GNU Radio Implementierungen zur Synchronisierung von OFDM-Systemen

von
Dominik Auras

Studienarbeit

im Studiengang Technische Informatik

vorgelegt der
Fakultät für Elektrotechnik und Informationstechnik
der Rheinisch-Westfälischen Technischen Hochschule Aachen
im: Juli 2009

Angefertigt am
Lehrstuhl für Theoretische Informationstechnik
bei
Prof. Dr. R. Mathar

Ich versichere, dass ich diese Studienarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtliche oder sinngemäße Wiedergaben aus diesen Quellen sind als solche kenntlich gemacht und durch Zitate belegt.

Aachen, den

Inhaltsverzeichnis

1 Einleitung

1.1 Allgemein

Einer der vielversprechendsten Kandidaten für die vierte Generation mobiler Kommunikationssysteme ist die OFDM[1]-Übertragungstechnik. Sie bietet, gegenüber anderen Breitbandtechniken, einfache und leicht zu implementierende Strukturen. Zudem wird sie dem Ziel, einen Informationszugang mit Hochgeschwindigkeit jedermann verfügbar zu machen, vollends gerecht.

Als Mehrträgersystem teilt es die verfügbare Bandbreite in eine Menge von schmalen Unterträgern auf, die getrennt moduliert werden. Dabei nutzt man die Orthogonalität von Sinuswellen mit gleicher Grundfrequenz aus um das Spektrum so dicht wie möglich zu besetzen. Die Unterträger beeinflussen sich also nicht gegenseitig. Zudem lässt sich, wie wir später noch zeigen werden, die Modulation aller Träger sehr einfach über die schnelle Fouriertransformation durchführen.

Seine besondere Stärke spielt diese Technologie in frequenz-selektiven Kanälen aus. In diesen ist die Dämpfung und die Verzögerung von der Frequenz abhängig. Diese Effekte werden durch Mehrwegeausbreitung verursacht. Während Einzelträgersysteme mit breiten Trägern aufwendige Echokompensationen durchführen, beschränkt sich die Schätzung und Korrektur von Kanaleinflüssen bei OFDM auf einen multiplikativen Faktor je Träger. Dabei gehen wir davon aus, dass die Unterträger jeweils sehr viel schmaler als die gesamte Bandbreite sind. Dann kann man davon ausgehen, dass ein Träger nur zusammenhängend gedämpft und verzögert wird.

Dennoch gibt es auch einen Nachteil. Die Systeme sind sehr empfindlich gegenüber Zeit- und Frequenzverschiebungen. Hier muss man teils komplexe Synchronisationsalgorithmen verwenden. Dabei gibt es zwei Ansätze: die sogenannten blinden Verfahren und die datenbasierten Verfahren. Die Ersten versuchen, den OFDM-Empfänger ohne spezifische Kenntnis vom gesendeten Signal zu synchronisieren. Z.b. kann man hier die zyklische Erweiterung (mehr dazu später) ausnutzen. In dieser Arbeit werden nur datenbasierte Verfahren vorgestellt. Diese fügen gezielt bekannte Informationen in das gesendete Signal ein. Der Empfänger sucht diese und synchronisiert sich mit Hilfe dieser Daten.

Die Studienarbeit ist wie folgt aufgebaut. Zuerst wird eine allgemeine Einführung in OFDM-Systeme gegeben und im speziellen das benutzte mathematische Modell beschrieben. Dabei werden auch die verschiedenen auftretenden Probleme bei der OFDM-Übertragung erfasst. Im zweiten Kapitel folgt eine kurze Einführung in die Aufgaben der Synchronisierung. Das dritte Kapitel beschreibt GNU Radio. Es stellt die Prinzipien vor und führt

[1] Orthogonal Frequency Division Multiplexing

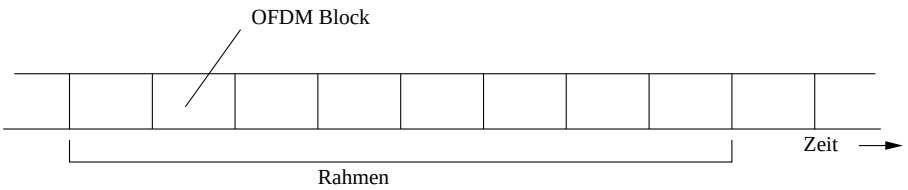


Abbildung 1.1: Aufbau eines OFDM-Rahmen

teilweise in die Details der Software ein. Abgerundet wird das Kapitel mit einem Beispiel. Im vierten Kapitel befassen wir uns mit der Zeitsynchronisierung. Zwei Algorithmen werden vorgestellt. Sie werden mathematisch definiert und in eigenen Worten beschrieben. Mögliche Implementierungsformen mit GNU Radio werden aufgeführt. Mit den durchgeführten Simulationen wird versucht, die Eigenschaften der Algorithmen zu erfassen und sie miteinander zu vergleichen. Das fünfte Kapitel beinhaltet die Vorstellung zweier Frequenzsynchronisierungsalgorithmen. Auch sie werden detailliert beschrieben und mögliche GNU Radio-Implementierungen präsentiert.

1.2 Systembeschreibung

Zuerst kommt eine kurze Beschreibung des allgemeinen Aufbaus einer OFDM-Übertragung. Dabei wird von einer datenbasierten Synchronisierung ausgegangen, die die Nutzdaten in Rahmen (Abb. 1.1) einteilt und Informationen am Anfang, oder ggf. auch an anderen Positionen, hinzufügt. Es werden immer komplette Rahmen übertragen, die aus einer fixen Anzahl von Blöcken bestehen. Diese wiederum bestehen aus einer festen Anzahl an Abtastwerten.

Sei N die Zahl der Unterträger und $\mathcal{I}$ sei die Menge der benutzten Unterträger. Die Mächtigkeit der Menge $\mathcal{I}$ ist $N_s = |\mathcal{I}|$. Zu jedem OFDM-Block gibt es einen Vektor der Länge N mit Informationssymbolen. Die Symbole liegen als komplexe Zahlen vor und repräsentieren Punkte im Signalraum, z.B. aus den PSK- oder QAM-Konstellationen. Unbenutzte, virtuelle Unterträger setzen wir zu Null. Die Elemente des Vektors des i-ten Blocks seien definiert als

$$\left.\begin{array}{ll} S_{i,k} \in \mathbb{C}, & \text{falls } k \in \mathcal{I} \\ S_{i,k} = 0, & \text{sonst} \end{array}\right\} \forall\, k = 0 \ldots N-1. \tag{1.1}$$

Dann seien

$$s_i(k) = \begin{cases} \sum_{n=0}^{N-1} S_{i,n} \exp(j2\pi kn/N), & \text{falls } -N_g \leq k \leq N-1 \\ 0, & \text{sonst} \end{cases} \tag{1.2}$$

die Abtastwerte des i-ten OFDM-Blocks, wobei N_g die Länge des zyklischen Präfix angibt. Aus dem Vektor wird eine Folge von Abtastwerten. Dabei entspricht der Blockabschnitt für

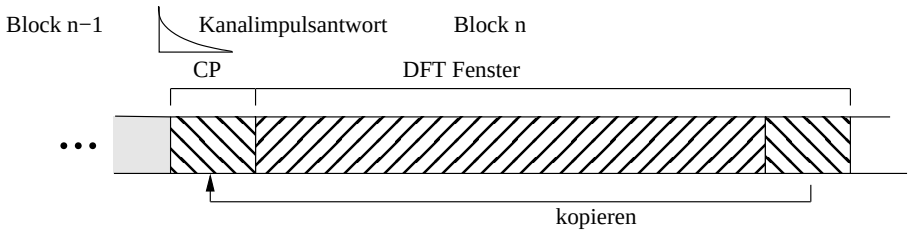


Abbildung 1.2: Struktur eines OFDM-Blocks

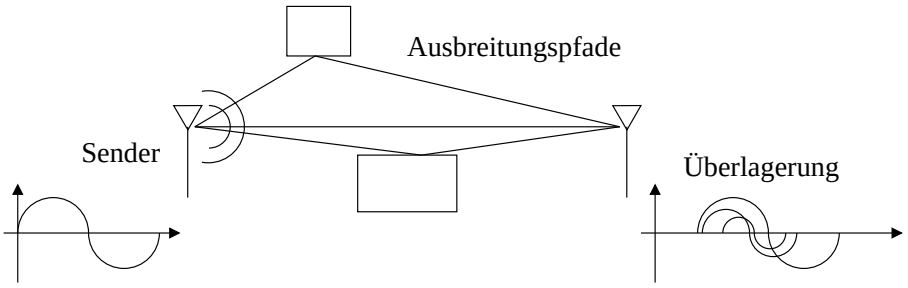


Abbildung 1.3: Mehrwegeausbreitung des Sendesignals

$0 \leq k \leq N-1$ der inversen diskreten Fouriertransformation, die sich besonders vorteilhaft mit dem FFT-Algorithmus berechnen lässt.

Die ersten N_g Abtastwerte sind das zyklische Präfix (Abb. 1.2). Sie sind identisch zu den letzten N_g Werten aus dem DFT-Fenster. Durch diese Erweiterung verringert man den Einfluss der Mehrwegeausbreitung. Werden nämlich mehrere Blöcke in Kanälen mit Gedächtnis unmittelbar hintereinander gesendet, so breitet sich das Signal auf verschiedenen Pfaden aus. Auf einigen dieser Pfade legt es möglicherweise einen längeren Weg zurück (s. Abb. 1.3). Im Empfänger überlagern sich dann mehrere Kopien desselben Signals, aber unterschiedlich gedämpft und verzögert. Die Länge des Präfix ist ein Designparameter des Systems und bestimmt die maximal akzeptierbare Kanalimpulsantwortlänge. Das Fenster für die Rücktransformation legt man im Empfänger idealerweise nur über den Teil, der nicht durch einen vorherigen Block gestört ist.

Das übertragene Basisband-OFDM-Signal im Zeitbereich setzt sich aus den verschiedenen Blöcken, die später ihrerseits in Rahmen gruppiert werden, zusammen:

$$s(k) = \sum_i s_i(k - i(N + N_g)). \tag{1.3}$$

Beim empfangenen Signal berücksichtigen wir noch den Kanaleinfluss. Sei $h(n)$ die abgetastete Kanalimpulsantwort bei Mehrwegeausbreitung auf dem Übertragungsmedium. Ferner bezeichne L die Länge der Impulsantwort. Gleichung 1.4 entspricht der Faltung des Sendesignals mit der Impulsantwort.

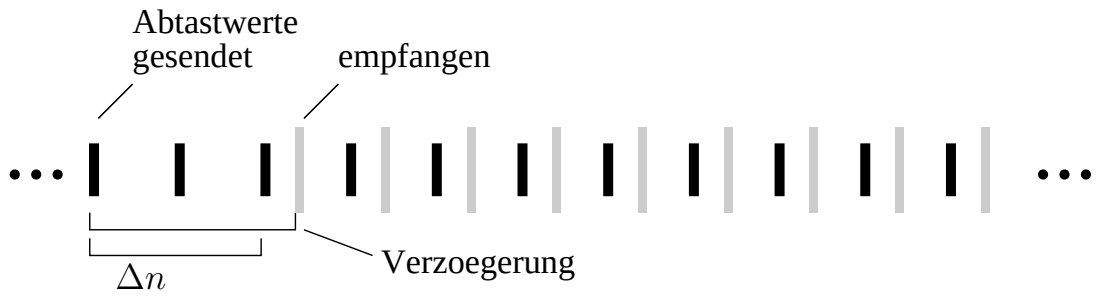


Abbildung 1.4: Zeitverzögerung des empfangenen Signals

$$y(k) = \sum_{n=0}^{L-1} h(n)s(k-n) \tag{1.4}$$

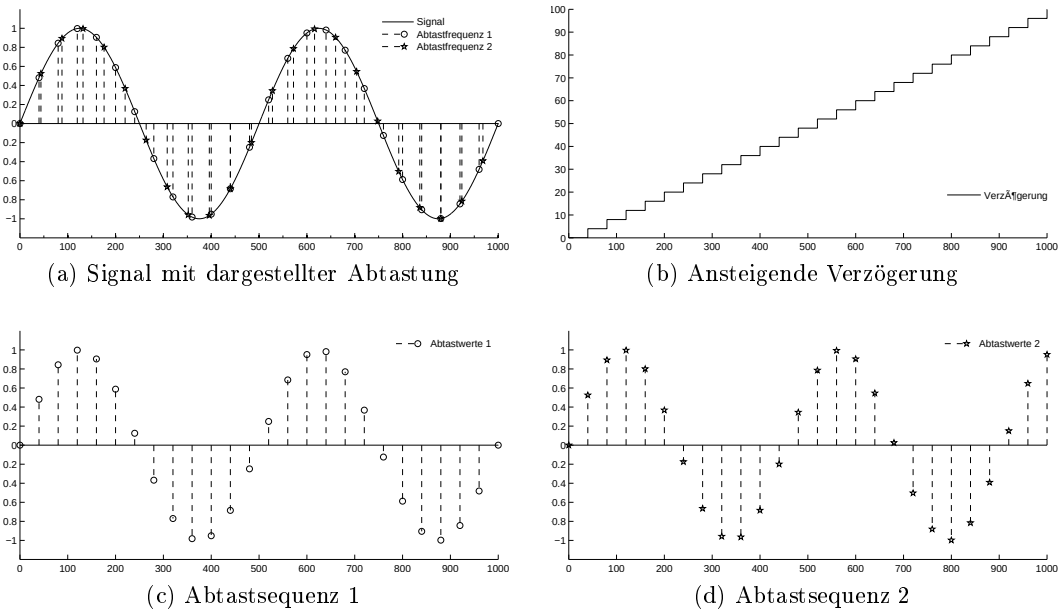
Zusätzlich repräsentiere $w(n)$ einen Rauschprozess mit $\{w(n)|n \in \mathbb{N}\} \sim SCN(0, \sigma_w^2)$. Außerdem sei die Ankunftszeit des Signals beim Empfänger unbekannt. Die Unbestimmtheit der Ankunft modellieren wir durch die Zeitverschiebung $\Delta n \in \mathbb{N}^+$ (Abb. 1.4). Dabei berücksichtigen wir nur Verschiebungen als Vielfaches des Abtastabstands und nehmen an, dass der restliche Anteil als Phasenverschiebung von der Kanalschätzung korrigiert wird. Diese Effekte fließen in die Gleichung 1.5 ein.

Im Sender wird das digitale Signal über Digital-Analog-Wandler in ein analoges Signal umgesetzt. Umgekehrt setzen Analog-Digital-Wandler das empfangene analoge Signal in ein digitales Signal um. Auf beiden Seiten werden die Wandler mit einer festen Taktfrequenz betrieben, weiter Abtastfrequenz genannt.

Nun ist es bei realen Taktquellen nicht zu vermeiden, dass zwei unterschiedliche Quellen nicht exakt dieselbe Frequenz generieren. Eine Divergenz zwischen den Abtastfrequenzen im Sender und Empfänger führt zu weiteren Verzerrungen des empfangenen Signals. In Abbildung 1.5 ist dieser Sachverhalt genauer dargestellt. Das abgetastete empfangene Signal wird zeitlich gesehen scheinbar expandiert (Abb. 1.5d) bzw. komprimiert. Es ergibt sich eine steigende Verzögerung (Abb. 1.5b) der empfangenen Abtastwerte gegenüber den ausgesendeten (Abb. 1.5c). Ein Block Signalwerte der Länge M im Sender resultiert in einem Block mit $M + 1$ Werten im Empfänger, wobei M vom Versatz der Taktquellen abhängt. Analog gilt auch, dass der empfangene Block gegebenenfalls um einen Wert kürzer ist, je nachdem, welche Taktquelle eine höhere Frequenz erzeugt.

Die Verzerrung durch einen Versatz der Abtastfrequenzen findet sich nicht im vorgestellten Modell wieder. Im Allgemeinen reicht eine Betrachtung unter der Annahme, dass die Takte synchronisiert sind. Die Simulation, die in einem späteren Kapitel vorgestellt wird, berücksichtigt aber den Versatz.

Desweiteren existiere eine Frequenzverschiebung ϕ des Signals (siehe Abbildung 1.6), normiert auf den Unterträgerabstand, die sich z.B. durch Fertigungsungenauigkeiten oder den



(a) Signal mit dargestellter Abtastung

(b) Ansteigende Verzögerung

(c) Abtastsequenz 1

(d) Abtastsequenz 2

Abbildung 1.5: Nicht ideale Abtastung

Temperaturdrift der lokalen Oszillatoren für die Zwischenfrequenzen erklären lässt. Der Versatz der Frequenzen führt dazu, dass die Mittenfrequenz des OFDM-Bands, und somit auch die Mittenfrequenzen sämtlicher Unterträger, zwischen Sender und Empfänger gegeneinander verschoben sind.

Insgesamt stellt sich das empfangene, abgetastete Signal dann dar als

$$r(k) = y(k - \Delta n) \exp(j2\pi\phi k/N) + w(k). \tag{1.5}$$

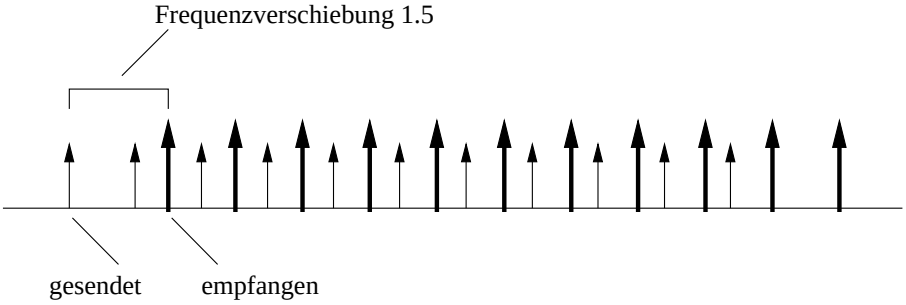


Abbildung 1.6: Normierte Frequenzverschiebung empfangenen Signals

2 Aufgaben der Synchronisierung

Die Synchronisierung eines OFDM-Systems kann allgemein in zwei separate Bereiche eingeteilt werden: die Zeit- und die Frequenzsynchronisierung. Dabei trifft man häufig auf denselben Aufbau.

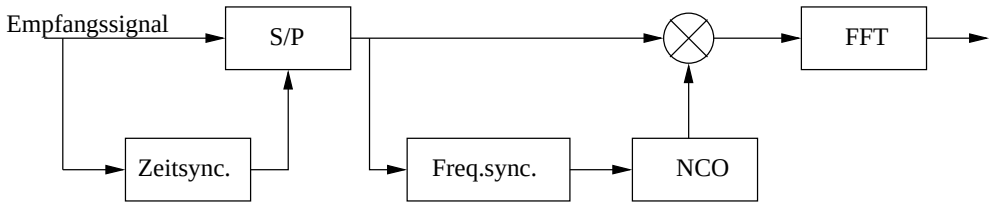


Abbildung 2.1: Typische Synchronisierungsstufe

Zuerst detektiert der Zeitsynchronisierungsalgorithmus das Signal. Die geschätzte Position teilt er dem Seriell-zu-Parallel-Konverter mit (S/P im Bild), welcher einen Vektor entsprechend der Rahmenlänge aus dem Datenstrom entnimmt. Diesen Vektor benutzt nun der Algorithmus für die Frequenzsynchronisierung um ein Fehlersignal, welches proportional zur auftretenden Frequenzverschiebung ist, zu erzeugen. Mit diesem Signal, das die Abweichung charakterisiert, wird ein digitaler Oszillator angesteuert, der hieraus ein Signal mit einer Frequenz proportional zur Abweichung generiert. Mischen wir nun den Vektor mit dem Ausgang des Oszillators, korrigieren wir so die Frequenzverschiebung. Die anschließende FFT transformiert den Vektor in den Frequenzbereich.

Selbstverständlich gibt es auch andere Optionen zum Aufbau der Empfängerstrukturen. Diese sind jedoch meist äquivalent und unterscheiden sich lediglich in der Implementierungsform des gewählten Algorithmus.

2.1 Zeitsynchronisierung

Das Ziel dieser Synchronisierungsstufe ist, beim Empfänger im Zeitbereich ein Fenster der Länge N passend über den Blöcken des empfangenen Signals zu positionieren. Der entstehende Vektor wird dann weiter verarbeitet. Bei den vorgestellten Algorithmen gruppieren wir, wie bereits früher angedeutet, die OFDM-Blöcke in Rahmen. Jeder Rahmen bekommt ein oder mehrere Präambelblöcke vorangesetzt. Die Algorithmen positionieren das Empfangsfenster möglichst genau über der Präambel. Die Position der Fenster für die Datenblöcke leiten wir relativ von der Position des Präambelblocks ab.

Der ideale Zeitpunkt soll nun genauer definiert werden, da er im nächsten Kapitel öfters erwähnt wird. Dieser Zeitpunkt wird von den Algorithmen geschätzt. Prinzipiell hängt seine genaue Position von der Implementierung ab. Z.b. kann er sowohl das Ende auch als den Anfang des Fensters, das über die Präambel gelegt wird, kennzeichnen. Im Folgenden wird, falls nicht anders erwähnt, der ideale Zeitpunkt als Beginn des Fensters betrachtet. Der Zeitpunkt ist optimal falls er das Verhältnis der Signalleistung zur Rauschleistung maximiert. Das Fenster sollte so über die Präambel gelegt werden, dass möglichst keine Störungen durch vorhergehende Blöcke auftreten. Am besten geeignet ist hierfür der erste Punkt, der nicht mehr zum zyklischen Präfix gehört. [MSM99] entnehmen wir ausserdem, dass alle Zeitpunkte, die in den Bereich des ungestörten zyklischen Präfix fallen, als ideal zu betrachten sind.

2.2 Frequenzsynchronisierung

Bei einem realen System treffen wir immer auch auf andere Störungen ausser dem allgegenwärtigen Rauschen. Man kann davon ausgehen, dass fast immer eine Frequenzverschiebung zwischen Sende- und Empfangssignal besteht, zumindest solange die Hardware nicht aufeinander abgestimmt wurde. Dieser Versatz ist z.B. durch die Ungenauigkeiten und möglichen Instabilitäten der lokalen Oszillatoren begründet.

Nehmen wir an, die Verschiebung sei durch ϕ beschrieben, und sie sei auf den Unterträgerabstand B/N normiert, wobei B die gesamte Bandbreite ist. Wir können ϕ in zwei Teile aufsplitten:

$$\phi = \theta + \eta \tag{2.1}$$

den gebrochenen Anteil η mit $-0.5 \leq \eta < 0.5, \eta \in \mathbb{R}$ und den ganzzahligen Anteil $\theta \in \mathbb{Z}$.

Die Verschiebung η zerstört die Orthogonalität zwischen den Eigenfrequenzen der FFT und den Trägern des Empfangssignals. Vor der Dekomposition des Zeitsignals in seine Trägeranteile stellen wir demnach die Koinzidenz der Sender- mit den Empfangsträgern durch Korrektur von η wieder her. Dies geschieht durch Multiplikation der Abtastwerte mit einer komplexen Exponentialfunktion. Die auftretende Disambiguität bezüglich der Träger wird nun durch θ ausgedrückt. Sie wird nach der FFT durch zyklische Rotation des transformierten Vektor berichtigt.

3 GNU Radio

Weiterer Bestandteil der Studienarbeit ist es, die vorgestellten Algorithmen als Anwendung mit Hilfe des GNU Radio Projekts zu erstellen. In diesem Kapitel wird das grundlegende Konzept von GNU Radio vorgestellt. In den Kapiteln über die Algorithmen stellen wir Ideen zur Realisierung dieser mit GNU Radio vor.

3.1 Was ist es?

GNU Radio ist ein Open Source-Projekt unter der GPL[1]. Es stellt dem Entwickler von Software-Defined-Radio-Anwendungen (kurz SDR) eine flexible Architektur zur schnellen Erstellung und Evaluierung von Konzepten zur Verfügung. Da die Programmentwicklung in Python, einer einfach zu erlernenden Skriptsprache, geschieht, kann man unkompliziert neue Ideen ausprobieren. Dennoch hat GNU Radio eine ausreichende Performanz um die meisten Anwendungen in Echtzeit auszuführen. Hierzu wird die gesamte Signalverarbeitung in C++ geschrieben.

Das GNU Radio-Projekt besteht aus zwei Komponenten. Das ist erstens das Framework und zweitens eine Sammlung von Implementierungen von Signalverarbeitungsblöcken, weiterhin auch kurz DSP-Blöcke (Digital Signal Processing) genannt.

3.2 Vorstellung des Frameworks

Mit dem Framework modelliert man Datenflüße zwischen Einheiten. Es beinhaltet alle nötigen Werkzeuge, um diese Modelle zu schreiben und auszuführen. Die Grundstruktur des Modells ist ein Graph. Er ist azyklisch und gerichtet, darf also keine Schleifen aufweisen und Verbindungen zwischen Knoten besitzen eine Richtung (Abb. 3.1). Jeder

[1] GNU General Public License

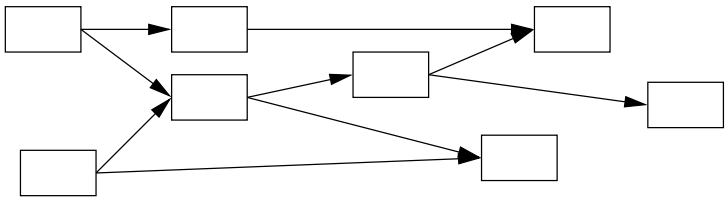


Abbildung 3.1: Beliebig komplizierter Signalflußgraph

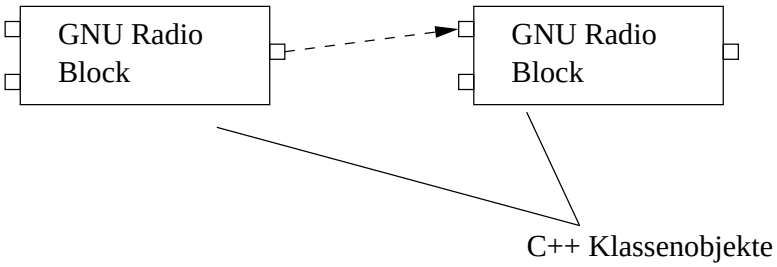


Abbildung 3.2: GNU Radio Blöcke mit Signalverbindung

Knoten ist eine Einheit die Daten transformiert. In GNU Radio nennt man dies einen "Block" (Abb. 3.2). Jeder Block hat Ein- und Ausgänge, deren Beschreibung in seinen beiden IO-Signaturen enthalten ist. Die Ein- und Ausgänge werden in GNU Radio Ports genannt. Jeder Port besitzt eine spezifische Breite, die in Anzahl von Bytes angegeben wird. Die Datentransformation im Knoten ist beliebiger Natur. Sie kann linear oder nichtlinear, gedächtnisbehaftet- oder -los usw. sein. Quellen und Senken sind in diesem Modell elementare Blöcke, die entweder keinen Ein- beziehungsweise keinen Ausgang besitzen.

Die Blöcke sind in zwei Unterkategorien einzuteilen. Es gibt hierarchische Blöcke. Sie sind aus beliebig vielen Blöcken zusammengesetzt. Und es gibt Terminalblöcke, die ihre Transformation algorithmisch definieren. Diese werden auch DSP-Blöcke genannten, da sie die Algorithmen zur digitalen Signalverarbeitung unmittelbar implementieren. Terminalblöcke werden mit C++ beschrieben, während hierarchische Blöcke im Moment nur in Python sinnvoll definiert werden können. Mit diesen beiden Kategorien erlaubt es das Framework, Graphen als Bäume zu beschreiben (Beispiel in Abb. 3.2). Der eigentliche Graph wird erzeugt, indem dessen Baum quasi platt gedrückt wird. Dabei sind die hierarchischen Blöcke die Knotenpunkte des Baums und die Terminalblöcke seine Blätter.

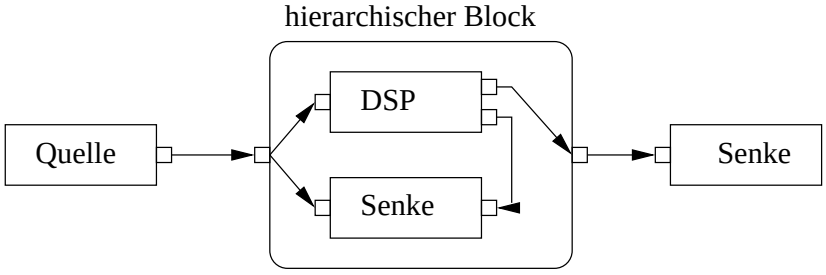


Abbildung 3.3: Beispiel hierarchisch aufgebauter Graph

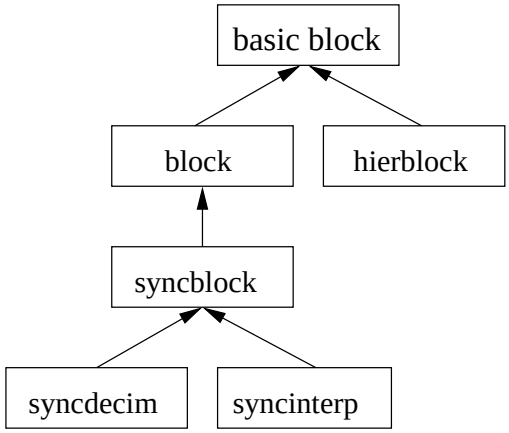


Abbildung 3.4: Hierarchie der GNU Radio Blöcke

3.2.1 Hierarchische Blöcke

Die hierarchischen Blöcke sind eine Komposition aus anderen Blöcken, gegebenenfalls auch wiederum aus hierarchischen Blöcken. Intern werden Verbindungen zwischen den Blöcken untereinander und zu den Ports des hierarchischen Blocks definiert. Extern gibt es für Anwendungen keinen Unterschied zu einem Terminalblock. Führt man ein Modell aus, ersetzt das Framework hierarchische Blöcke durch die Instantierungen der internen Blöcke und löst alle Verbindungen nach aussen passend auf. Es ensteht somit kein Overhead durch diese Blöcke.

3.2.2 Terminalblöcke

Die zweite Kategorie stellen die Terminalblöcke. Die Datentransformation in ihnen wird als C++-Algorithmus definiert. Dafür gibt es ein abstraktes Interface das jeder Block haben muss. Es ist in der Klasse *gr_block*, die Basis jedes Terminalblocks sein muss, spezifiziert. In der Klassenhierarchie der Blöcke in Abb. 3.2.2 befindet er sich neben der Basis der hierarchischen Blöcke. Ein einfacher Block muss IO-Signaturen angeben und zwei virtuelle Funktionen implementieren. Dies sind *general_work()* und *forecast()*. In der ersten von beiden wird der Algorithmus definiert. Das Framework ruft die Funktion mit Informationen zum Speicher für die Ports, wie z.B. Adresse und Menge des verfügbaren Speichers auf. Zu den Speichern folgt später eine kurze Erklärung. Der Algorithmus kann Elemente konsumieren, indem er *consume()* benutzt, und produzieren, indem er Werte in die Ausgangsspeicher schreibt und deren Anzahl zurückgibt.

Über *forecast()* erfragt das Framework Schätzungen für die Anzahl benötigter Eingabeelemente bei vorgegebener Anzahl von Ausgabeelemente. So können gegebenenfalls mehr Eingabeelemente gesammelt werden bevor der Algorithmus aufgerufen wird und möglicherweise ohne zu arbeiten zurückkehrt, weil ihm nicht ausreichend Daten zur Verfügung stehen.

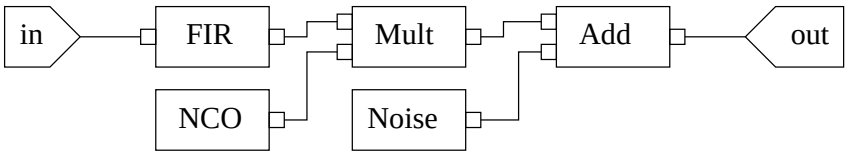


Abbildung 3.5: Implementierung eines einfachen simulierten Kanals

Für die häufigsten Anwendungsfälle liefert das Framework erweiterte Basisklassen mit. Das sind *gr_sync_block*, *gr_sync_decimator* und *gr_sync_interpolator* (s. Abb. 3.2.2). Sie entsprechen Transformationen mit den Übersetzungsverhältnissen 1:1, N:1 und 1:N, wobei N eine ganze Zahl ist. Die Funktionen *general_work()* und *forecast()* sind implementiert. So muss man nur noch die neue Funktion *work()* implementieren. In dieser werden keine Elemente mehr konsumiert. Dies erledigen die erweiterten Basisklassen automatisch an Hand des festgelegten Übersetzungsverhältnisses.

3.2.3 Laufzeitsystem

Wie bereits erwähnt, liefert das Framework auch die Werkzeuge um die Modelle auszuführen. Vor der Ausführung erfolgt erst der Analyse- und Initialisierungsschritt. Der Graph wird auf seine Richtigkeit überprüft, also dass er tatsächlich azyklisch ist und alle Ports verbunden sind. Danach wird er quasi platt gedrückt. Sämtliche Hierarchien werden aufgelöst. Das Produkt ist ein flacher, azyklischer und gerichteter Graph. Nun werden für alle Verbindungen die Zwischenspeicher angelegt. Für jeden Ausgangsport gibt es einen Speicherbereich. Über besondere Tricks mit der Memory Management Unit im Computer erzeugt das Framework zyklische Speicher und ordnet sie den Ports zu. Jeder Eingangsport erhält Lesezugriff auf den Ausgangsspeicher, mit dessem Port er verbunden ist. Sind mehrere Eingänge mit einem Ausgang verbunden, existiert nur ein Speicher. Dies ist besonders effizient, da keine Daten verdoppelt werden müssen. Und insbesondere erleichtert dieses Vorwissen die Synchronisierung der Blöcke untereinander. Denn Elemente werden erst freigegeben, wenn auf jeden Fall jeder Nachfolgeblock sie gelesen bzw. konsumiert hat.

3.2.4 Beispiel: Simulation eines statischen Kanals

Es wird beispielhaft gezeigt wie man einen Block implementiert, der einen einfachen Übertragungskanal simuliert. Dabei werden drei Einflüße berücksichtigt. Erstens die statische Mehrwegeausbreitung, zweitens eine Frequenzverschiebung und drittens eine einfache Rauschquelle. Abbildung 3.2.4 zeigt den schematischen Aufbau mit den internen Blöcken und den Ports.

Die Mehrwegeausbreitung wird durch einen FIR-Filter mit komplexwertigen Filterkoeffizienten modelliert. Die Koeffizienten sollen von einem beliebigen Kanalmodell berechnet worden sein. Dies kann z.B. ein Rayleigh-Fading-Modell sein. Auf die Generierung wird hier nicht weiter eingegangen. Der benötigte GNU Radio-Block hat den Namen

gr.fir_filter_ccc. Die Endung ccc gibt an, dass Eingangs-, Ausgangssignal und die Filterkoeffizienten komplexwertig sind.

Die Modulation des eingehenden Signals mit einer Sinuswelle modelliert die gewünschte Frequenzverschiebung. Dazu wird ein Sinussignal fester Frequenz mit dem eingehenden Signal multipliziert. Die GNU Radio-Blöcke dafür sind *gr.sig_source_c* und *gr.multiply_cc*. Auch hier geben die Endungen an, dass die Ein- und Ausgänge komplexe Werte erwarten beziehungsweise produzieren.

Zum Schluß wird der Ausgang einer einfachen Rauschquelle mit Gauß-Verteilung hinzu addiert. Der Block *gr.noise_source_c* erzeugt ein gauß-verteiltes Zufallssignal. Mit *gr.add_cc* wird dies zum Datensignal hinzugefügt.

Die Implementierung trägt den Klassennamen "*simple_channel*". Sie ist als ein hierarchischer Block in Python definiert. Ihre Initialisierungsparameter sind die RMS-Amplitude der Rauschquelle, die Frequenzverschiebung auf die Abtastfrequenz normiert und die Koeffizienten für den FIR-Filter. Die Signatur der Ein- und Ausgangsports ist identisch. Dem Framework wird über die Signaturen mitgeteilt, dass es mindestens einen Port, und maximal einen Port für je Aus- und Eingang gibt. Also muss jeder Port unbedingt verbunden sein. Die Datenbreite entspricht der Bytegröße von komplexen Werte.

Nach den Instantierungen und Initialisierungen der internen Blöcke werden die Verbindungen hergestellt. Dazu stellt die Basisklasse für hierarchische Blöcke eine entsprechende connect-Funktion zur Verfügung. Die Implementierung in Python ist nachfolgend angeführt.

```
class simple_channel(gr.hier_block2):
  def __init__(self, noise_rms, frequency_offset, channel_coefficients):
    gr.hier_block2.__init__(self, "simple_channel", # Blocktype Identifier
        gr.io_signature(1,1,gr.sizeof_gr_complex),  # incoming
        gr.io_signature(1,1,gr.sizeof_gr_complex))  # outgoing

    # for example channel_coefficients = [0.5+0.1j, 0.2-0.01j]
    multipath_sim = gr.fir_filter_ccc(1, channel_coefficients)

    # frequency_offset normalized to sampling frequency
    # amplitude = 1.0, DC offset = 0.0
    offset_src = gr.sig_source_c(1, gr.GR_SIN_WAVE, frequency_offset, 1.0, 0.0)
    mix = gr.multiply_cc()

    # noise_rms -> var(noise) = noise_rms**2
    noise_src = gr.noise_source_c(gr.GR_GAUSSIAN, noise_rms/sqrt(2))
    add_noise = gr.add_cc()

    # describe signal paths
    self.connect(self, multipath_sim)       # incoming port
    self.connect(multipath_sim, (mix,0))
```

```
    self.connect(offset_src, (mix,1))
    self.connect(mix, (noise_add,0))
    self.connect(noise_src, (noise_add,1))
    self.connect(noise_add, self)           # outgoing port
```

4 Zeitsynchronisierung

4.1 Schmidl & Cox Algorithmus

4.1.1 Vorstellung

In [SC97] beschreiben die beiden Autoren Timothy Schmidl und Donald Cox einen Algorithmus zur Zeitsynchronisierung von OFDM-Systemen, der auf der Beobachtung fußt, dass identische Blöcke aus Abtastwerten bei der Übertragung in einem quasistationären Kanal, der vereinfacht als LTI-Kanal betrachtet werden kann, gleichermaßen gestört werden. Die identischen Blöcke sind auch nach dem Kanal annähernd gleich.

Fügt man, wie bei den Daten-OFDM-Blöcken, das zyklische Präfix zur Abschwächung von Störungen durch Mehrwegeausbreitung an, wirken fast alle Störungen, bis auf das Rauschen, im quasistationären Fall gleichermaßen auf die identischen Abschnitte. Unter Ausnutzung dieser Eigenschaft haben die Autoren eine Metrik definiert, die nach einer periodischen Struktur im Empfangssignal sucht.

4.1.2 Senderseite

Ihr Algorithmus setzt einen Block mit zwei identischen Hälften als Präambel (siehe Abbildung 4.1) vor jeden OFDM-Rahmen. Der Empfänger berechnet die Schmidl & Cox-Metrik und sucht mit Hilfe dieser den idealen Abtastzeitpunkt. Um eine Rückgewinnung des Signals ohne Verluste durch Interferenzen zu gewährleisten, muss der geschätzte Zeitpunkt innerhalb des Bereichs liegen, der nicht durch ein vorher gesendetes Signal gestört wird. Durch die Mehrwegeausbreitung werden die gesendeten Abtastwerte „verschmiert“. Die ersten Abtastwerte im zyklischen Präfix der Präambel werden durch verzögerte Werte des vorhergehenden Blocks überlagert. Die Länge des Bereichs, indem eine verlustlose Rückgewinnung möglich ist, weil er nur aus Überlagerungen desselben Blocks besteht, entspricht gerade der Differenz des zyklischen Präfix und der längsten anzunehmenden Verzögerung eines Signalpfades, L. Entspricht 0 dem Beginn des Blocks ohne Präfix, so sind Zeitpunkte im Intervall $(-N_g + L, 0]$ als ideal zu betrachten.

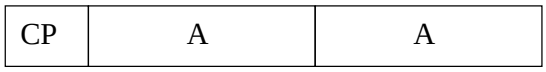


Abbildung 4.1: Präambel im Zeitbereich

Für die Präambel $s_{p_1}(k)$ gilt

$$s_{p_1}(k) = s_{p_1}(k + N/2) \quad \forall\, k = 0\,..\,N/2 - 1 \tag{4.1}$$

beziehungsweise

$$s_{p_1} = [\, A_{N/2} \quad A_{N/2}\,]. \tag{4.2}$$

Prinzipiell ist die Modulation der Unterträger nicht relevant. Die Autoren schlagen vor, die Symbole aus der QPSK-Konstellation oder einer Untermenge des 64-QAM-Signalsraums zu wählen, wobei die Amplitude so gewählt sein soll, dass die Energie der Blöcke gleich mit der von voll besetzten Blöcken ist. Sei $\mathcal{I}_{gz} \subset \mathcal{I}$ die Menge der Unterträger mit geradzahligem Index. Die Abbildung

$$I_{gz}(k) : [0, N_s/2) \rightarrow \mathcal{I}_{gz} \tag{4.3}$$

sei bijektiv. Sie ist eine eindeutige Zuordnung auf die geradzahligen Unterträger. Im Frequenzbereich hat die Präambel die Form

$$\begin{array}{lll} S_{p_1,k} & = X_{1,n} \in \mathbb{C}, & k = I_{gz}(n) \quad \forall\, n = 0\,..\,N_s/2 - 1 \\ S_{p_1,k} & = 0 & k \in \mathcal{I} \setminus \mathcal{I}_{gz} \end{array} \tag{4.4}$$

Sie erstellt man einmalig, z.B. beim Entwurf des Systems. Dazu generiert man $N/2$ komplexe Symbole $X_{1,n}$ mit einem Zufallsgenerator, z.B. aus der Menge

$$X_{1,n} \in \{+1 + j, +1 - j, -1 + j, -1 - j\}$$

Dann verfährt man weiter wie bei einem Datenblock. Die $N/2$ Symbole werden auf die geradzahligen Unterträger verteilt. Die restlichen Unterträger werden zu Null gesetzt. Nach der IFFT erhält man einen Block Abtastwerte mit der gewünschten Eigenschaft und fügt das zyklische Präfix vorne an.

4.1.3 Empfangsseite

Nun zum Empfänger. Sei $r(d)$ das empfangene Signal, dann berechnen wir

$$P(d) = \sum_{n=0}^{N/2-1} r^*(d + n) r(d + N/2 + n) \tag{4.5}$$

und

$$R(d) = \sum_{n=0}^{N/2-1} |r(d + n + N/2)|^2 \tag{4.6}$$

wobei $(.)^*$ die komplexe Konjugation bezeichnet. Die Metrik nach [SC97]

$$M(d) = \frac{|P(d)|^2}{(R(d))^2} \tag{4.7}$$

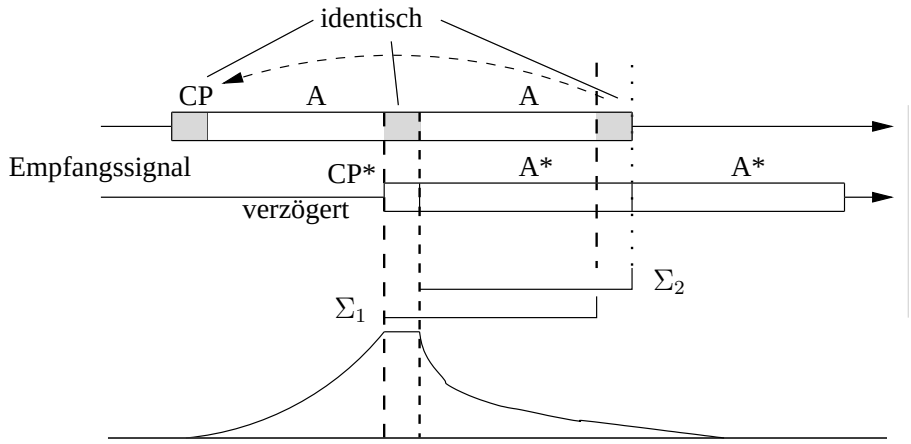


Abbildung 4.2: Grafische Darstellung der Schmidl & Cox-Metrik

nimmt Werte zwischen 0 und 1 an. Sie erreicht ihr Maximum gerade bei Koinzidenz der beiden Halbblöcke. Sei d_{opt} der optimale Zeitpunkt, dann gilt unter Vernachlässigung aller Störungen beispielhaft

$$r(d_{opt} + n) = r(d_{opt} + n + N/2) \quad \forall\, n = 0\,..\,N/2 - 1. \tag{4.8}$$

Somit vereinfacht sich Gleichung (4.5) zu

$$P(d_{opt}) = \sum_{n=0}^{N/2-1} |r(d_{opt} + n)|^2 = R(d_{opt}) \tag{4.9}$$

und es gilt

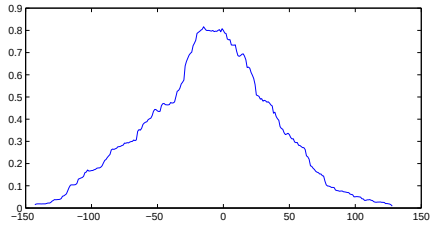
$$M(d_{opt}) = 1. \tag{4.10}$$

Durch das zyklische Präfix gilt dies aber auch für $d_{opt} - N_g < d \leq d_{opt}$. Es ensteht ein Plateau dessen Ende dem idealen Zeitpunkt entspricht. Die Funktion ist in Abbildung 4.1.3 skizziert. Durch die additive Überlagerung mehrerer Signalpfade im Empfänger reduziert sich die Plateaulänge. Zudem wurde in [SC97] gezeigt, dass die Plateauhöhe vom Signal zu Rausch-Verhältnis abhängt. Des weiteren verformen Abweichungen von der Annahme der Quasistationarität des Kanals das Plateau.

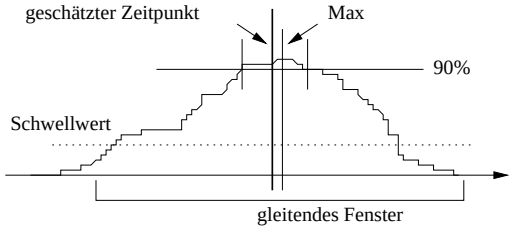
4.1.4 Plateau-Erkennung

Zur Erkennung des Plateaus wurden in [SC97] zwei Vorschläge gemacht. Bei beiden Verfahren wendet man zuerst einen Schwellwert auf die Metrik an. Werte unterhalb der Schwelle werden nicht beachtet.

Nun kann man z.b. einfach nach dem Maximum innerhalb eines Fensters der Länge N suchen und dieses Fenster über die Metrik bewegen. Die Maximumssuche sollte nur Werte



(a) Typische Plateauform bei 10dB SNR, ohne Mehrwegeausbreitung



(b) Erkennung des Plateaus

Abbildung 4.3: Zum Plateau der Metrik

über dem Schwellwert beachten. So finden wir in den meisten Fällen einen Punkt innerhalb des Plateaus. Ein Punkt sollte als (lokales) Maximum erkannt werden, wenn im Abstand von $N/2$ Abtastwerten um ihn herum kein größerer Wert existiert und der Wert des Punkts größer Null ist.

Wird das Fenster auf Grund eines falsch geschätzten Zeitpunkts ausserhalb des idealen Bereichs platziert (der am wenigsten gestörte Bereich ist ideal), so sinkt das Verhältnis der Signalleistung zur Störleistung dadurch. Bei schwachem Nutzsignal verschlechtern Fehlschätzungen die Situation unnötig.

Viel robuster ist der zweite Vorschlag der Autoren. Sie suchen erst nach dem Maximum. Dann finden sie die 90%-Punkte links und rechts vom Maximum, die 90% der Amplitude des Maximums aufweisen. Der geschätzte Zeitpunkt ist dann der Mittelpunkt dieses Intervalls. Mit diesem Vorgehen findet man wahrscheinlicher einen Zeitpunkt innerhalb des Plateaus.

4.1.5 Rekursive Form

Betrachten wir die Gleichungen 4.5 und 4.6. Diese lassen sich in eine rekursive Form bringen.

$$P(d+1) = P(d) + r^*(d+N/2)r(d+N) - r^*(d)r(d+N/2) \tag{4.11}$$

$$R(d+1) = R(d) + |r(d+N/2)|^2 - |r(d)|^2 \tag{4.12}$$

Der typische Anwendungsfall in praktischen OFDM-Systemen ist die sukzessive Berechnung der Metrik. Sie wird durch die neue Form der Gleichungen enorm beschleunigt. Die Anzahl der Operationen pro neuem Metrikwert reduziert sich stark. Dies ist ein großer Vorteil gegenüber den anderen vorgestellten Algorithmen. Der Nachteil ist jedoch ein gegebenenfalls höheres Quantisierungsrauschen durch die schlechtere Konditionierung des Problems.

4.1.6 GNU Radio Implementierung

Mit der GNU Radio-Architektur lässt sich die Metrik mit den Standardblöcken aus der Bibliothek einfach konstruieren (Abb. 4.4).

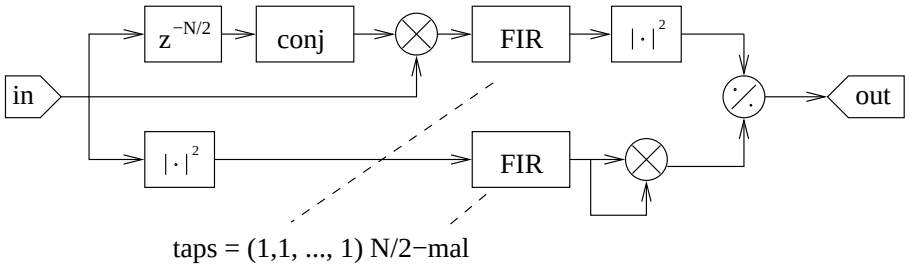


Abbildung 4.4: Darstellung einer möglichen Umsetzung mit GNU Radio

Dazu wird das Eingangssignal zuerst verzögert. Dies entspricht dem Block „z^{-N}“ in Abbildung 4.4. Eigentlich kopiert dieser Block das Signal nur 1:1, teilt aber bei seiner Initialisierung dem Framework mit, dass einige Nullen anfangs in seinen Ausgangsspeicher zu laden sind.
Die verzögerte Signalkopie wird dann komplex konjugiert und anschließend mit dem Original multipliziert. Als Ausgabe erhalten wir das Zwischenprodukt

$$r^*(d - N/2)r(d).$$

Gleichung 4.5 wird umgeschrieben zu

$$P(d) = \sum_{n=0}^{-N/2+1} r^*(d - N/2 + n)r(d + n)$$

um realisierbar zu sein, sodass nun im Prinzip auf das Zwischenprodukt eine Kurzzeitintegration angewendet wird. Dies erledigt der FIR-Filter. Seine Filterkoeffizienten sind $N/2$ Einsen. Den Abschluss des oberen Zweiges bildet die Berechnung des Absolutquadrats der Summe.

Im unteren Zweig lassen wir das Absolutquadrat der Eingabe berechnen. Es wird ebenfalls aufsummiert. Wir erhalten als Zwischenprodukt

$$\sum_{n=0}^{-N/2+1} |r(d + n)|^2 = R(d).$$

Als letzte Stufe bauen wir die Division von $|P(d)|^2$ und $(R(d))^2$ ein und erhalten als Ausgang $M(d)$. Das Ende des Plateaus fällt in dieser Implementierung auf den letzten Abtastwert des Präambelblocks. Genau dann integriert der obere FIR-Filter über das elementweise Produkt der beiden Hälften und der untere Filter über das Absolutquadrat der Elemente der zweiten Hälfte.

Die Ausgabe des hierarchischen Blocks sehen wir in Abbildung 4.5. Die FFT-Länge beträgt 512 Abtastwerte und das Präfix enthält 51 Werte. Die Präambel wurde mit der Funkhardware des Lehrstuhls bei einer Bandbreite von 8 MHz und einer Trägerfrequenz von 2.46 GHz übertragen.

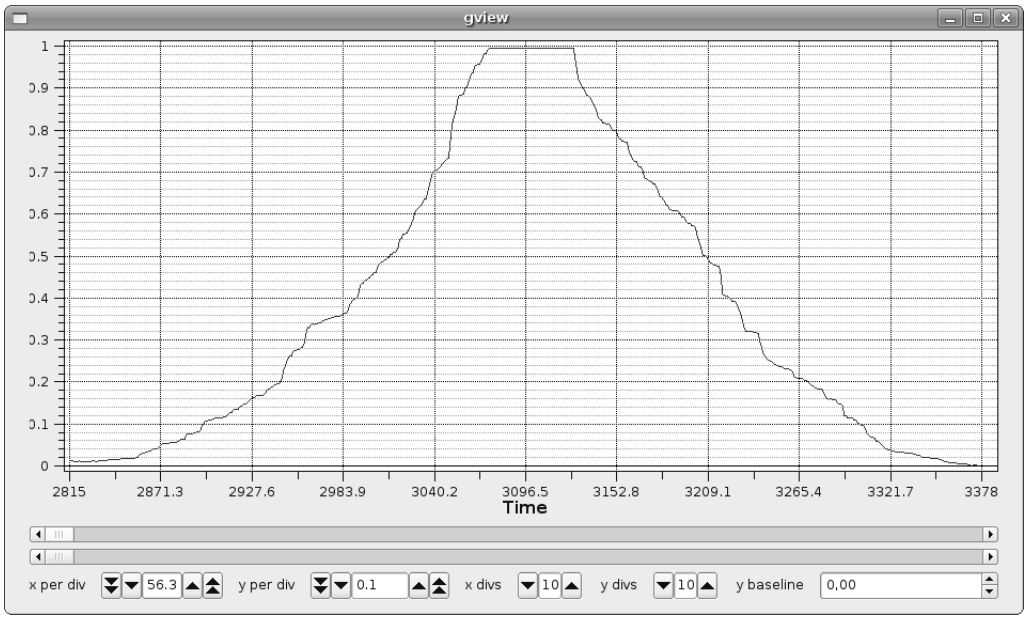


Abbildung 4.5: Bildschirmfoto der Ausgabe des GNU Radio-Blocks

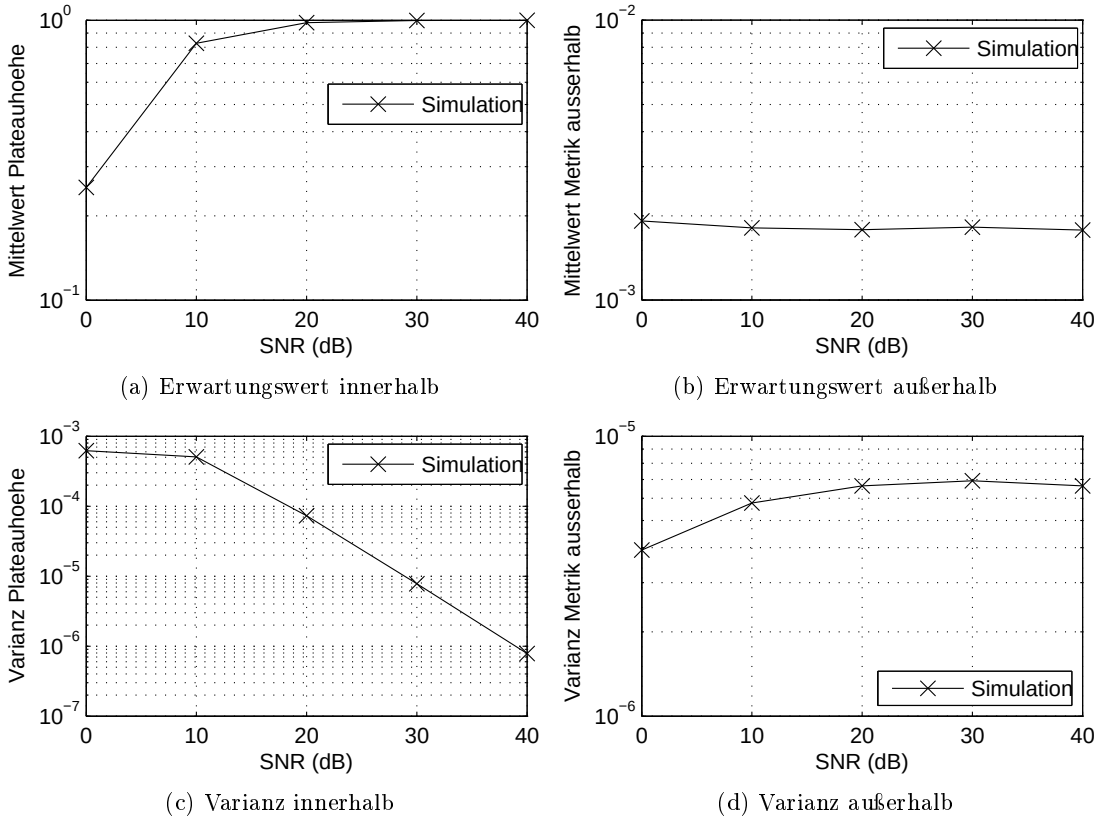
4.1.7 Matlab-Simulation

Mit einer Matlabsimulation werden der oben vorgestellte sowie die folgenden Algorithmen evaluiert. Sie simuliert einen Übertragungskanal mit Rayleigh-Fading und Gaußschem Rauschen. Wenn nicht anders angegeben, werden jeweils 10.000 Simulationsdurchläufe durchgeführt. Je Lauf ist der Kanal statisch. Es werden mehrere Rayleigh-Pfade simuliert. Diese werden zu Beginn jedes Laufs zurückgesetzt. Ihre Amplituden und Verzögerungen sowie Anzahl werden von der jeweiligen Simulationseinstellung bestimmt. In jedem Durchlauf werden die Koeffizienten der Pfade sowie die Realisierungen des Rauschprozess neu generiert.

Es besteht keine Korrelation zwischen aufeinanderfolgenden Simulationsläufen. Durch Multiplikation der Signalform mit einem Sinus fester Frequenz wird die Frequenzverschiebung zwischen Sender und Empfänger nachgestellt. Zudem kann ein Versatz der Abtastfrequenzen erzeugt werden. Hierzu wird das Signal mit reellen Faktoren interpoliert. Die Phasenverschiebung des ersten Abtastwerts kann beliebig verändert werden.

4.1.8 Verifikation der Simulation

Zur Überprüfung der Aussagekraft der Simulation wurden die Ergebnisse aus [SC97] nachvollzogen. Die FFT-Länge beträgt 1024 Werte. Es werden 1000 Unterkanäle benutzt. Die Länge des zyklischen Präfix beträgt 102 Abtastwerte. Es gibt 16 Rayleigh-Pfade. Der erste hat keine Verzögerung, während die weiteren Pfade äquidistant alle 4 Werte folgen. Ihre



(a) Erwartungswert innerhalb

(b) Erwartungswert außerhalb

(c) Varianz innerhalb

(d) Varianz außerhalb

Abbildung 4.6: Eigenschaften der Schmidl & Cox-Metrik

Amplituden berechnen sich nach

$$A_i = e^{-\frac{\tau_i}{60}} \quad \tau \in \{0,\, 4,\, ..,\, 60\,\} \tag{4.13}$$

mit τ_i als Verzögerung des i-ten Pfads. Sie entsprechen den Erwartungswerten der jeweiligen Pfadgewinne. Der Frequenzversatz entspricht 10.4 mal dem Unterträgerabstand. Die Abtastfrequenzen seien synchronisiert. In jedem Durchlauf wird eine neue PN-Sequenz für die Präambel generiert.

Die Abbildungen 4.6a, 4.6b, 4.6c und 4.6d stellen den Zusammenhang der Varianz und des Erwartungswerts der Metrik mit dem Signal-zu-Rausch-Verhältnis da. Gemessen wurde am optimalen Zeitpunkt innerhalb der Präambel und an einem Punkt ausserhalb, genau einen Block später.

Der gemessene Zusammenhang zwischen Erwartungswert der Plateauhöhe und SNR ist nochmals in Abbildung 4.7 dargestellt. Die gestrichelten Linien entsprechen drei Standardabweichungen über- und unterhalb der Mittelwertkurve. Die Werte der Plateauhöhe sind in guter Näherung gaußverteilt ([SC97]). Innerhalb des Intervalls der drei Standardabweichungen vom Mittelwert aus befinden sich so circa 99.7% der Werte.

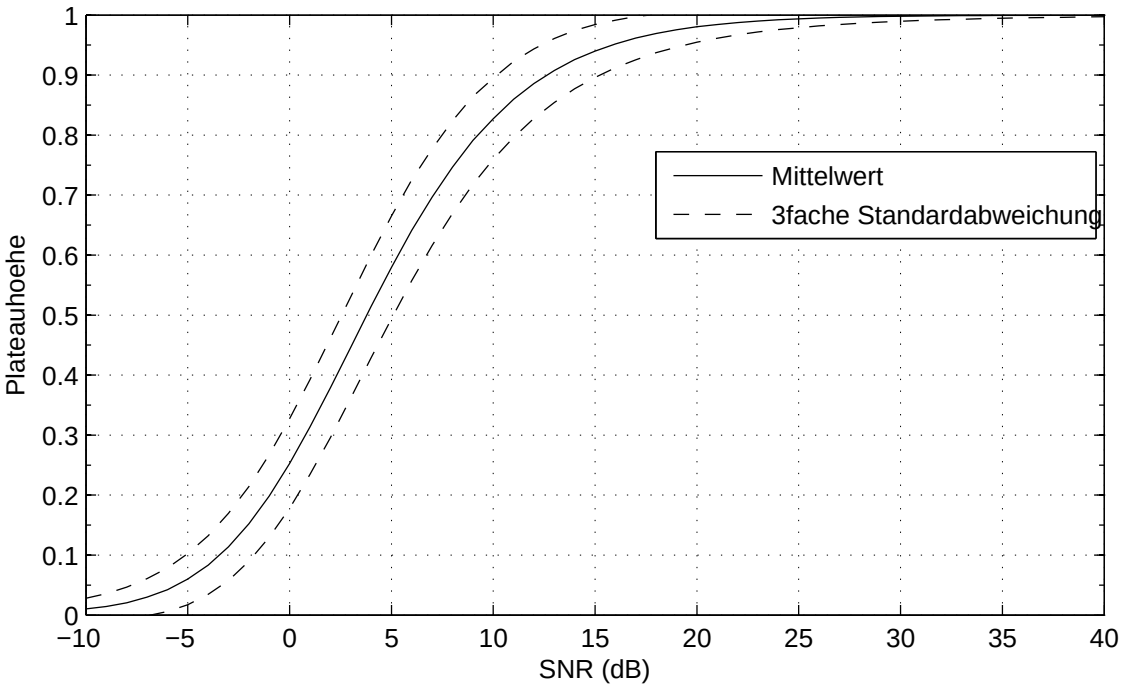


Abbildung 4.7: Erwartungswert in Abhängigkeit vom SNR

Die Ergebnisse der Simulation entsprechen im Rahmen der möglichen Genauigkeit den in [SC97] veröffentlichten Werten. Es kann davon ausgegangen werden, dass die Simulation ausreichende Aussagekraft besitzen.

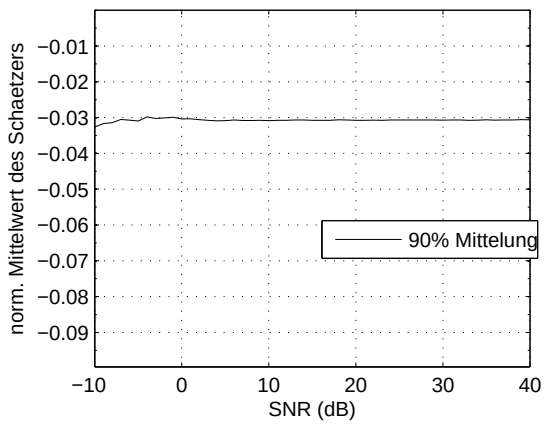
Über den gesamten dargestellten SNR-Bereich von 0 dB bis 40 dB hebt sich der Mittelwert des Plateaus deutlich von Werten ausserhalb der Präambel ab. Ab ungefähr 10 dB SNR hat das Plateau bereits annähernd den Wert 1.0, während der Wert ausserhalb der Präambel annähernd konstant bei $2 \cdot 10^{-3}$ verweilt. Über 10 dB sinkt zudem die Varianz der Plateauhöhe approximativ linear. Unter 10 dB scheint die Varianz nicht weiter zu steigen. In Simulationen mit geringeren Schrittweiten hat sich gezeigt, dass sie noch bis runter zu in etwa 5 dB weiter steigt, danach aber wieder abnimmt. Die Varianz der Werte ausserhalb ist sehr gering und bleibt im Intervall $4 \cdot 10^{-6}$ bis $7 \cdot 10^{-6}$.

Bei sinnvollen SNR-Werten sind die Werte der Metrik annähernd gaußverteilt [SC97]. Am idealen Zeitpunkt d_{opt} gilt

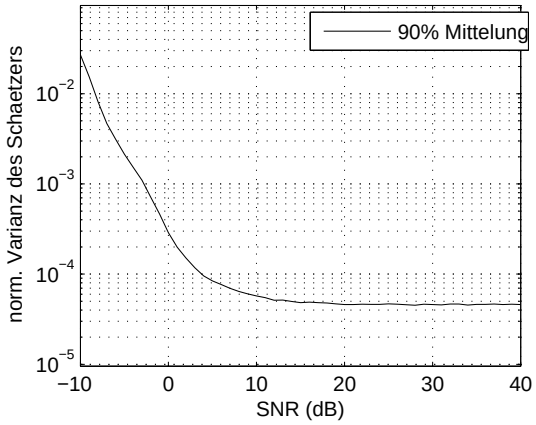
$$E[M(d_{opt})] = \frac{\sigma_s^4}{\sigma_s^2 + \sigma_n^2} \tag{4.14}$$

für den Erwartungswert. Die Varianz ist

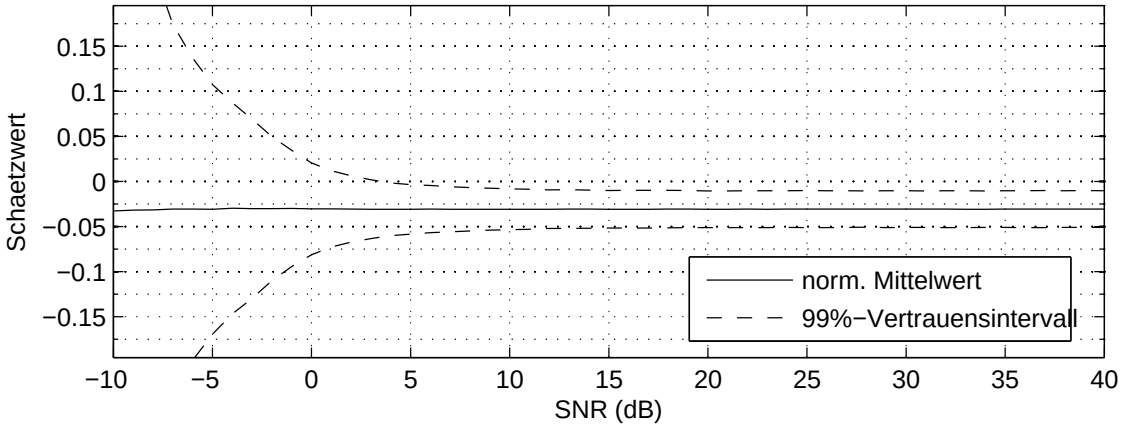
$$var[M(d_{opt})] \approx \frac{2}{N \cdot SNR} \tag{4.15}$$



(a) Mittelwert des 90%-Schätzers



(b) Varianz des 90%-Schätzers



(c) 90%-Schätzer

Abbildung 4.8: S&C-Schätzalgorithmus mit Parametern aus Originalartikel

falls das SNR ausreichend groß ist. Für die Werte ausserhalb der Präambel ist der Erwartungswert

$$E[M(d_{outside})] = \frac{1}{N/2} \tag{4.16}$$

und die Varianz

$$var[M(d_{outside})] = \frac{1}{(N/2)^2}. \tag{4.17}$$

Zusätzlich zur Berechnung der Metrik wurde die Schätzung des optimalen Zeitpunkts mit der 90%-Methode durchgeführt, unter anderem in Abbildung 4.3b dargestellt. In den Abbildungen 4.8a und 4.8b ist der Mittelwert und die Varianz des Schätzer über dem Signal zu Rausch-Verhältnis aufgetragen. Der Beginn des Präambelblocks ohne Präfix entspricht dem Wert 0. Negative Werte sind Zeitpunkte vor dem Blockbeginn. Alle Werte sind auf

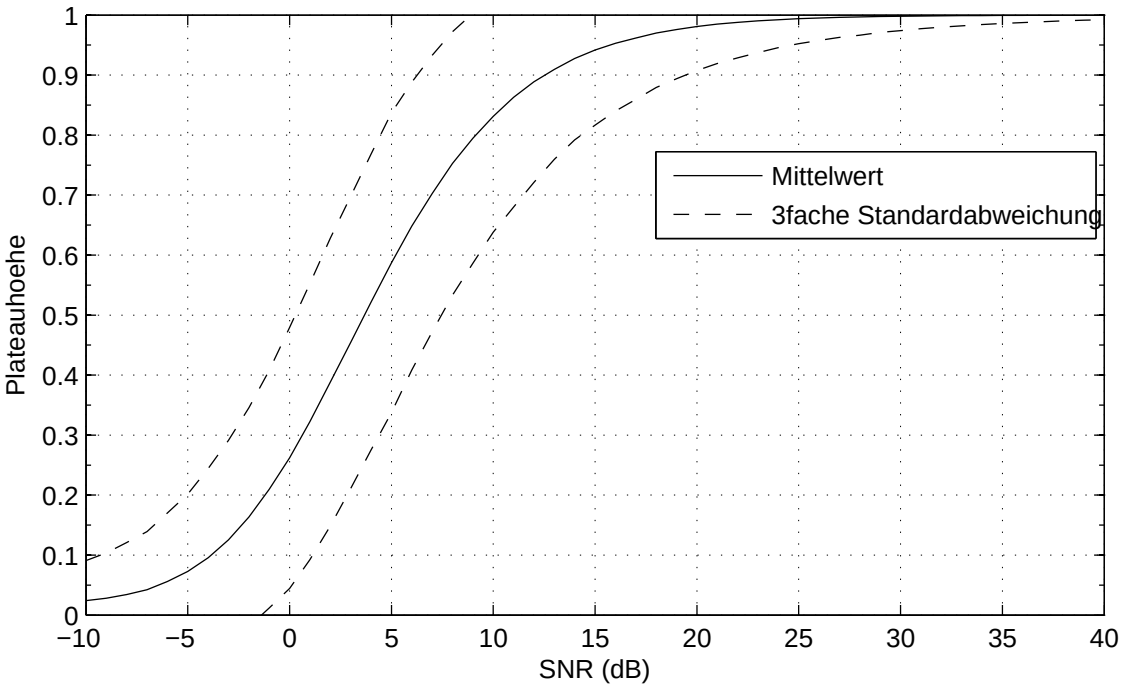


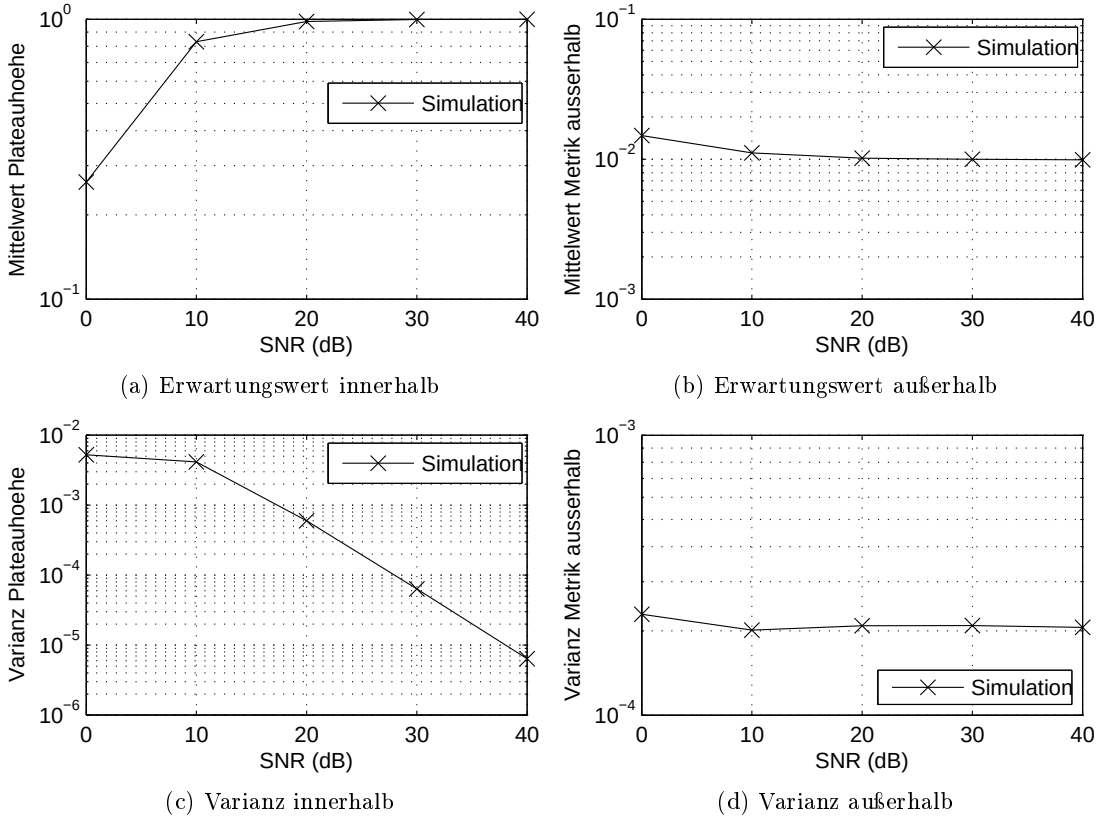
Abbildung 4.9: Metrikeigenschaften bei veränderter Parameterwahl

die Länge $N = 1024$ des Fensters normiert. Das Intervall des zyklischen Präfix liegt bei $[-0.1, 0]$.

Der Mittelwert ist sehr stabil. Selbst im Bereich von sehr kleinen SNR-Werten ist er annähernd in der Nähe von -30. Die Varianz ist ebenfalls sehr stabil und steigt nennenswert erst ab unter ungefähr 5 dB an. Mit einem Wert von ca. 50 bei hohen SNR-Werten ist sie aber ziemlich groß. Interessanterweise ist dieser Wendepunkt bei 5 dB der weiter oben besprochene Punkt, an dem Mittelwert und Varianz der Plateauhöhe anfangen stark abzuweichen. Wie man in Abbildung 4.8c, in der zusätzlich die dreifache Standardabweichung eingetragen ist, erkennt, ist es durchaus möglich, dass der geschätzte Zeitpunkt ausserhalb des optimalen Intervalls (dem ungestörten Teil des zyklischen Präfix) liegt. Innerhalb des zyklischen Präfix allerdings liegen die Schätzwerte mit sehr hoher Wahrscheinlichkeit.

4.1.9 Eigene Parameterwahl

Für die Realisierung des OFDM-Systems am Lehrstuhl für Thereotische Informationstechnik haben wir andere Parameter ausgewählt. Auch für diese wurde die Simulation durchgeführt. Gemessen wird wieder an den gleichen Punkten. Von 128 Unterträgern sind 100 in Benutzung. Die Länge des zyklischen Präfix ist 16. Es wird keine Mehrwegeausbreitung simuliert. Nur ein reiner AWGN-Kanal ist vorhanden. Der typische, bei unserer Hardware beobachtete, Frequenzversatz von 5 kHz bei einer typischen Bandbreite von 400 kHz



(a) Erwartungswert innerhalb

(b) Erwartungswert außerhalb

(c) Varianz innerhalb

(d) Varianz außerhalb

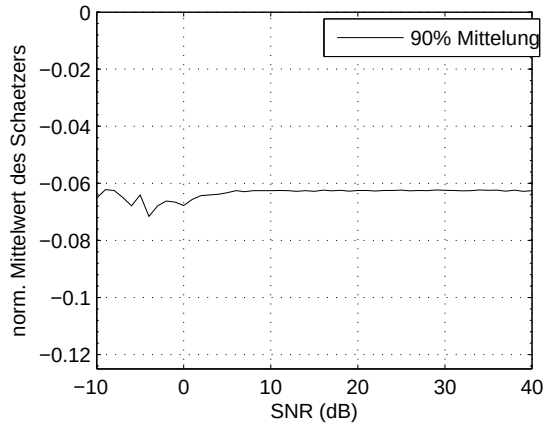
Abbildung 4.10: Metrikeigenschaften mit TI-Parametern

entspricht 1.6 Unterträgerabständen. Die Abtastfrequenzen sind synchronisiert.

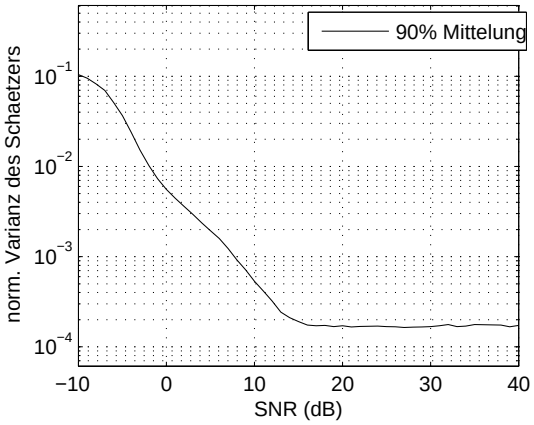
Der Mittelwert der Metrikwerte in Abhängigkeit des SNR, zusammen mit dem 99%-Vertrauensintervall, ist in der Abbildung 4.9 dargestellt. Beobachten kann man das deutlich vergrößerte Vertrauensintervall. Der Mittelwert hingegen ist im Vergleich zur vorherigen Simulation des Algorithmus annähernd gleich geblieben. Dies ergibt sich auch aus der analytisch hergeleiteten Beziehung 4.15 der Varianz der Werte zur Länge N des Fensters.

Im direkten Vergleich der Abbildungen 4.10a, 4.10b, 4.10c und 4.10d mit den korrespondieren Abbildungen der vorhergehenden Simulation wird die Verschlechterung des Schätzers durch das verkürzte Fenster noch deutlicher. Die Änderung von $N = 1024$ auf $N = 128$ verursacht bei den in den Abbildungen (b) - (d) dargestellten Eigenschaften eine Verschiebung um ungefähr eine Größenordnung. Wieder ist erkennbar, dass der Erwartungswert am optimalen Zeitpunkt (Abb. 4.10a) annähernd unberührt bleibt.

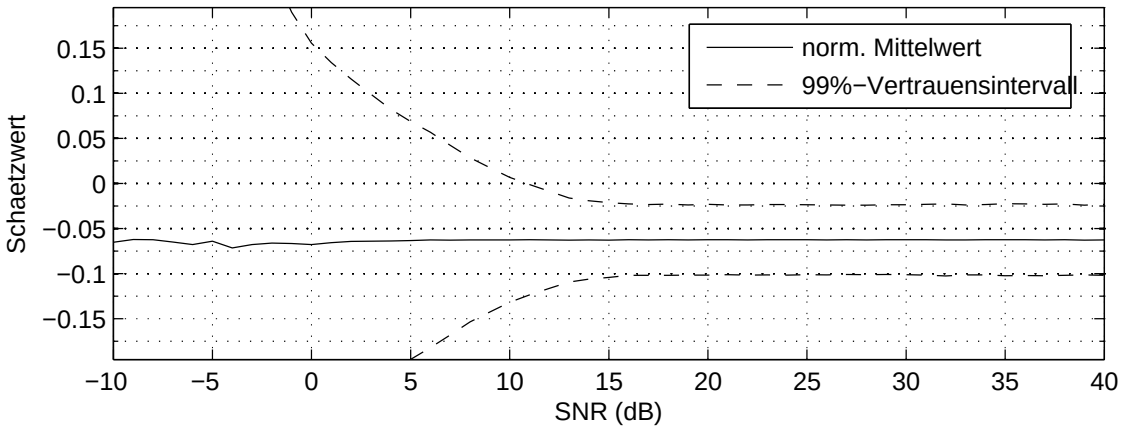
Die Performanz des 90%-Schätzers hängt stark von den Eigenschaften der Metrik ab, wie bereits die Simulation mit den vorhergehenden Parametern gezeigt hat. Dementsprechend schneidet diese im Vergleich auch schlechter ab. Erneut sind Mittelwert und Varianz auf die FFT-Länge $N = 128$ normiert. Sie sind in 4.11a und 4.11b abgebildet. In Abbildung



(a) Mittelwert des 90%-Schätzers



(b) Varianz des 90%-Schätzers



(c) 90%-Schätzer

Abbildung 4.11: S&C-Schätzer mit TI-Parametern

4.11c findet sich weiter noch das 99%-Vertrauensintervall um den normierten Mittelwert. Das zyklische Präfix entspricht dem Intervall $[0.125, 0]$.

Oberhalb von ungefähr 15 dB verhält sich der Schätzer sehr stabil. Das Vertrauensintervall liegt vollständig innerhalb des zyklischen Präfix. Bei den gewählten Parametern ist dies optimal. Da ein reiner AWGN-Kanal betrachtet wurde, ist der komplette Bereich des Präfix nicht durch Interferenzen der vorher ausgesendeten Blöcke gestört. Natürlich ist das Präfix in diesem Fall komplett überflüssig. Im Vergleich zur Simulation mit den ursprünglichen Parametern fällt auf, dass sich die Grenze für diese empirisch festgestellte Stabilität um in etwa 10 dB verschoben hat. Für die erste Simulation galt die Beobachtung für SNR-Werte oberhalb von 5 dB. Dies ist doch eine deutliche Verschlechterung, die sehr wahrscheinlich auf die verringerte Fensterlänge zurückzuführen ist.

Unterhalb dieser Grenze offenbart sich eine ähnlich “starke” Divergenz. Die Varianz der

Schätzwerte steigt drastisch an und verlässt bei circa 11 dB die Schwelle des zyklischen Präfix. Interessanterweise verlässt sowohl die untere als auch die obere Schwelle des 99%-Vertrauensintervalls bei demselben SNR-Wert das Präfix-Intervall. Der Mittelwert ist ziemlich genau auf den Mittelpunkt des Präfix zentriert. Demgegenüber war der Mittelwert des Schätzers in der ersten Simulation nicht auf das zyklische Präfix, sondern annähernd auf dem Mittelpunkt des ungestörten Bereichs des Präfix platziert.

4.1.10 Einfluss eines Abtastratenversatz

Es wurde ebenfalls der Einfluss eines Versatz der Abtastraten im Empfänger und Sender untersucht. Wie in der Einleitung bereits erläutert, können die Oszillatoren der AD-Wandler nicht perfekt aufeinander abgestimmt sein. Aus Sicht des Empfängers kommt es scheinbar zu einer Dehnung bzw. Stauchung der Präambel. Diese Verzerrung zerstört die Periodizität. Die Korrelation zwischen den beiden Hälften sinkt. Dies beeinflusst natürlich Erwartungswert und Varianz der Schmidl & Cox-Metrik.

Die veränderten Eigenschaften sind in der Simulation ermittelt worden. Der simulierte Kanal entspricht dem aus [SC97]. Der SNR-Wert ist konstant auf 10 dB gesetzt. Die Abweichung der Takte wird prozentual in der Einheit ppm (Parts per Million) angegeben. Es wurde eine Bandbreite von 25 MHz angenommen. Im Abstand von 1 kHz wurde die Simulation mit Taktratenversätzen im Intervall von -10 kHz bis +10 kHz durchgeführt. Dies entspricht einer maximalen Abweichung von 400 ppm und einer Schrittweite von 40 ppm. Der Fehler des Mittelwerts sowie die neue Varianz der Plateauhöhe wurde untersucht. Als Fehler ist die Abweichung des Mittelwerts vom Mittelwert ohne Versatz, normiert auf diesen Mittelwert, definiert.

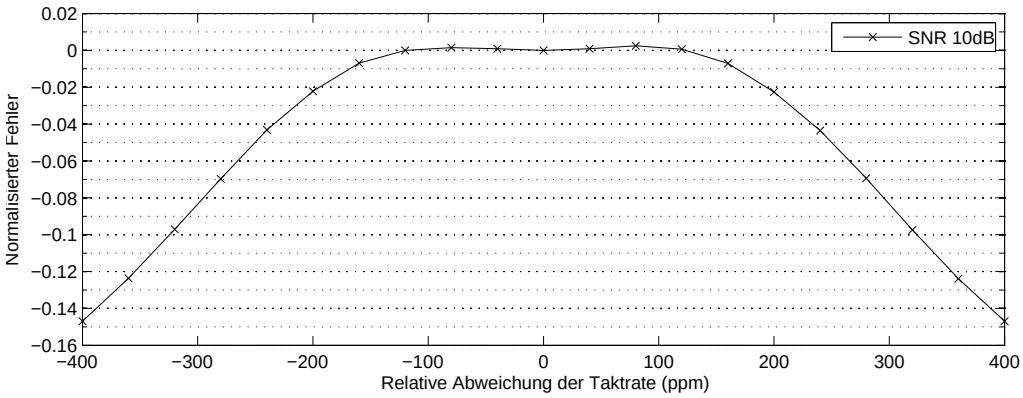
Sei μ_0 der Mittelwert bei $\Delta f_s = 0$. Der gemessene normierte Fehler ist dann

$$e_i = \frac{\mu_i - \mu_0}{\mu_0}, \tag{4.18}$$

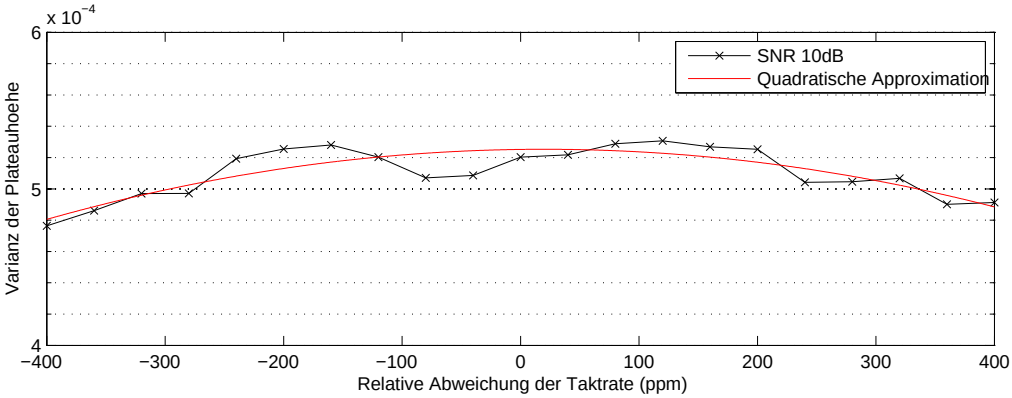
wobei μ_i der Mittelwert des i-ten Simulationslauf ist. Abbildung 4.12a stellt diesen Sachverhalt dar. In Abbildung 4.12b sind die Varianzen unverändert über dem Versatz aufgetragen. Zusätzlich ist eine Approximation mit einem quadratischen Polynom eingezeichnet.

Ob die Verzerrung eine Dehnung oder Stauchung ist, also ob der Takt beim Empfänger dem des Senders voraus- oder hinterhereilt, scheint irrelevant zu sein. In beiden Richtungen, ungefähr im Intervall -120 ppm bis +120 ppm verändert sich der Erwartungswert nicht nennenswert. Der normierte Fehler ist annähernd Null. Bei höheren Abweichungen nimmt der Fehler nun rapide zu. Bei geschätzten +/- 320ppm Abweichung entspricht er bereits 10%.

Die Veränderung der Varianz ist leider nicht symmetrisch. Betrachtet man die quadratische Approximation, lässt sich jedoch ein leichter Trend zu einer niedrigeren Varianz bei steigendem Versatz deuten. Allerdings liegen die Werte alle innerhalb derselben Größenordnung. Eine Abweichung könnte durchaus auch auf andere Faktoren zurückzuführen sein. Man kann aber festhalten, dass die Varianz der Plateauhöhe annähernd konstant bleibt.



(a) Mittelwert der Metrik



(b) Varianz der Metrik

Abbildung 4.12: Veränderte Metrikeigenschaften bei vorhandenem Abtastratenversatz

Auch unter diesen nicht idealen Bedingungen zeigt der Algorithmus eine sehr gute Robustheit. Ist der Versatz nicht größer als ungefähr 120 ppm scheint die Verzerrung keine nennenswerte Verschlechterung der Metrikeigenschaften zu bewirken.

4.1.11 Mögliches Kriterium zur Selektion einer Präambel

Die Wahl der Pseudozufallssequenz für die Präambel hat nach [SC97] keinen wesentlichen Einfluß auf Erwartungswert und Varianz der Metrik. Daher kann diese nach anderen Kriterien ausgesucht werden.

Eine mögliche recht interessante Optimierung, die unterem anderem in [SC97] angedeutet wird, ist die Reduktion des Verhältnis der Spitzenleistung zur mittleren Leistung. Auf Grund der Relevanz für den nächsten Algorithmus, aber weil diese Optimierung auch für diese Methode möglich ist, wird nun die zu optimierende Eigenschaft definiert und ein Lösungsansatz vorgestellt.

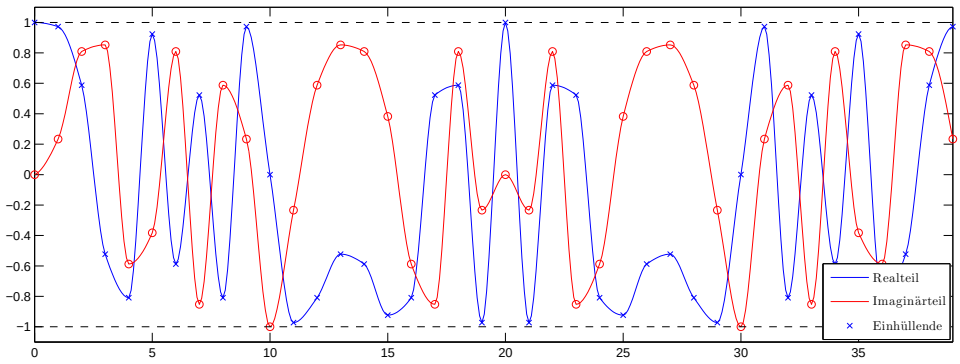


Abbildung 4.13: Beispiel CAZAC-Sequenz N = 40

Sei

$$s(k) \in \mathbb{C}, \quad k = 0..N-1 \tag{4.19}$$

ein OFDM-Block. Das Verhältnis der Spitzenleistung zur mittleren Leistung (Peak to Average Power Ratio) ist

$$PAPR = \frac{\max_k |s(k)|^2}{\frac{1}{N}\sum_{k=0}^{N} |s(k)|^2}. \tag{4.20}$$

Es wird üblicherweise in Dezibel angegeben. Offensichtlich wird ein Minimum für

$$|s(k)| = c, \quad c = const \quad \forall k \tag{4.21}$$

erreicht. Das Signal hat eine konstante Einhüllende.

An dieser Stelle soll nur eine kurze Konstruktionsvorschrift für diesen Sequenztyp gegeben werden. Der nächste vorgestellte Algorithmus verwendet eben diese.
Eine passend gewählte Sequenz wird auf die geraden Unterträger verteilt werden. Damit die Signalform nach der inversen FFT eine konstante Einhüllende besitzt, muss nach [JH04] die Sequenz eine Autokorrelation besitzen, die gerade 1 bei keiner Verschiebung und sonst 0 ist. Diese **Z**ero-**A**uto**c**orrelation-Eigenschaft führt zu einer konstanten Amplitude (gebräuchliche Bezeichnung ist **C**onstant **A**mplitude) im Zeitbereich. Idealerweise ist die Leistung gleichmäßig auf die Unterträger verteilt. Nach [JH04] führt diese CA-Eigenschaft wiederum zur ZAC-Eigenschaft der Sequenz im Frequenzbereich. Kombiniert man dies, kann man festhalten, dass eine CAZAC-Sequenz auf den Unterträgern zu einer CAZAC-Sequenz im Zeitbereich führt.

Eine Konstruktionsvorschrift für CAZAC-Sequenzen beliebiger Länge findet sich bei [Chu72]. Sei N die Länge der gewünschten Sequenz. Falls N gerade ist, so wähle

$$c_k = \exp\left(i\frac{M\pi k^2}{N}\right) \tag{4.22}$$

wo M eine ganze Zahl relativ prim zu N sei, also

$$\gcd(M, N) = 1. \tag{4.23}$$

Andernfalls für N ungerade wähle

$$c_k = \exp\left(i\frac{M\pi k(k+1)}{N}\right). \tag{4.24}$$

4.2 Guangliang Ren Algorithmus

Der Algorithmus vorgeschlagen in [GRZ05] versucht die Zeitsynchronisierung zu verbessern, indem der Wert am optimalen Zeitpunkt in der Metrik deutlicher hervorgehoben wird als benachbarte Werte. Statt einem Plateau erhalten wir eine Metrik, die der Impulsfunktion ähnelt.

Es wird wieder eine Präambel $s_p(k)$ und eine Metrik definiert. Die Präambel besitzt im Prinzip dieselbe Periodizität wie bei [SC97]. Desweiteren hat sie eine konstante Einhüllende. Es gilt

$$\left.\begin{array}{ll} s'_p(k) & = s'_p(k+N/2) \\ |s'_p(k)| & = c = const \end{array}\right\} \forall\, k = 0\,..\,N/2-1. \tag{4.25}$$

Sei nun eine Pseudozufallsequenz

$$p_k \in \{-1, +1\}, \quad k = 0\,..\,N-1 \tag{4.26}$$

gegeben. Die Präambel

$$s_p(k) = s'_p(k)p_k \tag{4.27}$$

wird mit den Werten der Pseudozufallsequenz gewichtet. Diese ist einmalig fest gewählt und Sender sowie Empfänger bekannt. Durch die Multiplikation mit der Präambel wird das Plateau in der Metrik vermieden. Man verifiziert leicht, dass weiterhin durch

$$|s_p(k)| = c \tag{4.28}$$

die CA-Eigenschaft vorhanden ist.

Die Metrik wird ebenfalls neu definiert. Mit den beiden veränderten Definitionen

$$P(d) = \sum_{n=0}^{N/2-1} p_n p_{n+N/2} r^*(d+n) r(d+N/2+n) \tag{4.29}$$

$$R(d) = \frac{1}{2}\sum_{n=0}^{N-1} |r(d+n)|^2 \tag{4.30}$$

ist die Form der Metrik

$$M(d) = \frac{|P(d)|^2}{(R(d))^2} \tag{4.31}$$

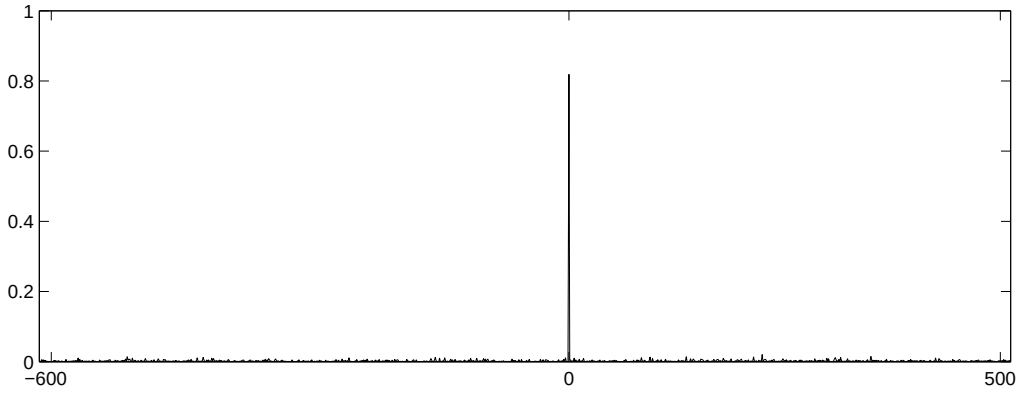


Abbildung 4.14: Typische Werte der Metrik bei 10dB SNR, ohne Mehrwegeausbreitung

ähnlich der aus [SC97]. Durch die neuen Terme in Gleichung 4.29 wird die Gewichtung mit der Pseudozufallsequenz aufgehoben. Gleichzeitig kann eine maximale Korrelation nur auftreten, wenn sowohl die Folge als auch die Periodizität vorhanden sind.

Die Summe in Gleichung 4.30 geht nun über die komplette Präambel. Diese Änderung führt zu einer geringeren Varianz gegenüber der alten Definition in [SC97], wie unter anderem in [HMB00] angeführt ist.
Abbildung 4.14 zeigt eine typische Ausprägung der neuen Metrikfunktion. Deutlich zu erkennen ist der optimale Zeitpunkt.

Bei der Methode aus [SC97] besitzen benachbarte Werte der Metrik eine ähnliche Summe. Betrachtet man die Gleichungen 4.11 und 4.12 erkennt man, dass nur zwei Produkte den Unterschied zwischen aufeinanderfolgenden Werten aus machen. Die Gewichtung durch die Pseudozufallsequenz p_k vergrößert diesen Abstand, indem im Prinzip die zusätzliche Korrelation mit der Sequenz als Filter wirkt.

Zur Detektion des optimalen Zeitpunkts sucht man das Maximum der Metrik. Aus praktischen Gründen betrachtet man nur Werte überhalb einer festen Schwelle. Die Wahl dieser Schwelle beeinflusst direkt die Wahrscheinlichkeiten für Fehlerkennung (optimaler Zeitpunkt nicht erkannt) und für die Erkennung falscher Zeitpunkte.

4.2.1 GNU Radio Implementierung

Für den Einsatz mit der GNU Radio Architektur wird dieselbe Struktur wie in Abbildung 4.4 benutzt. Nur die Filterkoeffizienten der beiden FIR-Filter müssen angepasst werden. Für den ersten Filter, in der Abbildung ist dies der obere, wählt man die Koeffizienten

$$a_{1,k} = s_{N/2-1-k} s_{N-1-k}, \quad \forall\, k = 0 \ldots N/2-1$$

also gerade die beiden Hälften der Pseudozufallsequenz miteinander elementweise multipliziert und rückwärts ausgegeben. Im weiten Sinne kann der Filter als “matched filter” betrachtet werden.

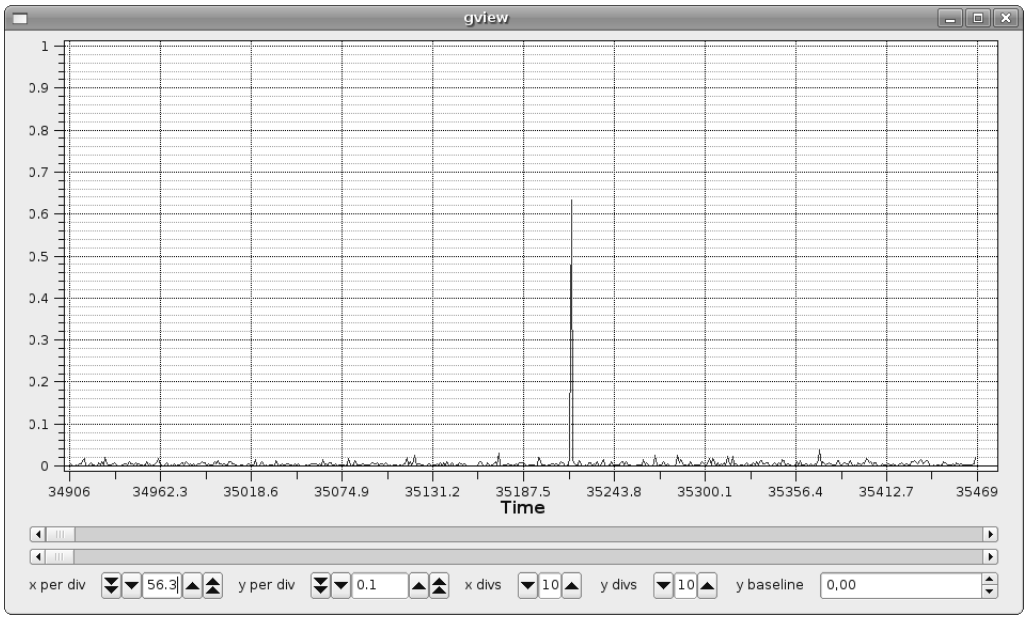


Abbildung 4.15: Bildschirmfoto der Ausgabe des GNU Radio-Blocks

Die Sequenz der Koeffizienten des zweiten Filters ist nun

$$a_{2,k} = \frac{1}{2}, \quad \forall\, k = 0 \ldots N-1.$$

Sie ist doppelt so lang wie beim ersten Algorithmus und gewichtet die Summenelemente nur mit $\frac{1}{2}$.
Ein Bildschirmfoto der Ausgabe des Blocks ist in Abbildung 4.15 zu sehen. Der Ausgabestrom ist derselbe wie beim ersten Algorithmus. Es wurde dabei abwechselnd die Präambel aus [SC97] und die neue Präambel mit der Funkhardware des Lehrstuhls gesendet. Die Bandbreite des gesendeten Signals betrug 8 MHz und die Trägerfrequenz war 2.46 GHz. Die FFT-Länge umfasste 512 Werte und das Präfix wiederum 51.

4.2.2 Simulation

Dieser Algorithmus wurde ebenso in die Simulation eingepflegt. Um einen empirischen Vergleich des zuerst vorgestellten Algorithmus ([SC97]) zu ermöglichen, sind die Parameter der Simulation identisch gewählt. Sie werden hier nochmal angeführt. Es werden 1000 Unterträger benutzt. Das Signal ist geringfügig überabgetastet. Die FFT erfolgt über 1024 Werte. Das zyklische Präfix enthält 102 Abtastwerte. Wiederum sind 16 Rayleigh-Pfade vorhanden, deren Amplituden die Erwartungswerte definiert in 4.13 haben. Der erste Pfad hat keine Verzögerung, während die weiteren mit einer Schrittweite von vier Abtastwerten konsekutiv verzögert sind. Das Signal wird im Frequenzbereich um 10.4 Unterträgerabstände verschoben. Zur Vereinfachung sind die Abtastfrequenzen identisch. Die PN-Sequenz,

mit der die Präambel multipliziert wird, genereien wir bei jedem Durchlauf neu. Während dessen ist die CAZAC-Sequenz, die auf den Unterträgern der Präambel verteilt ist, konstant. Es erfolgen jeweils 10.000 Simulationsläufe je Parametereinstellung.

Für jeden Durchlauf werden vier OFDM-Blöcke generiert. Der zweite Block ist durch den Algorithmus definiert und repräsentiert die Präambel. Die restlichen drei Blöcke sind Datenblöcke. Sie tragen zufällige Bitdaten, die per QPSK moduliert und auf die Unterträger verteilt werden.

Sei D_p eine Menge von ausgesuchten Indizes, an denen Maximalwerte auf Grund der Mehrwegeausbreitung erwartet werden. Das erste Element der Menge entspricht dem Ende des zyklischen Präfix. Für jeden simulierten Signalpfad kommt ein weiteres Element hinzu, wobei zum ersten Index die spezifische Verzögerung des Pfades addiert wird. D sei die Menge aller Indizes. Dann werden die drei Werte

$$\begin{aligned} w_1 &= \max_{d \in D_p} M(d) \\ w_2 &= \max_{d \in D \setminus D_p} M(d) \\ w_3 &= \arg\max_{d \in D_p}(M(d)) - d_{opt} \end{aligned} \tag{4.32}$$

in jedem Durchlauf aufgezeichnet. Sie entsprechen den Maxima der Metrik innerhalb und ausserhalb der Präambel sowie dem Versatz des geschätzten Zeitpunkts zum optimalen Zeitpunkt. Zur Analyse der Eigenschaften dieses Algorithmus werden sie als Realisierungen von Zufallsprozessen mit unbekannten Verteilungen betrachtet. Es wird versucht, die Eigenschaften zu erfassen indem die Verteilung der Werte abgeschätzt wird. Alle folgenden Schlussfolgerungen basieren nur auf empirischen Beobachtungen und haben keine analytisch belegte Fundierung.

Betrachten wir zuerst die Werte w_1 und w_2. Ähnlich dem zentralen Grenzwertsatz gibt es einen zentralen Satz über die Verteilungsfunktionen von Extrema beliebiger Zufallsprozesse. Dieser besagt, dass die Verteilungsfunktion für Extrema von einer Folge von Zufallswerten mit beliebiger Verteilung, für den Grenzübergang zu einer unendlich langen Folge sich der verallgemeinerten Extremwertverteilung ("generalized extreme value distribution", GEV) annähert, falls die Verteilung existiert.

Sei $X_1, X_2.., X_n$ eine Folge von unabhängigen und identisch verteilten Zufallswerten mit beliebiger Verteilung, sei weiterhin $M_n = max\{X_1, .., X_n\}$ das Maximum der Folge. Falls zwei Sequenzen $a_n, b_n \in \mathbb{R}$ existieren, so dass $a_n > 0$ und

$$\lim_{n \to \infty} P\left(\frac{M_n - b_n}{a_n} \leq x\right) = F(x) \tag{4.33}$$

dann gehört F, falls es keine degenerierte Verteilungsfunktion ist, zur Gumbel-, Fréchet- oder Weibull-Familie und ist damit Mitglied der GEV-Familie. Die Verteilung von M_n kann demnach durch die GEV-Verteilung angenähert und parametrisiert werden.

Die empirischen Resultate der Simulation zeigen tatsächlich annähernd eine derartige Verteilung, so dass die Verteilungen von w_1 und w_2 im Folgenden durch die verallgemeinerte Extremwertverteilung charaktisiert werden. Zum Abschätzen der Verteilungsfunktionen wurden die Histogramme für eine große Anzahl an Zwischenwerten begutachtet. Beobachtungen zeigten eine auffällige Ähnlichkeit zur GEV-Verteilung. Weitere numerische

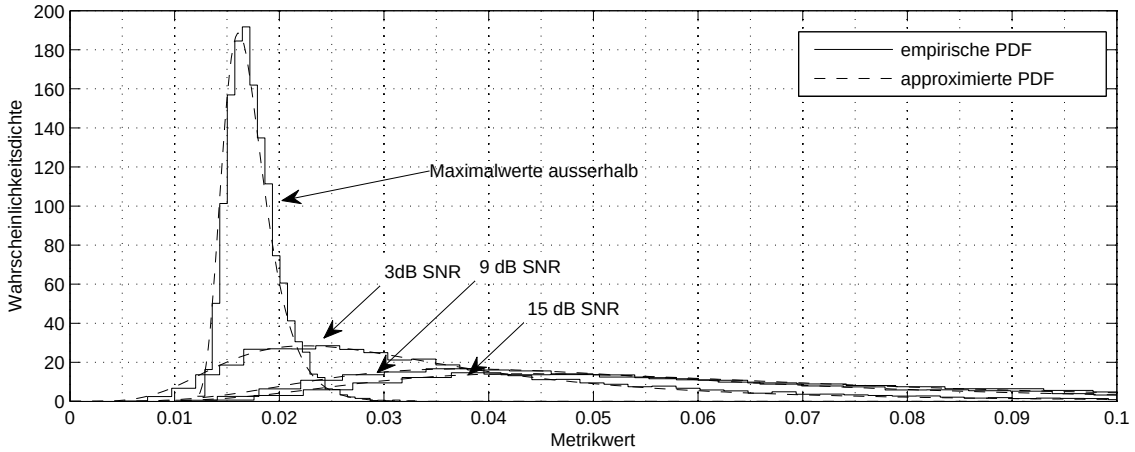


Abbildung 4.16: Empirisch bestimmte Wahrscheinlichkeitsdichtefunktionen

Berechnungen offenbarten dann, dass die Verteilungen scheinbar annähernd durch GEV-Verteilungen beschreibbar sind. Die Definition 4.32 in Kombination mit dem zentralen Grenzwertsatz für Extremwerte bestärken diese Argumentation.

Die geschätzten Verteilungen werden im Folgenden präsentiert. Sie sind das Ergebnis der Simulationen mit den SNR-Werten 3 dB, 9 dB und 15 dB. In Abbildung 4.16 sind die empirisch bestimmten und die approximierten Wahrscheinlichkeitsdichtefunktionen dargestellt. Die gestrichelten Kurven entsprechen den Approximationen der Verteilungen. Dabei gehören die flachen Kurven zur Zufallsvariable w_1 und die ausgeprägte hohe Kurve zur Variable w_2. Für letztere genügte zur Darstellung eine Kurve. Die Verteilungen für die verschiedenen SNR-Werte sind auf der Grafik nicht unterscheidbar. Für die Funktionen der Variable w_1 erkennt man, dass sich die Verteilung mit steigendem SNR-Wert in Richtung höherer Maxima verschiebt. Die untere Grenze der Verteilungsdichte bleibt, so weit dies erkennbar ist, ungefähr bei dem gleichen Wert 0.01. Des weiteren sind die Kurven ab circa 9 dB in dieser Darstellung kaum mehr differenzierbar. Vermutlich strebt die Dichte für größere SNR-Werte asymptotisch gegen eine Verteilung, die der dargestellten Funktion bei 15 dB ähnlich sein wird. Die Abbildung 4.17 zeigt nur die flachen Kurven, sodass die Darstellung etwas detaillierter ist.

Die Abbildung 4.18 zeigt die empirisch ermittelten Verteilungsfunktionen (und inversen) für die verschiedenen Simulationsdurchläufe. Für die Werte der Metrik-Extrema ausserhalb der Präambel, also für die Zufallsvariable w_2, gibt die Verteilungsfunktion $P(w_2 \leq x)$ die Wahrscheinlichkeit an, dass die Realisationen von w_2 unter der Schwelle x liegt. Für die Variable w_1 ist die Schar der inversen Verteilungsfunktionen eingezeichnet. Sie geben die Wahrscheinlichkeit an, dass w_1 über der Schwelle x liegt.

Zur Detektion der Präambel muss man in der Praxis einen Schwellwert wählen. Nur Werte oberhalb dieser Schwelle werden im Maximalwert-Detektor berücksichtigt. Mit diesem Funktionenplot kann man abschätzen, wie wahrscheinlich falsche Zeitpunkte detektiert

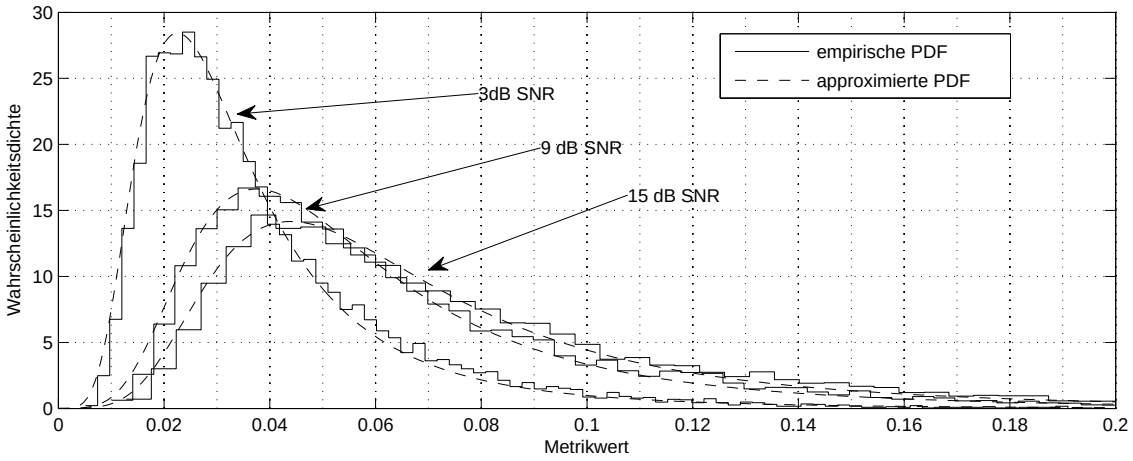


Abbildung 4.17: Empirisch bestimmte PDF der Maximalwerte an den ausgesuchten Stellen

werden, und mit welcher Wahrscheinlichkeit richtige Zeitpunkte nicht detektiert werden. Wählt man z.B. den Schnittpunkt der beiden Verteilungen in Abbildung 4.16 für 9 dB, der abgelesen bei circa 0.023 liegt, als Schwellwert. Dann liest man in Abbildung 4.18 ab, dass mit $P(w_1 > 0.023) \approx 0.92$ ungefähr 92% der Metrik-Extrema oberhalb der Grenze liegen, während mit $P(w_2 \leq 0.023) \approx 0.98$ circa 98% der Werte zu falschen Zeitpunkten darunter liegen (für 9 dB). Der optimale Schwellwert hängt vom Empfängerdesign ab. Ist der Empfänger z.B. robust gegen Fehlentscheidungen, könnte man die Schwelle sehr niedrig wählen. Dadurch werden mehr falsche Zeitpunkte detektiert, dafür aber weniger Rahmen verpasst.

Insgesamt lässt sich festhalten, dass der Schwellwert vom niedrigsten SNR, das der Empfänger unterstützen soll, abhängt. Er hängt aber auch von dem zu erwartenden Mehrwegekanal ab. Dieser Algorithmus kommt beim OFDM-System des Lehrstuhls zum Einsatz. Dort wurde der Schwellwert empirisch bestimmt und liegt bei 0.1.

Die Häufigkeitsverteilung der geschätzten Zeitpunkte ist in Abbildung 4.19 dargestellt. Die Grafik zeigt die relative Häufigkeit der Realisationen von w_3. Gemessen wird der Abstand der Schätzwerte zum idealen Zeitpunkt (hier an der Stelle Null angenommen). Auffällig ist, dass die Häufigkeiten ähnlich dem Vektor der Erwartungswerte der Pfad-Amplituden ist (siehe Definition der Simulationsparameter Gleichung 4.13). Diese kann man auch so umdeuten, dass sie die Wahrscheinlichkeit bestimmen, dass ein Pfad die maximale Amplitude unter allen Pfaden besitzt. Umso höher der erwartete Pfadgewinn, umso höher eben auch die Wahrscheinlichkeit für hohe momentane Pfadgewinne. Durch die Maximumsselektion kommt dadurch der Pfad mit dem höchsten erwarteten Gewinn am häufigsten vor.

Ist der Schwellwert optimal gewählt, so wird der Maximumsdetektor mit hoher Wahrscheinlichkeit nur Zeitpunkte finden, die durch die Verteilung der Verzögerung der Signalpfade vorgegeben sind. Ist dann auch das zyklische Präfix so entworfen worden, dass ein Pfad mit der maximal zu berücksichtigenden Verzögerung dennoch innerhalb des Schutzinter-

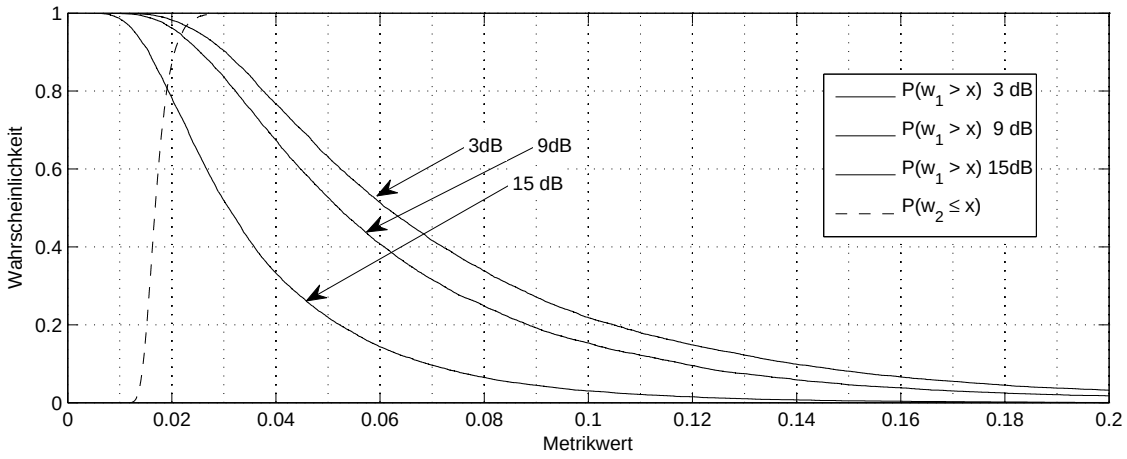


Abbildung 4.18: Empirisch bestimmte (inv.) Verteilungsfunktionen

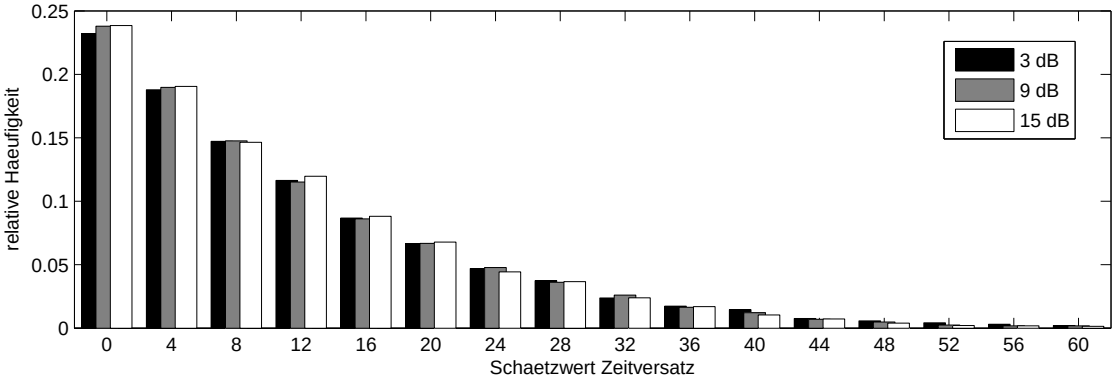


Abbildung 4.19: Empirische Häufigkeitsverteilung der Schätzwerte

valls ankommt, sind somit mit hoher Wahrscheinlichkeit alle detektierten Zeitpunkte im gleichen Intervall. Allerdings wird immer der dominante Pfad erkannt, welcher unter der Annahme eines Rayleigh-Fading-Kanals nicht unbedingt der erste Pfad sein muss. Idealer wäre nämlich eigentlich die Erkennung eines Zeitpunkts im Intervall des zyklischen Präfix, welches nicht durch Interferenzen von vorhergehenden Signalen gestört ist. Es gibt Ansätze um auch diesem Problem beizukommen (z.B. [SDCL06]). Diese sind nicht Teil dieser Arbeit.

4.2.3 Einfluss eines Abtastratenversatz

Bei der Simulation eines Abtastratenversatzes wurde besonderes Augenmerk auf die Abhängigkeit der Metrikwerte von der Phasenverschiebung gelegt. Durch die Interpolation fallen die Abtastzeitpunkte am Empfänger nicht exakt auf die Werte des Senders. Die

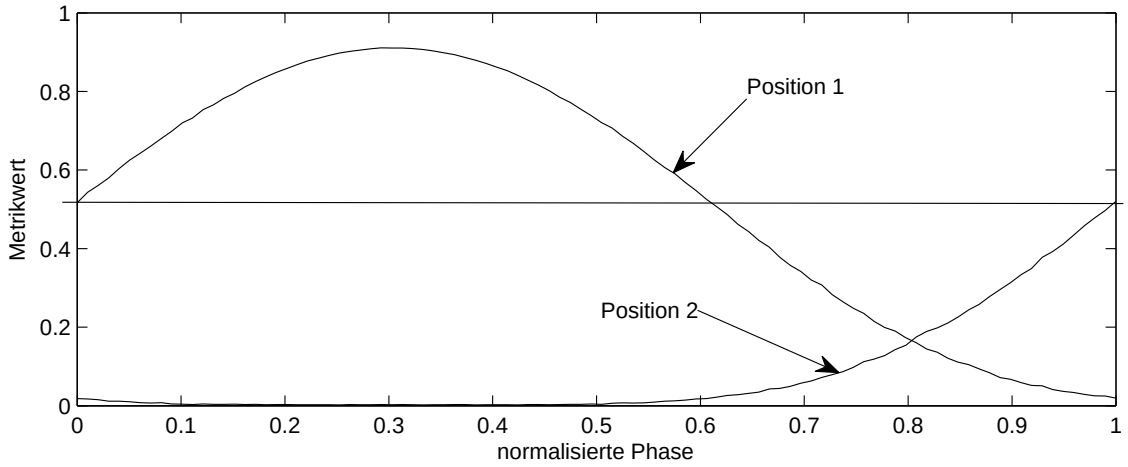


Abbildung 4.20: Metrikwerte in Abhängigkeit der Phase bei konstantem Abtastratenversatz

Phasenverschiebung bestimmt die initiale Verschiebung des ersten Abtastwertes. Sie liegt im Intervall $[0, 1) \subset \mathbb{R}$. Eine Phasenverschiebung von Null bedeutet, dass der erste empfangene Abtastwert exakt auf den gleichen Zeitpunkt fällt wie der gesendete Wert (unter Vernachlässigung jeglicher anderer Verzögerungen), während der Wert 1 dazu führt, dass der erste Wert um einen Abtastzeitpunkt verschoben ist.

Bei der Simulation wird der Versatz der Abtastrate konstant gehalten. Er beträgt 10 KHz bei einer Bandbreite von 25 Mhz (entsprechend 400 ppm relativer Versatz). Die FFT-Länge betrug 1024 Werte und das zyklische Präfix umfasste 102 Werte. Dies sind die gleichen Simulationsparameter wie für die erste Simulation des Schmidl & Cox-Algorithmus. Nun wurde aber keine Mehrwegeausbreitung miteinbezogen. Statt dessen existiert nur ein Rayleighpfad mit dem erwarteten Pfadgewinn 1. Das SNR ist konstant auf 20 dB eingestellt. Verändert wurde die Phasenverschiebung im Intervall $[0, 1)$, wobei diese in 100 Schritten mit konstanter Schrittweite erhöht wurde. Gemessen wurden zwei Metrikwerte. Und zwar erstens der Wert am idealen Zeitpunkt und zweitens der Wert genau einen Abtastzeitpunkt später. Für jede Einstellung wurden wiederum 10.000 Simulationen durchlaufen und der Mittelwert der ermittelten Messungen bestimmt. Das Ergebnis ist in Abbildung 4.20 aufgetragen. Es zeigt die Abhängigkeit der Metrikwerte an beiden Position von der Phasenverschiebung. Dabei entspricht Position 1 dem idealen Abtastzeitpunkt.

Deutlich erkennbar ist die starke Abhängigkeit des Metrikwerts von der Phase. Zudem zeigt sich, dass bei Phasenwerten gegen 1 der Maximalwert sich auf Position 2 verschiebt. Folgende Vermutung wird aufgestellt. Die Metrikberechnung ist im Prinzip ein Filter. Dieser wird an Position 1 und Position 2 ausgewertet, also nur an diskreten Positionen. Fällt allerdings das tatsächliche Signal zwischen die beiden Positionen, so verteilt sich die „Energie“ auf die gefilterten Werte beider Positionen. Strebt die Phasenverschiebung gegen 1, also entspricht sie annähernd dem Abstand der Abtastwerte, hat sich das Maximum um einen Zeitpunkt verschoben.

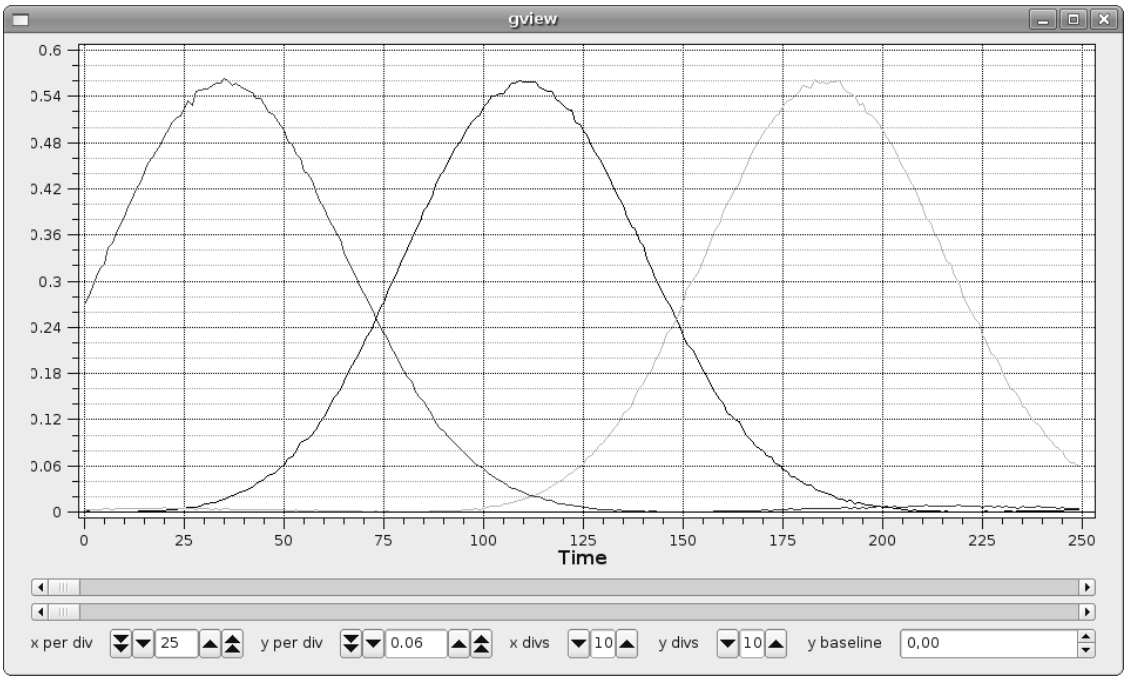


Abbildung 4.21: Aufzeichnung vom realen Kanal

Bei der Übertragung mit der vorhandenen Funkhardware des Lehrstuhls ist genau dieser Effekt beobachtbar. Selbst bei geringer Bandbreite oszilliert der Maximalwert. Bei der Festlegung der Schwelle muss man auch diese Effekte berücksichtigen. Im konkreten Fall fiel das Problem auf, weil der Schwellwert zu hoch gewählt wurde. Dadurch wurden periodisch Rahmen verpasst und die Fehlerrate war entsprechend hoch. Der Plot in Abbildung 4.21 entstand von der Ausgabe der Synchronisierungsstufe, in GNU Radio implementiert, angewendet auf die bereits vorher verwendete Übertragung (siehe GNU Radio-Implementierungen der Algorithmen). Dabei wurde die erste Präambel gesucht (manuell) und anschließend in äquidistanten Abständen ausgehend von diesem Maximalwert der Metrik die weiteren Werte gesammelt. Die erste Kurve zeigt also alle Werte, die im Abstand $n \cdot (N + N_g)$ zum Maximalwert der ersten erkannten Präambel liegen. Die zweite Kurve zeigt alle genau einen Abtastwert vor den Werten der ersten Kurve, und analog zeigt die dritte Kurve Werte mit einer Verschiebung von -2 zur ersten Kurve. Der Maximalwert wandert langsam „nach links“ im Bild. Folglich ist die Abtastfrequenz der Empfangshardware etwas zu niedrig eingestellt.

5 Frequenzsynchronisierung

Bevor das zeitsynchronisierte Signal mit der FFT weiter bearbeitet wird, stellt man die Übereinstimmung der FFT-Träger mit den Trägern des empfangenen Signals sicher. Bei existenter Frequenzverschiebung

$$\phi = \theta + \eta \tag{5.1}$$

(siehe Gleichung 2.1), ist gerade η mit

$$-0.5 \leq \eta < 0.5, \eta \in \mathbb{R}$$

der zuerst zu korrigierende Anteil.

Alle Kanaleffekte wie Rauschen und Mehrwegeausbreitung werden zuerst einmal vernachlässigt. Lediglich eine Frequenzverschiebung um ϕ sei vorhanden.

Ist $s(k)$ das gesendete Signal, dann empfängt man

$$r(k) = s(k) \exp(j2\pi\phi k/N). \tag{5.2}$$

Insbesondere ist der nach der Zeitsynchronisation gewonnene Block in Anlehnung an Gleichung 1.2

$$r_i(k) = r(k_{opt,i} + k) \quad \text{für } 0 \leq k < N - 1. \tag{5.3}$$

Hier bezeichnet $k_{opt,i}$ den durch die vorherige Synchronisierungsstufe geschätzten Zeitpunkt des Beginns des i-ten Blocks. Der Wert kann direkt aus d_{opt} berechnet werden.

Nimmt man eine optimale Zeitsynchronisierung an, gilt

$$\begin{aligned} r_i(k) &= s_i(k) \exp(j2\pi\phi k/N) && (5.4)\\ &= \left(\sum_{n=0}^{N-1} c_{i,n} \exp(j2\pi kn/N) \right) \exp(j2\pi\phi k/N) && (5.5)\\ &= \sum_{n=0}^{N-1} c_{i,n} \exp(j2\pi k(n+\phi)/N). && (5.6) \end{aligned}$$

5.1 Schmidl & Cox

Die Korrektur von η bezeichnet man als grobe Synchronisierung, die von θ als Feinsynchronisierung. In [SC97] wird ein Algorithmus vorgestellt, der ausgehend von derselben Präambel, die bereits für die Zeitsynchronisierung benutzt wird, die grobe Frequenzverschiebung schätzt. Die Verschiebung ϕ wird jedoch anders aufgeteilt, und zwar in

$$\phi = \nu + 2g, \quad g \in \mathbb{Z}, \nu \in [-1, 1) \subset \mathbb{R}. \tag{5.7}$$

Der zweite Schätzer im Artikel, für die feine Synchronisierung, benötigt eine zweite Präambel und benutzt dann beide bekannten Blöcke.

5.1.1 Grobe Synchronisierung

Bei Berechnung der Metrik benutzt man die Gleichung 4.5. Mit 5.3 ergibt sich

$$P(d_{opt}) = \sum_{n=0}^{N/2-1} r_p^*(n) r_p(n + N/2) \tag{5.8}$$

wobei p der Index der verwendeten Präambel ist. Setzt man Gleichung 5.4 für $r_p(k)$ ein führt dies zu

$$P(d_{opt}) = \sum_{n=0}^{N/2-1} s_p^*(n) s_p(n + N/2) \exp(-j2\pi\phi n/N) \exp(j2\pi\phi(n + N/2)/N) \tag{5.9}$$

$$= \sum_{n=0}^{N/2-1} s_p^*(n) s_p(n + N/2) \exp(j\pi\phi) \tag{5.10}$$

und zusammen mit der Eigenschaft 4.1 der Präambel folgt

$$P(d_{opt}) = \sum_{n=0}^{N/2-1} |x(n)|^2 \exp(j\pi\phi) \tag{5.11}$$

$$= \exp(j\pi\phi) \left(\sum_{n=0}^{N/2-1} |x(n)|^2 \right). \tag{5.12}$$

Dieser Sachverhalt ist in Abbildung 5.1.1 dargestellt. Unter diesen Bedingungen kann man

$$\hat{\nu} = \frac{arg(P(d_{opt}))}{\pi} \tag{5.13}$$

schätzen, denn die Summe in 5.12 ist auf jeden Fall reell und die Phase des Zahlenwerts ist proportional zur Frequenzverschiebung ϕ. Wegen der Mehrdeutigkeit der Argument-Funktion ergeben sich Schätzwerte im Bereich $-1 \leq \hat{\nu} < 1$. Die verbleibende Mehrdeutigkeit in

$$\phi = \nu + 2g \tag{5.14}$$

wird in der nächsten Stufe aufgelöst.

5.1.2 Feinsynchronisierung

Nach dem $\hat{\nu}$ geschätzt und korrigiert wurde, bleibt aus Gleichung 5.7 nur noch $g \in \mathbb{Z}$ zu schätzen. Die weitere Verarbeitung findet im Frequenzbereich statt. Der verbleibende Versatz ist eine zyklische Verschiebung der Unterträger.

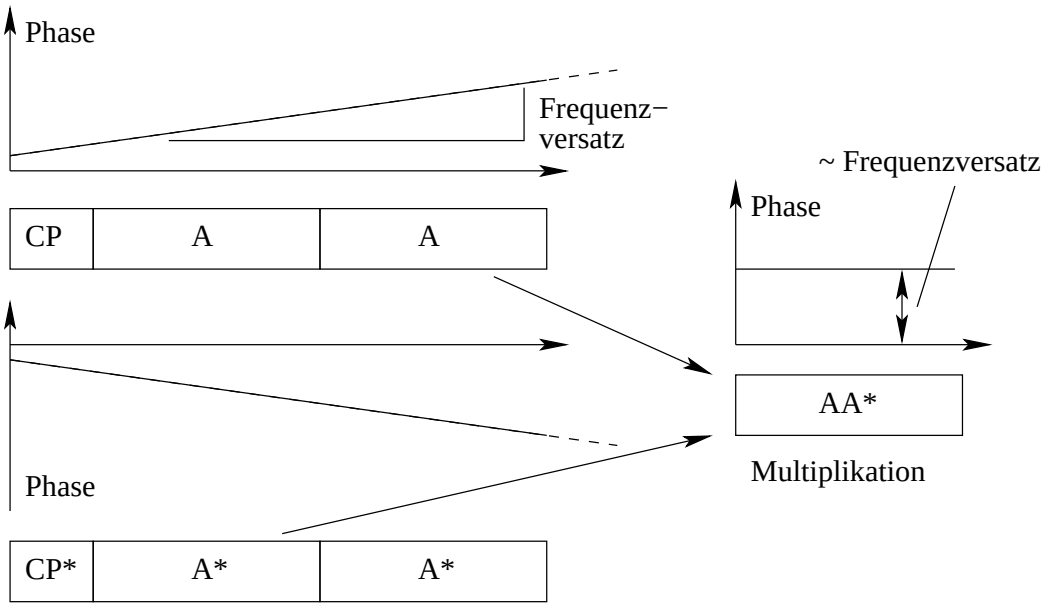


Abbildung 5.1: Phaseninkrement durch Frequenzversatz

Es wird eine neue Präambel eingeführt, die zusätzlich zur ersten Präambel in den Sendesignalstrom eingefügt wird. Die Präambel wird im Frequenzbereich definiert. Sie hat im Zeitbereich keine besondere Eigenschaft. Mit p_2 als Index der zweiten Präambel und der bijektiven Abbildung

$$I(k) : [0, N_s) \rightarrow \mathcal{I}, \tag{5.15}$$

für die

$$I_{gz}(k) = I(2k) \quad \forall\, k = 0\,..\,N_s/2 - 1 \tag{5.16}$$

gilt ($I_{gz}(k)$ definiert in 4.3), wählt man

$$c_{p_2,k} = \begin{cases} X_{2,n} \in \mathbb{C}, & k = I(n), \quad \forall\, n = 0\,..\,N_s - 1 \\ 0 & k \notin \mathcal{I} \end{cases} \tag{5.17}$$

Die Modulation der Unterträger ist nicht relevant. Wiederum schlagen die Autoren den QPSK-Signalraum als Quelle vor. Alle Unterträger werden besetzt.

Für den folgenden Algorithmus braucht man aber nur die Symbole auf den Unterträgern mit geradzahligem Index. Die anderen Unterträger können z.B. für die Kanalschätzung verwendet werden. Mit 4.4 und 5.17 wird die neue Sequenz

$$v_n = \sqrt{2}\frac{X_{2,2n}}{X_{1,n}} \quad \forall\, n = 0\,..\,N_s/2 - 1 \tag{5.18}$$

definiert. Sei nun

$$R_{i,n} = \sum_{k=0}^{N-1} r_i(k) \exp(-j2\pi k/N) \tag{5.19}$$

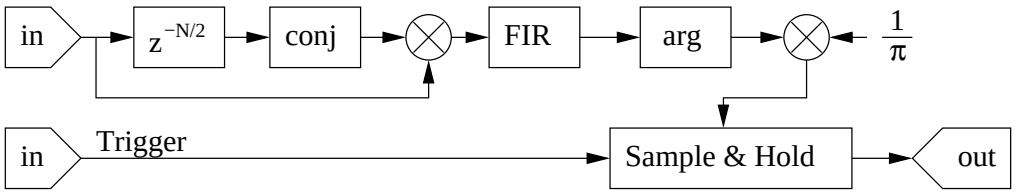


Abbildung 5.2: GR-Implementierung, Alternative 1

die Werte der DFT angewendet auf die durch die Zeitsynchronisierung gewonnenen OFDM-Blöcke aus Gleichung 5.3. Die Schätzung für g kann gefunden werden, indem das $\hat{g}$ gesucht, das die Funktion

$$B(g) = \frac{|\sum_{n\in\mathcal{I}_e} R^*_{p_1,n+2g} R_{p_2,n+2g} v^*_n|^2}{2\left(\sum_{n\in\mathcal{I}_e} |X_{p_2,n}|^2\right)} \tag{5.20}$$

maximiert. Die komplette Schätzung des Frequenzversatz ergibt sich dann zu

$$\hat{\phi} = \hat{\nu} + 2\hat{g}. \tag{5.21}$$

5.1.3 GNU Radio-Implementierungen

Nun folgen einige Ideen, wie man den groben Schätzer nach Schmidl & Cox als GNU Radio-Block implementieren kann. Die Zeitsynchronisierung hat schon statt gefunden. Sie hat einen Triggerimpuls erzeugt, der die Position der Präambel markiert. Den Triggerstrom kann man sich so vorstellen, dass es für jeden Abtastwert einen Triggerwert gibt, der entweder Null ist, wenn dieser Zeitpunkt kein Triggerpunkt ist, oder eben Eins, wenn der zugehörige Abtastwert eine besondere Position markiert, z.B. den Anfang der Präambel. Allen alternativen Implementierungen ist gemein, dass sie den empfangenen Datenstrom und den Triggerstrom als Eingabe brauchen.

Zuerst zum einfachsten Fall, dargestellt in Abbildung 5.2. Die Metrikfunktion $P(d)$ wird berechnet wie bereits bei der Implementierung der Zeitsynchronisierung. Die ersten Blöcke der Kette können ohne Probleme mit dieser Stufe geteilt werden. Nach dem FIR-Filter kommt dann die Argument-Funktion und die Normierung hinzu. Der berechnete Schätzwert wird von dem Sample&Hold-Glied zwischengespeichert ("Sample") falls ein Triggerimpuls vorhanden ist. Ist kein Impuls vorhanden, wird der zwischengespeicherte Wert ausgegeben ("Hold"). Dahinter kann mit dem Ausgangssignal z.B. ein Oszillator gesteuert werden, der ein Signal mit einer Frequenz proportional zu diesem Wert generiert. Mischt man dieses Signal mit dem empfangenen Signalstrom, korrigiert dies den groben Frequenzversatz.

Dieser spezielle Aufbau hat den Vorteil, dass an dieser Stelle nicht bekannt sein muss, wieviele Blöcke nach der Präambel noch kommen, also wie oft er seinen Schätzwert wiederholen muss. Dafür werden aber ständig Berechnungen durchgeführt und möglicherweise unnötig viele Ausgabewerte produziert, die ungenutzt verworfen werden.

Um diese möglichweise (aber nicht zwangsweise) überflüssigen Berechnungen zu vermeiden, kann man den Abtastblock weiter nach vorne in der Verarbeitungskette schieben.

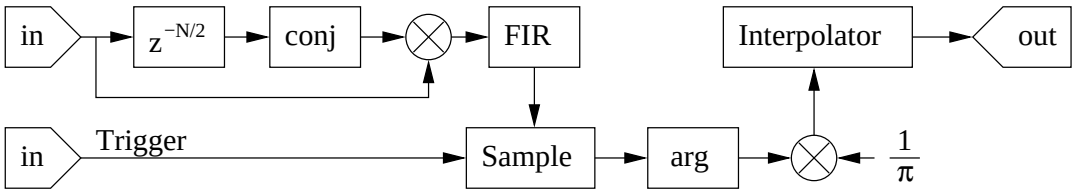


Abbildung 5.3: GR-Implementierung, Alternative 2

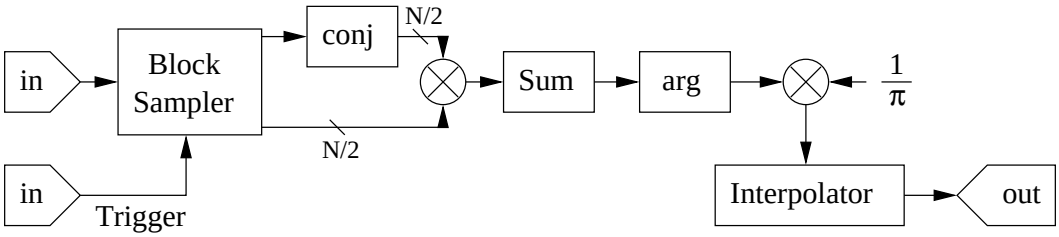


Abbildung 5.4: GR-Implementierung, Alternative 3

Abbildung 5.3 stellt eine Implementierung dar. Der Sample&Hold-Block wird in seine zwei Funktionen, "Sample" und "Hold", geteilt. Die Extraktion aus dem Datenstrom ("Sample") erfolgt nun für jeden Triggerimpuls einmalig. Nun wird auch die Argument-Funktion und die Normalisierung nur einmal pro extrahiertem Wert ausgeführt. Findet die Zeitsynchronisierung keine Präambel, finden folglich auch keine Berechnungen statt.

Dafür wird der "Hold"-Teil komplizierter. An seine Stelle tritt ein Interpolator, der jedoch genau wissen muss, wieviele Werte er pro Schätzwert erzeugt. Bei statischen Rahmen- und Blocklängen ist dies gar kein Problem. Sollte aber z.B. die Rahmenlänge variabel sein, so muss der Interpolator ersetzt werden. Die Information über die Anzahl der auszugebenden Kopien seines Zwischenspeichers braucht er nun als Eingabe(-strom). Um die Blöcke im Empfänger synchron zu halten, bedarf es bei GNU Radio einer gewissen Anstrengung und einem sorgfältigen Design, da keine inhärente Synchronisierung durch die verstrichene Zeit gegeben ist. Denn keinem Block ist "Zeit" ein Begriff. Ein Lösungsansatz wäre, für jeden zu generierenden Ausgangswert einen Eingangswert von einem intelligenten Block zu erzeugen. Durch die 1:1 Übersetzung können die Datenströme dann nicht auseinanderlaufen. Allerdings bedarf es eines zentralen Blocks, der über das entsprechende Wissen verfügt und an alle Blöcke, die diese Informationen benötigen, angeschlossen ist.

Der letzte Implementierungsvorschlag (Abb. 5.4) ist günstig für den Fall, dass Abschnitte der Berechnung nicht mit anderen Stufen gemeinsam genutzt werden können. Z.B. wenn die Zeitsynchronisierung auf einem anderen Algorithmus fußt, aber auch, wenn die Berechnung des Schätzwertes mit höherer Rechengenauigkeit statt finden soll.

Die Abtastvorrichtung ist nun am Eingang des Blocks angekommen, während der Inter-

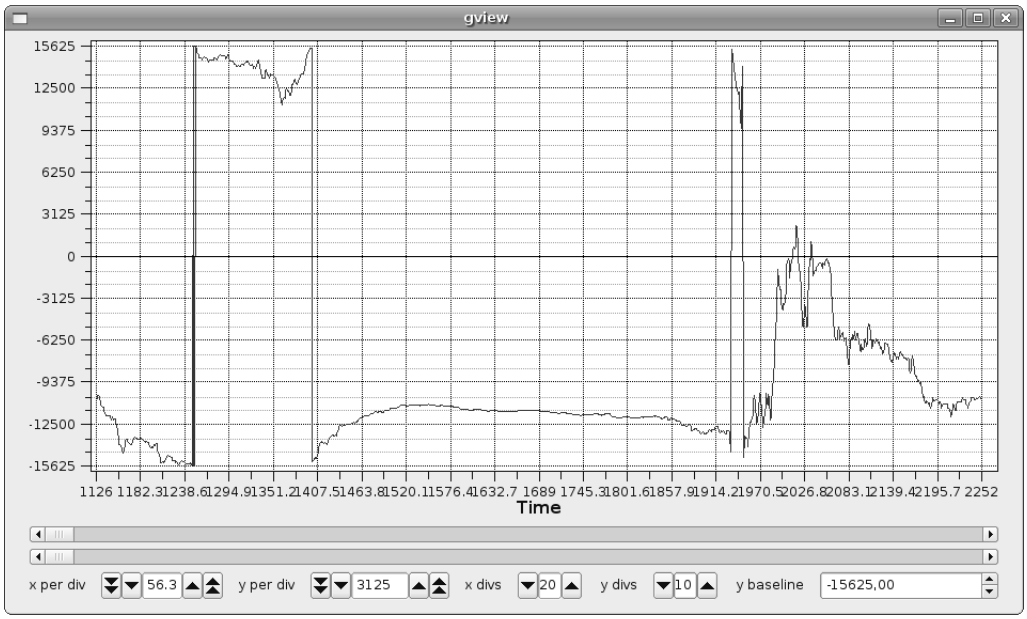


Abbildung 5.5: Bildschirmfoto Ausgabe des Algorithmus als GNU Radio-Block

polator weiterhin am Ausgang ist. Jegliche Berechnungen werden nun nur noch bei einem Triggerimpuls, also bei Detektion der Präambel, ausgeführt. Dazu wird die komplette Präambel als Block aus dem Datenstrom entnommen. Diesen Block teilt man dann in zwei gleich große Hälften. Die erste Hälfte wird elementweise komplex konjugiert. Die Blöcke als Vektoren interpretiert, folgt dann die Bildung des Skalarprodukts. Der weitere Teil der Verarbeitung des Skalarprodukts, insbesondere auch der Nachteil des Interpolators, ist identisch zum vorhergehenden Implementierungsvorschlag.

Die Ausgabe der GNU Radio-Implementierung ist in Abbildung 5.5 auf dem Bildschirmfoto dargestellt. Dabei wurde der Ausgabewert mit der Unterträgerbandbreite multipliziert, sodass die Y-Skala der Einheit Hertz entspricht. In der Mitte des Bildes findet sich der ideale Abtastzeitpunkt (Ende des Plateaus). Wie bereits bei den vorherigen Bildschirmfotos entstand diese Ausgabe als Verarbeitung einer Übertragung über die zur Verfügung stehende Funkhardware. Bei einer FFT-Länge von 512 Werte und 51 Werte als Schutzintervall wurden abwechselnd die Präambel für den Schmidl&Cox-Algorithmus und die für den nächsten Algorithmus gesendet, wobei zwischen den Präambel ein Block als Abstandshalter eingefügt wurde. Die Bandbreite betrug wiederum 8 MHz und die Trägerfrequenz lag bei 2.46 GHz. Der Algorithmus schätzt den Frequenzversatz grob auf -11.5 KHz ein.

5.2 Morelli & Mengali

Der Algorithmus in [SC97] zur groben Schätzung des Frequenzversatz arbeitet nur im Bereich $[-1, 1)$ ohne Mehrdeutigkeiten. Um diesen Bereich zu vergrößern, schlagen die

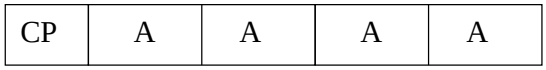


Abbildung 5.6: Präambel im Zeitbereich, L = 4

Autoren Morrelli und Mengali in ihrem Artikel [MM99] vor, die Anzahl der identischen Blöcke innerhalb der Präambel zu erhöhen. Sie zeigen, dass deren Anzahl L unmittelbar den Bereich $[-L/2, L/2)$ des Schätzers bestimmt.

Wird die Anzahl der Blöcke sorgfältig gewählt, so dass der maximal zu erwartende Frequenzversatz innerhalb des Schätzbereichs liegt, also ohne Mehrdeutigkeiten aufgelöst wird, spart man dadurch unmittelbar die zweite Präambel ein. Dies geht auf Kosten der Komplexität der Berechnungen. Hinzu kommt aber, dass der vorgeschlagene Algorithmus ausserdem näher an Cramer-Rao-Grenze, die die maximal zu erreichende Genauigkeit angibt, als der S&C-Algorithmus liegt.

Die neue Präambel besteht aus L identischen Blöcken der Länge N/L. Es gilt

$$s(k) = s(k + mN/L) \quad \forall k = 0\,..\,N/L - 1,\, m = 1\,..\,L - 1 \tag{5.22}$$

bzw. die Präambel hat die Form

$$s_{p_1} = [\, A_{N/L} \quad A_{N/L} \quad .. \quad A_{N/L}\,]. \tag{5.23}$$

In Anlehnung an [MM99] und [MMP07] definiert man die Autokorrelation

$$R(m) = \sum_{k=mL}^{N-1} r^*(k - mN/L)r(k) \quad m = 0\,..\,L/2 \tag{5.24}$$

und die Gewichte

$$w(m) = \frac{12(L-m)(L-m+1) - 3L^2}{2L(L^2-1)}. \tag{5.25}$$

Die grobe Schätzung des Frequenzversatz ergibt sich zu

$$\hat{\phi} = \frac{1}{2\pi/L} \sum_{m=1}^{L/2} w(m) \arg\{R(m)R^*(m-1)\}. \tag{5.26}$$

Die Funktionen $R(m)$ und $w(m)$ sind beispielhaft für $L = 4$ in Abbildung 5.7 dargestellt. $R(m)$ summiert über die elementweise multiplizierten Ausschnitte. Der Winkel von $R(1)$ wird mit $w(1)$ gewichtet. Die Differenz der Winkel von $R(2)$ und $R(1)$ wird mit $w(2)$ gewichtet. Die Schätzung $\hat{\phi}$ entspricht der Summe der gewichteten Winkeldifferenzen und liegt im Intervall $\hat{\phi} \in [-2, +2)$.

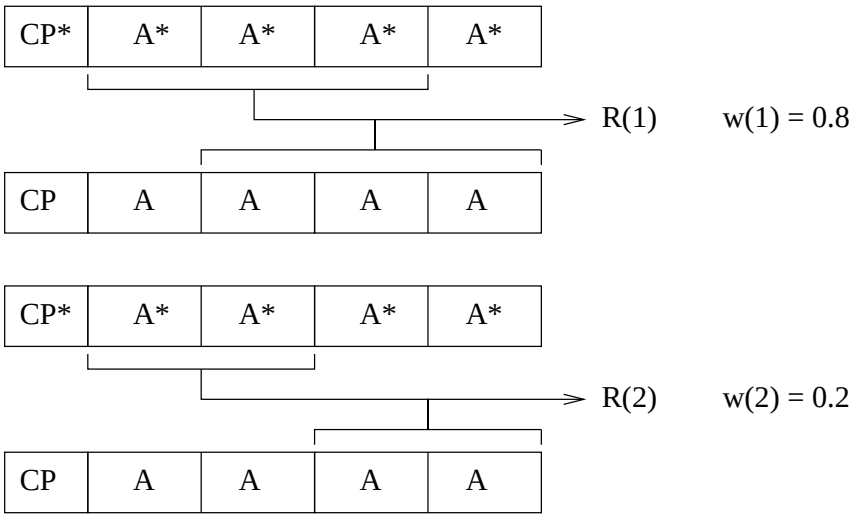


Abbildung 5.7: Berechnung der Zwischenergebnisse, L = 4

5.2.1 GNU Radio-Implementierung

Den erweiterten Schätzbereich dieses Algorithmus erkaufen wir uns mit einer erhöhten Komplexität der Implementierung. Dafür können wir die zweite Stufe, die die Mehrdeutigkeiten auflöst, sparen falls wir den Bereich für unser System groß genug gewählt habe. Eine Implementierung für $L = 4$ ist in Abbildung 5.8 dargestellt. Die „Block Sampler“ entnehmen die vier Blöcke entsprechend der Abbildung 5.7. Die jeweils linken Blöcke werden komplex konjugiert und danach die Vektoren gleicher Länge elementweise multipliziert. Die Summation über diese Produkte führt zu den Zwischenergebnissen

$$R(1) = \sum_{k=L}^{N-1} r^*(k - N/L)r(k)$$

und

$$R(2) = \sum_{k=2L}^{N-1} r^*(k - 2N/L)r(k).$$

Dabei ist $R(0)$ immer rein reell, deswegen vereinfacht sich der erste Summenterm in Gleichung 5.26 zu

$$\arg\{R(1)R^*(0)\} = \arg\{R(1)\}.$$

Der erste Summenterm ist also einfach nur das Argument von $R(1)$. Der zweite Summenterm berechnet sich dann aus dem Argument des Produkts von $R(2)$ und dem konjugierten $R(1)$. Alternativ könnten wir auch die Argumente beider Zwischenprodukte berechnen und subtrahieren, solange wir das Ergebnis modulo 2π reduzieren. Die beiden Argumente werden nun noch mit den konstanten Faktoren gewichtet und anschließend aufsummiert. Um

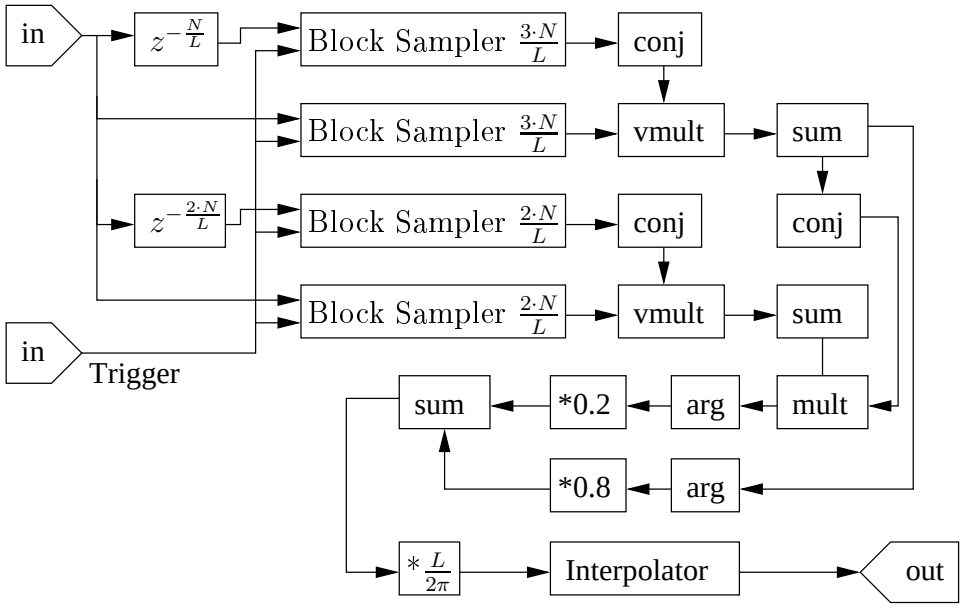


Abbildung 5.8: Mögliche Implementierung als GNU Radio-Block, L = 4

den Schätzwert als Vielfaches des Unterträgerabstandes zu erhalten, wird diese Summe mit dem konstanten Wert $\frac{L}{2\pi}$ normiert. Das Ergebnis muss, wie bereits bei den Betrachtungen zur Implementierung des vorherigen Algorithmus, gegebenenfalls interpoliert werden. Diesbezüglich gelten die selben Überlegungen.
Auch die GNU Radio-Implementierung dieses Algorithmus wurde auf dem realen Funkkanal getestet. Wie bereits beim vorherigen Algorithmus ist die Ausgabe als Bildschirmfoto vorhanden (Abb. 5.9). Die Y-Skala trägt dieselbe Einheit (Hertz) wie in Abbildung 5.5. In der Mitte des Bildes findet sich der ideale Abtastzeitpunkt (Ende des Plateaus). Es ist dieselbe Übertragung, nur dass der neue Algorithmus die Verarbeitung übernimmt. Der Frequenzversatz wird nun auf ungefähr 19.75 KHz geschätzt. Berücksichtigen wir, dass der S&C-Schätzer eine Mehrdeutigkeit $\hat{g}$ mit

$$\hat{\phi} = \hat{\nu} + 2\hat{g}, \quad \hat{g} \in \mathbb{Z} \tag{5.27}$$

besitzt, sind die Ergebnisse der beiden Schätzer (zur Erinnerung: 1. Schätzung lag bei -11.5 KHz) annähernd gleich (für $\hat{g} = 1$):

$$\text{-11.5 KHz} + \frac{2}{512}\ 8\ \text{MHz} \approx 19.75\ \text{KHz}.$$

Der reale Versatz liegt vermutlich bei ungefähr 20 KHz, war damit ausserhalb des Schätz-Bereichs des groben S&C-Schätzers und führte bei dessen Ergebnis zur Reduktion um zwei Unterträgerabstände.

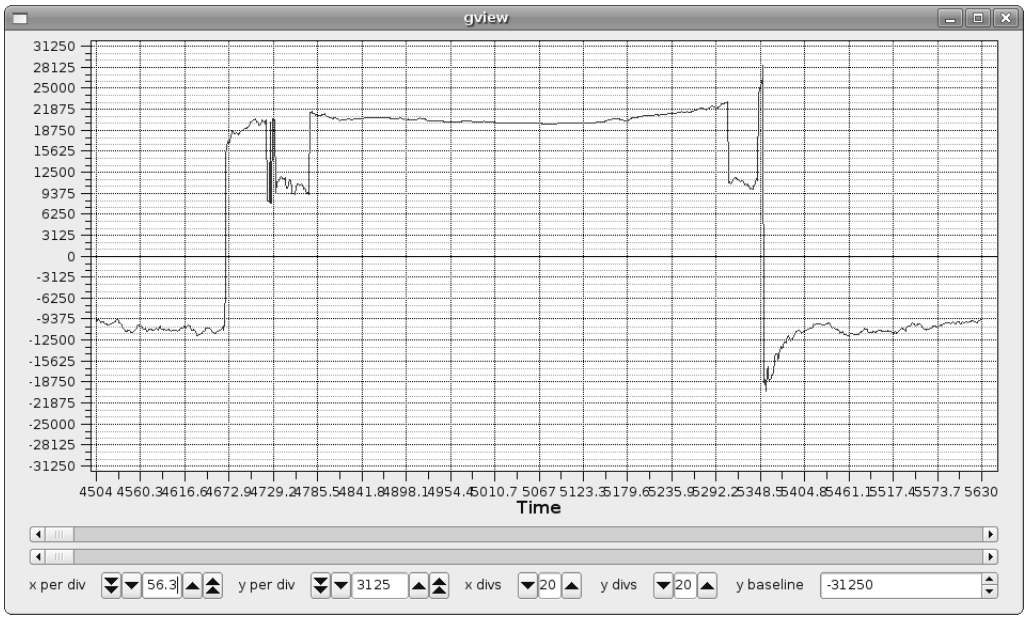


Abbildung 5.9: Bildschirmfoto Ausgabe des Algorithmus als GNU Radio-Block

6 Zusammenfassung und Fazit

In dieser Studienarbeit wurde die OFDM-Technologie betrachtet. Zuerst haben wir gesehen, wie ein OFDM-Signal aufgebaut ist, wie wir es mathematische beschreiben können und wie wir es es in der Praxis einfach durch Anwendung der FFT erzeugen können. Dann haben wir das Signalmodell des Kanals, das Basis der Algorithmen in dieser Arbeit ist, vorgestellt. Wir haben verschiedene Kanaleinflüße identifiziert, die wir unabhängig von einander beschrieben haben. Das Standardkanalmodell beinhaltet Zeitversatz, Frequenzversatz, Mehrwegeausbreitung des Signals sowie weißes Gaußsches Rauschen. Als zusätzlichen Modelleinfluß wurde ein Versatz der Abtastraten erwähnt.

Im nächsten Kapitel wurde GNU Radio vorgestellt. Es ist ein modulares Software-Framework für Software Defined Radio, das durch seine enge Anbindung an die USRP[1]-Plattform Basis für SDR-Prototypen ist.

Anschließende haben wir die Aufgaben der Zeit- und Frequenzsynchronisierungsstufen für OFDM-Empfänger beschrieben. Sie sind essentieller Bestandteil eines Empfängerentwurfs, werden in der gängigen Literatur aber meist als gegeben und perfekt angenommen. Der erste vorgestellte Algorithmus zur Zeitsynchronisierung stammt von Schmidl & Cox aus dem Jahre 1997. Wir definierten eine Präambel mit zwei identischen Hälften und eine Metrikfunktion, die im Empfangssignal nach der spezifischen Struktur der Präambel sucht. Nehmen wir einen linearen quasistationären Kanal an, so werden beide Präambelhälften durch den Kanal identisch gestört. Bei dieser Beobachtung vernachlässigten wir das Rauschen und nahmen an, dass das zyklische Präfix entsprechend seiner Funktion lang genug gewählt wurde.

Bei diesem Algorithmus treffen wir auf Mehrdeutigkeiten in der Metrik, deren Ursache das zyklische Präfix ist. Durch den nicht durch Interblockinterferenz gestörten Bereich des Präfix ensteht ein Plateau in der Metrik, das eine erhöhte Varianz des Schätzers zur Folge hat. In [GRZ05] wurde der S&C-Algorithmus angepasst um diese unerwünschte Mehrdeutigkeit zu verhindern und damit die Varianz des Schätzers zu verbessern. Dazu wird die ursprüngliche Präambel zusätzlich mit einer Pseudorauschsequenz moduliert. Der Empfänger berücksichtig in der ebenfalls modifizierten Metrikfunktion die Korrelation mit dieser neuen Sequenz. Die Metrikfunktion nimmt Impulsform an.

Beide Algorithmen wurden mit Hilfe von Simulationen analysiert. Dazu haben wir zuerst die Ergebnisse aus [SC97] nachvollzogen. Für den S&C-Algorithmen verwendeten wir außerdem in einem zweiten Durchlauf eine eigene Parameterwahl, die der Wahl für die Experimentalplattform des Lehrstuhls entsprach. Als Erweiterungs des Kanalmodells berücksichtigten wir in einigen Simulationen einen Versatz der Abtastrate und analsierten

[1] Universal Software Radio Peripheral

den Einfluß auf die Schätzer für die Zeitsynchronisierung. Es stellte sich heraus, dass der erste Algorithmus sehr robust ist, während die Modifikation durch den zweiten bei Berücksichtigung dieses Kanalparameters zu einer starken Verschlechterung der Ergebnisse führt.

Im nächsten Kapitel betrachteten wir zwei Algorithmen zur Frequenzsynchronisierung. Der erste vorgestellte stammt aus [SC97] und ist in zwei Stufen aufgeteilt. Vor der FFT-Operation schätzt er den gebrochenen Anteil des Frequenzversatzes, kann aber Mehrdeutigkeiten als Vielfache des zweifachen Unterträgerabstandes nicht auflösen. Da die Orthogonalität der Träger wiederhergestellt werden kann, siedelt sich die zweite Stufe dann hinter der FFT-Operation an, um die verbleibende Mehrdeutigkeit aufzulösen. Die erste Stufe verwendet die Präambel, die bereits die Zeitsynchronisierungsstufe aus [SC97] benutzt, während die zweite Stufe eine weitere Präambel einführt.

Morelli und Mengali verfeinerten in [MM99] den Ansatz der ersten Stufe aus [SC97], indem sie die Anzahl der identischen Abschnitte in der Präambel erhöhten. Durch diese Modifikation erweiterten sie das Intervall des Schätzers. Durch einen geeigneten Entwurf lässt sich die zweite Stufe hinter der FFT-Operation einsparen.

Der Zeitsynchronisierungsalgorithmus aus [SC97] findet auf Grund seiner Robustheit und Einfachheit breite Verwendung. Viele OFDM-System verwenden Präambel mit ähnlichem Aufbau und diesen Algorithmus oder Varianten davon. Ein Nachteil ist die hohe Varianz des Schätzers, die durch das zyklische Präfix bedingt ist. In [GRZ05] wird dieser behoben und auf den ersten Blick scheint kein weiterer Nachteil zu entstehen. Dies liegt am häufig verwendeten Kanalmodell. Modellieren wir diesen genauer, indem wir den unvermeidbaren Abtastratenversatz mit einbeziehen, erkennen wir, dass der erste Algorithmus sehr robust bleibt, während die Leistung des zweiten stark abnimmt. Die Erfahrung mit realen Kanälen auf der Experimentalplattform des Lehrstuhls bestätigen diese Einschätzung.

Bei den Frequenzsynchronisierungsalgorithmen hingegen ist festzuhalten, dass das Verfahren aus [MM99] nicht nur ein größeres Schätzintervall als der andere vorgestellte Algorithmus bietet, sondern zudem eine geringere Varianz aufweist. Er liegt im Mittel gut 1 dB näher an der Cramer-Rao-Bound. Sein einziger direkt zu erkennender Nachteil ist die erhöhte Komplexität und damit verbunden der erhöhte Rechenaufwand. Dies wird aber teilweise dadurch kompensiert, dass die zweite Stufe nach dem FFT-Block eingespart werden kann.

Zu GNU Radio lässt sich noch sagen, dass es ein sehr modulares und nach einiger Eingewöhungszeit einfach zu verwendendes Werkzeug für den SDR-Prototypenentwickler darstellt. Als erstes Open-Source-Projekt überhaupt ermöglicht es SDR-Anwendungen auf Standard-PC-Hardware. Es steht aber definitiv am Anfang seiner Entwicklung. Die Software ist sehr flexibel und generisch gehalten. Mangels Adaption an die unterliegende Hardware verschenkt sie viele Ressourcen. Besonders fällt hier auf, dass eine Echtzeitfähigkeit mit Soft oder Hard Deadlines nicht vorhanden ist. Die Einhaltung von Zeitvorgaben der Netzwerkstandards ist nicht vorhersagbar. Möglich sind nur eigene Protokolle, die diesem Stand Rechnung tragen. Es zeichnet sich allerdings ab, das muss hinzugefügt werden, dass sich GNU Radio langsam in die entsprechende Richtung bewegt.

6.1 Ausblick

Mangels einer wirklichen Echtzeitfähigkeit von GNU Radio sind momentan andere Ansätze zu bevorzugen, deren Leistungsfähigkeit und Determinismus ausreichend sind, um die Echtzeitanforderung von Radio-Anwendungen zu erfüllen. Eine Realisierung von kritischen Komponenten auf FPGAs als dedizierte rekonfigurierbare Hardware, zusammen mit Softwarekomponenten, die auf unterster Ebene ohne zwischengeschaltetes Betriebssystem agieren, stellt eine mögliche Lösung dar. Mit dem WARP-Board der Rice Universität existiert diese Variante bereits [MSA06].

Literaturverzeichnis

[Chu72] David C. Chu. Polyphase codes with good periodic correlation properties. *IEEE Transactions on Information Theory, Correspondence*, July 1972.

[Czy99] Andreas Czylwik. Synchronization for systems with antenna diversity. In *Vehicular Technology Conference*, volume 2, September 1999.

[GRZ05] Hui Zhang Guanglian Ren, Yilin Chang and Huining Zhang. Synchronization method based on a new constant envelop preamble for ofdm systems. *IEEE Transactions on Broadcasting*, 51(1), March 2005.

[HIP98] Channel models for hiperlan/2 in different indoor scenarios, 1998. ETSI BRAN 3ERI085B.

[HMB00] M. Zeng H. Minn and V. K. Bhargava. On timing offset estimation for ofdm systems. *IEEE Communications Letters*, 4(7), July 2000.

[JH04] Sean Cai Dazi Feng Yonggang Fang Jason Hou, Jing Wang. Preamble sequence for fast cell search, low computational complexity, and low papr, 2004. `http://wirelessman.org/tge/contrib/C80216e-04_265r1.pdf`.

[MM99] M. Morelli and U. Mengali. An improved frequency offset estimator for ofdm applications. *IEEE Communications Letters*, 3(3), March 1999.

[MMP07] Jay Kuo Michele Morelli and Man-On Pun. Synchronization techniques for orthogonal frequency division multiple access (ofdma): A tutorial review. *Proceedings of the IEEE*, 95(7), July 2007.

[MSA06] P. Murphy, A. Sabharwal, and B. Aazhang. Design of warp: A flexible wireless open-access research platform. In *Proceedings of EUSIPCO*, 2006.

[MSM99] Gunnar Fock Michael Speth, Stefan A. Fechtel and Heinrich Meyr. Optimum receiver design for wireless broad-band systems using ofdm—part i. *IEEE Transactions on Communications*, 47(11), November 1999.

[MSM01] Gunnar Fock Michael Speth, Stefan A. Fechtel and Heinrich Meyr. Optimum receiver design for wireless broad-band systems using ofdm—part ii: A case study. *IEEE Transactions on Communications*, 49(4), April 2001.

[SC97] Timothy M. Schmidl and Donald C. Cox. Robust frequency and timing synchronization for ofdm. *IEEE Transactions on Communications*, 45(12), December 1997.

[SDCL06] Jung Min Choi Seung Duk Choi and Jae Hong Lee. An initial timing offset estimation method for ofdm systems in rayleigh fading channel. In *Vehicular Technology Conference, 2006. VTC-2006 Fall. 2006 IEEE 64th*, September 2006.