatei: Tab-getrennte Tabelle in 1°- oder 10°-Schritten -> gleiches Format wie die Vorgabedaten • PDF: für internen Gebrauch nur Diagramme mit Titel
Abspeichern und Öffnen von Projekten	• Speichern der Datei: „Speichern unter“, Namen und Pfad selbst aussuchen, keine Vorlage
GUI – Optik	• Benutzerfreundlich, Erweiterbar, Intuitiv • Button zum Erweitern und Löschen von beliebig viele Spalten • Textfeld zur Eingabe der Rechenfrequenz
GUI – Funktionalität	• keine Reiter, alles ohne Fensterwechsel bzw. extra Klick änderbar • bei Eingabe der Rechenfrequenz: automatische Auswahl der Primärdaten über den Dateinamen; nachträglich änderbar, aber bei Änderung von Antenne oder Rechenfrequenz sollen die Daten nicht neu eingegeben werden müssen • Leistung: im Programm in Amplitude umrechnen
Fertige Anwendung	• Als zip-Datei mit allen wichtigen Dateien zum Entpacken -> keine Anwendung zum Installieren

Tabelle 13: Anhang B Ideen für die Zukunft

Thema	**Beschreibung**
Pythagoras-Tool	• über Rechentool xy-Koordinaten (Elemententfernung und Seitenversatz) in Elemententfernung und Elementrichtung umrechnen (Pythagoras), per Button einfügen oder per Hand übertragen, Umrechnung z. B. übers Menü, Pop-up-Fenster oder aufklappbares Feld neben dem Eingabefeld
Anpassen des Wunsch-diagramms	• Option zum Übereinanderlegen mehrerer Soll-Diagramme, Bilden des kleinsten gemeinsamen Nenners, einzelne Diagramme ausblenden, farbige Kennzeichnung von Diagrammen unterschiedlicher Herkunft an den Einzügen (damit erkennbar ist, zu welchem Netz die Einzüge gehören) • Anpassen des Diagramms durch Verschieben mit dem Mauszeiger, automatische Anpassung der Parameter
Auswahl der Primärdaten	• Verzeichnis-Sortierung über Antennentyp (Band II, Bd III oder Bd IV/V), bei Auswahl der richtigen Antenne wird beim Darüberfahren mit der Maus das jeweilige Bild der Antenne aus dem Katalog angezeigt • Muss erweiterbar sein, einheitliche Dateibezeichnung festlegen, Ordner- bzw. Pfadvorgabe
Import von Daten	• LabVIEW-Exportdateien importierbar machen
Spalten	• beim Schließen/ Zuklappen von Spalten werden die Einträge der anderen Spalten gespeichert • Expertenmodus: Für jede Spalte ein eigenes Antennenelement mit unterschiedlichen Primärdaten auswählen
Ebenen	• nachträgliches Hinzufügen von mehreren Ebenen mit jeweils mehreren Spalten
Eingabefelder	• Eingabefelder mit Pfeilen, um die Werte erhöhen/ verringern und die Veränderung im Diagramm beobachten zu können • Wo es Sinn macht: automatisches Ausfüllen der Spalten, manuell änderbar; bei mehreren Ebenen: Standartkonfiguration einfügen
Machbarkeits-prüfung	• mechanische Machbarkeit der geometrischen Anordnung der Antennenelemente prüfen

Fortsetzung Tabelle 13: Anhang B Ideen für die Zukunft

Thema	**Beschreibung**
Vertikal-diagramm	• Vertikaldiagramm berechnen und darstellen, in PDF ausgeben
Werte berechnen	• Anzeige der Werte: GH (horizontaler Gewinn), GV (vertikaler Gewinn) • Integralgewinn daraus berechnen und anzeigen; Winkel an der maximalen Amplitude ermitteln und zur Berechnung nutzen; Winkel änderbar
PDF erzeugen	• für Antennenhersteller mit Konfigurationsdaten (z. B. mit Angabe der Leistung statt Amplitude)

Anhang C Variablentabelle

Tabelle 14: Anhang C Variablentabelle

Bezeichnung	**Quick-BASIC**	**LabVIEW**	**Java (neu)**
Amplitude	A	A; A[Zeile]	diagrammamplitude
Amplitude des Antennenelements (meistens 1)	A(X, Y, 0)	Axy[0]	elementamplitudeGesamt
Anfangswinkel horizontal	AH		*fällt weg*
Anfangswinkel vertikal	AV		*fällt weg*
Azimutwinkel	AW	AW	azimutwinkel
Betrag der Entfernung zwischen Antennenmast-mittelpunt und Antennenelement-mittelpunkt	BB	BB	*fällt weg (= elemententfernung)*
Cosinuswinkel		CW	*fällt weg*
Dipolabstand in Metern	D(7)	D[7]	*fällt weg*
Dipolebenen pro Element	D(5)	D[5]	*fällt weg*
Elementart (Antennenfeld, Yagi, Dipol, Super T Stile, Verwendung der Fourier-Koeffizienten			*fällt weg*
Elemententfernung	A(X, Y, 4)	Axy[4]	elemententfernung
Elementrichtung	A(X, Y, 5)	Axy[5]	elementrichtung
Endwinkel horizontal	EH		*fällt weg*
Endwinkel vertikal	EV		*fällt weg*

Fortsetzung Tabelle 14: Anhang C Variablentabelle

Bezeichnung	**Quick-BASIC**	**LabVIEW**	**Java (neu)**
Entfernung imaginär	JJ	JJ	*entfernungImaginaer*
Entfernung imaginär zwischen Antennenmast-mittelpunt und Antennenelement-mittelpunkt	ENTFJ		*fällt weg (=JJ)*
Entfernung Quellpunkt bis Bezugspunt der Antenne (Strecke)	D(9)	D[9]	*fällt weg*
Entfernung real	RR	RR	entfernungReal
Entfernung real zwischen Antennenmast-mittelpunt und Antennenelement-mittelpunkt	ENTFR		*fällt weg (=RR)*
entspricht der Differenz zwischen dem Azimutwinkel und der Strahlrichtung in Bogenmaß	B (α)	B	differenzwinkel
entspricht der Differenz zwischen Erhebungswinkel und dem Neigungswinkel in Bogenmaß	C (β)	C	differenzwinkel
entspricht der Wellenzahl kz	UF	UF	kRechen
Erhebungswinkel	EW	EW	erhebungswinkel
Erste zu berechnende Spalte	XA bzw. YA		*fällt weg*
Gewinn je Element in dB	D(3)	D[3]	*fällt weg*
Hilfsvariable zur Berechnung von VK	D	D; D[Zeile]	*fällt weg*

Fortsetzung Tabelle 14: Anhang C Variablentabelle

Bezeichnung	**Quick-BASIC**	**LabVIEW**	**Java (neu)**
Horizontale Phase		PH	horizontalePhase
Horizontaler Faktor für die Amplitude	AX	AX	*fällt weg*
Imaginärteil (Aufsummierter Imaginärteil von allen Antennenelementen von einem Winkel)	Q		imaginaerteilGesamt
Interpolationsvariable oberer Wert		UP	winkelAufgerundet
Interpolationsvariable unterer Wert		LO	winkelAbgerundet
ist = π wenn Theta größer 90°ist	P0		*fällt weg*
Konstante Phase	A(X, Y, 7)	Axy[7]	konstantePhase
Korrektur Horizontal	D(2)	D[2]	*fällt weg*
Letze zu berechnende Spalte	YE bzw. XE		*fällt weg*
Neigungswinkel des Antennenelementes der Spalte X und der Zeile Y	A(X, Y, 3)	Axy[3]	*neigungswinkel*
Nennfrequenz	D(6)	D[6]	*nennfrequenz*
nicht belegt	D(0)	D[0]	*fällt weg*
Phase	P	AP	*diagrammphase*
Primärdatenmatrix horizontal 2D		H; H[Zeile][Spalte]	*PrimaerdatenHorizontal[][]*
Primärdatenmatrix vertikal 1D		V; V[Zeile]	*PrimaerdatenVertikal[][]*
Realteil (Aufsummierter Realteil von allen Antennenelementen von einem Winkel)	R		*realteilGesamt*
Rechenfrequenz	FM	FM	*rechenfrequenz*
Rückwärtskorrektur	D(4)	D[4]	*fällt weg*

Fortsetzung Tabelle 14: Anhang C Variablentabelle

Bezeichnung	**Quick-BASIC**	**LabVIEW**	**Java (neu)**
Schrittweite d. horizontalen Primärdaten	N		*fällt weg*
Schrittweite horizontal	SH		*fällt weg*
Schrittweite vertikal	SV		*fällt weg*
Sinuswinkel		SW	*fällt weg*
Spaltenanzahl der Antennenelemente	Y		spalten
Speisephase	A(X, Y, 1)	Axy[1]	speisephase
Strahlrichtung des Antennenelementes der Spalte X und der Zeile Y	A(X, Y, 2)	Axy[2]	strahlrichtung
Vertikaler Abstand	A(X, Y, 6)	Axy[6]	vertikalabstand
Vertikaler Faktor für die Amplitude	VK	VK	fällt weg
Vertikaler Nullwinkel	D(1)	D[1]	fällt weg
Winkel zwischen Antennenmast-mittelpunt und Antennenelement-mittelpunkt	WW	WW	fällt weg (= elementrichtung)
Zeilenanzahl der Antennenelemente	X		zeilen
π/180	C1	C1	fällt weg

Anhang D Diagramme

Um die Programmstruktur zu veranschaulichen sind im Anhang folgende Diagramme zu finden:

- Ablaufdiagramm
- Klassendiagramm
- Struktogramm Rechenalgorithmus detailliert
- Struktogramm Rechenalgorithmus vereinfacht

Anhang E Quellcode

Der Quellcode befindet sich in Form von PDF-Dateien in nach der Programmstruktur untergliederten Verzeichnissen auf der beigelegten CD-ROM.

Anhang F Ausführbare Programmdatei der Antennensoftware

Die Antennensoftware befindet sich zusammen mit der aktuellen Java Runtime Environment (Version 8 Update 45, Releasedatum 14. April 2015) auf der beigelegten CD-ROM. Die Java-Runtime muss installiert sein, damit die Antennensoftware funktioniert.
Auf der CD-ROM befinden sich außerdem ein Ordner mit Vorgabe- und Primärdaten sowie eine Projektdatei, die in der Anwendung geöffnet werden kann.

Anhang G Schritt-für-Schritt-Anleitung zur Nutzung der Software

(System-)voraussetzungen

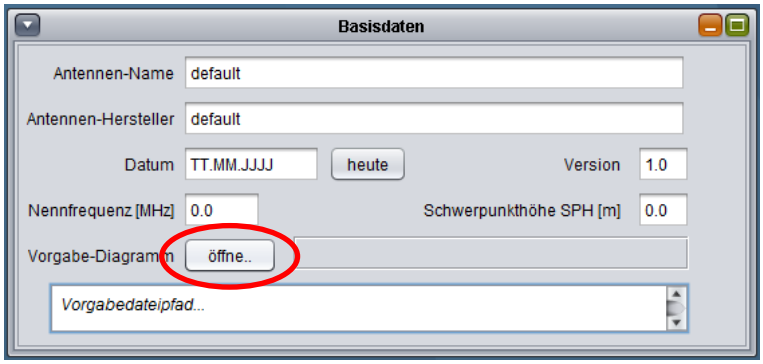
- Java Runtime muss installiert sein
- Ausführbare Java-Datei und der dazugehörige Bibliotheksordner „lib“ müssen in einem Ordner sein
- Vorgabe- und Primärdaten müssen im richtigen Format vorhanden sein

Programm starten

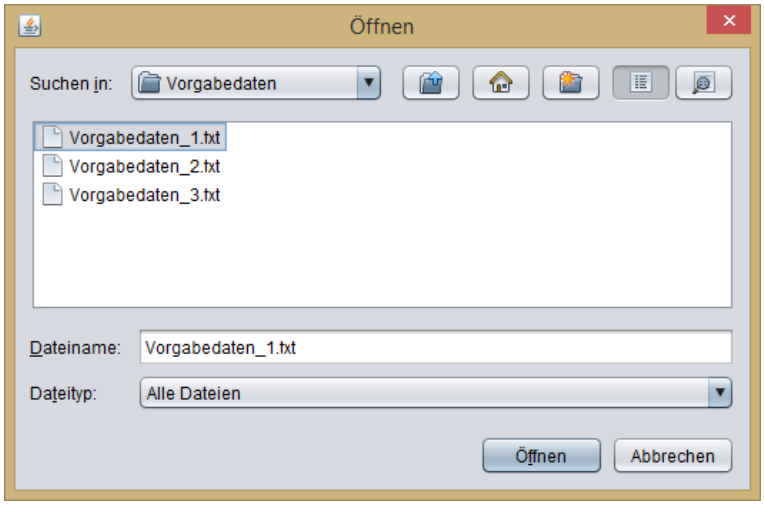
- Öffnen der Java-Datei mit Doppelklick: Die GUI der Antennensoftware öffnet sich

Vorgabedaten einlesen

- Klick auf den Button „öffne..“

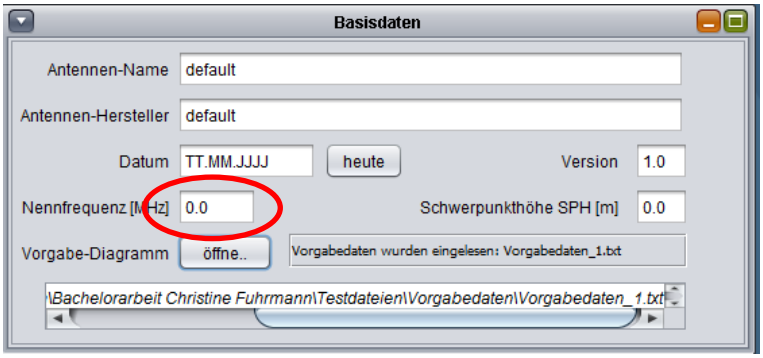


- Auswahl der Textdatei, „Öffnen“



- Im Textfeld wird der Verzeichnispfad der ausgewählten Datei angezeigt

Nennfrequenz eingeben

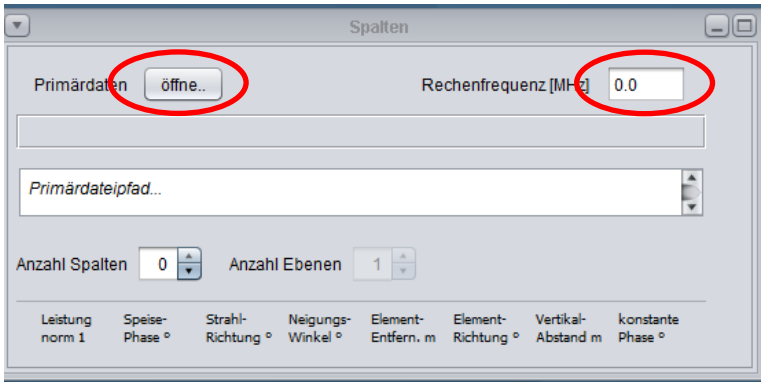


Rechenfrequenz eingeben

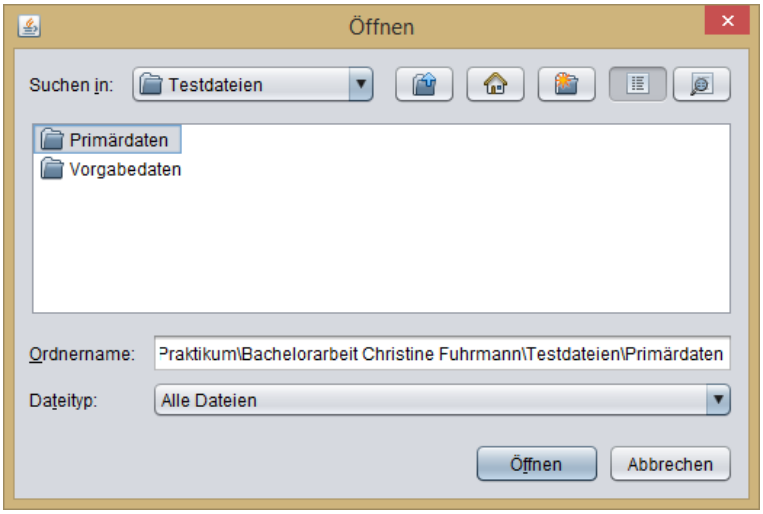
- Eingabe der Rechenfrequenz im Textfeld, damit im nächsten Schritt anhand der Rechenfrequenz die richtigen Primärdateien ausgewählt werden (die beiden Schritte können auch vertauscht werden, der Default-Wert aller Zahlen-Eingabefelder ist 0)

Primärdaten einlesen

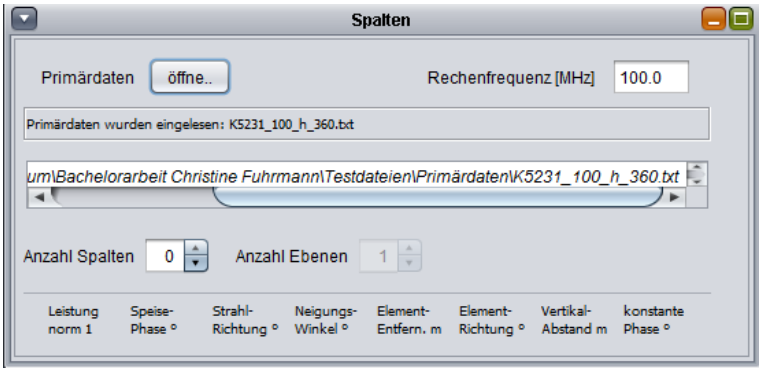
- Klick auf den Button „öffne..“



- Auswahl des Verzeichnisses, in dem die Primärdateien liegen, OK

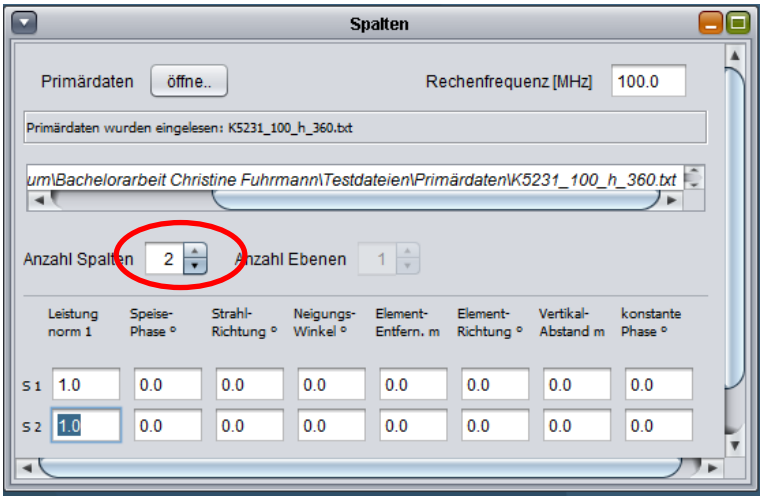


- Im grau hinterlegten Textfeld wird die anhand der Rechenfrequenz gewählte Primärdatei des Horizontaldiagramms angezeigt (die Daten des Vertikaldiagramms werden analog ausgewählt und eingelesen)
- Im weiß hinterlegten Textfeld wird der komplette Dateipfad angezeigt

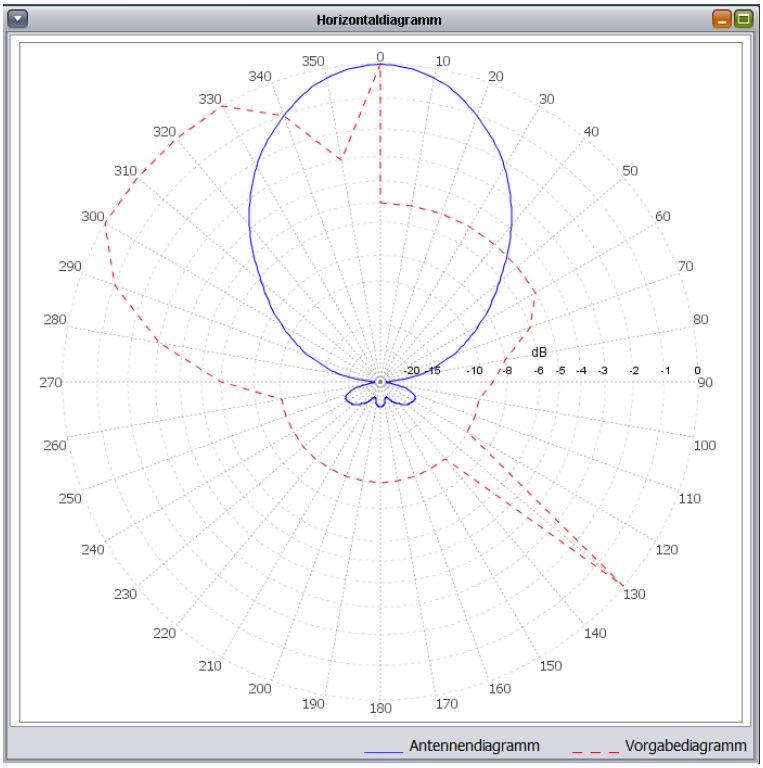


Erstellen von Spalten

- Klick auf den Zeiger-Button zur Generierung beliebig vieler Spalten, das Fenster kann mit der Maus nach unten hin vergrößert werden



- Im Diagrammfenster wird automatisch das Horizontaldiagramm dargestellt, das aus den aktuellen Daten berechnet wird



Eingeben der antennenspezifischen Daten

- Antennenname, Datum, usw. im Fenster „Basisdaten“
- Eingabe der Spaltendaten im Fenster „Spalten“

Speichern oder Exportieren der Projektdaten

- Über „Datei“ in der oberen Leiste kann das Projekt gespeichert oder das Diagramm inklusive aller Daten in unterschiedlichen Formaten exportiert werden
- Projekte können später über „Datei – Öffnen“ eingelesen werden
- Wichtig bei der Auswahl des Zielverzeichnisses und der Eingabe eines Dateinamens: Projekte und Export-Textdateien immer mit der Endung „.txt“ angeben (!)

Inhaltsverzeichnis CD-ROM

Tabelle 15: Inhaltsverzeichnis beigelegte CD-ROM

Verzeichnis	Dateien
Bachelorarbeit	BA_Christine_Fuhrmann.docx: Diese Arbeit als Word-Dokument BA_Christine_Fuhrmann.pdf: Diese Arbeit als PDF-Dokument
Diagramme	Diagramme aus Anhang D in Großformat als PDF-Dateien
Java Runtime Environment	jre-8u45-windows-i586.exe: Ausführbare Java-Installationsdatei für Windows (32-Bit) jre-8u45-windows-x64.exe: Ausführbare Java-Installationsdatei für Windows (64-Bit)
Literaturverzeichnis > Internetseiten	Alle in der Arbeit angegeben Webseiten als HTML-Dateien
Literaturverzeichnis > Sonstige	Bachelorarbeit von Sabine Sengelmann als PDF-Datei
Programmdateien > Antennen-software	Dateien zum Öffnen der Anwendung als Projekt in der Entwicklungsumgebung NetBeans
Programmdateien > Ausführbare Datei	Antennensoftware.jar: Datei zum Ausführen der Antennensoftware Ordner lib: enthält die importierten Bibliotheken und muss immer (!) im gleichen Verzeichnis wie die jar-Datei sein
Quelltext	Gesamter Quelltext als PDF-Dateien; Die Ordnerstruktur entspricht der Paket-Struktur im Sourcecode, jede PDF-Datei entspricht einer Klasse. Die Dateien sind mit den Klassennamen benannt
Testdateien > Vorgabedaten	Vorgabedaten_1.txt, Vorgabedaten _2.txt: Vorgabedaten zum Einlesen in die Antennensoftware
Testdateien > Primärdaten	K5231_100_h_360.txt usw.: Primärdaten eines Antennenelements mit unterschiedlichen Rechenfrequenzen zum Einlesen in die Software

Versicherung

Ich versichere, dass ich die Bachelorarbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegeben Quellen angefertigt habe, und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat. Alle Ausführungen der Arbeit, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Hamburg, 31.07.2015

Ort, Datum

Unterschrift Christine Fuhrmann