



DIPLOMARBEIT

**Femtosekundengenauere Stabilisierung
von optischen Glasfaserstrecken
basierend auf HF-Leistungsmessung**

**Femtosecond Stabilization
of Optical Fiber Links
Based on RF Power Detection**

vorgelegt dem

Lehrstuhl Mechatronik

Institut für Mobile Systeme (IMS)

Fakultät für Maschinenbau (FMB)

angefertigt am

Deutschen Elektronen-Synchrotron, Hamburg



Verfasser:

Thorsten Lamb

Magdeburg, 15. November 2010

Erstprüfer:

Prof. Dr.-Ing. Roland Kasper
Universität Magdeburg

Zweitprüfer:

Prof. Dr.-Ing. Helmut Tschöke
Universität Magdeburg

Betreuer:

Dipl.-Ing. (FH) Matthias Felber
Deutsches Elektronen-Synchrotron, Hamburg

Aufgabenstellung – Nr. 05/2010 – für eine Diplomarbeit für Herrn Thorsten Lamb

Thema: Femtosekunden Stabilisierung von optischen Glasfaserstrecken basierend auf RF Leistungsmessung

Das Deutsche Elektronen Synchrotron, DESY, in Hamburg entwickelt und betreibt Beschleunigeranlagen zur Produktion weicher und harter Synchrotronstrahlung. Neuartige Synchrotronstrahlungsquellen wie der Freie Elektronen LASer in Hamburg, FLASH, und der im Bau befindliche Europäische XFEL stellen dabei höchste Anforderungen an die Synchronisation von Beschleunigerkomponenten. Die erforderliche Genauigkeit von wenigen zehn Femtosekunden (10^{-14} sec) können nicht mehr mit herkömmlichen Hochfrequenzmethoden erreicht werden. Eine Femtosekunden genau Synchronisation ist nur noch mit optischen Methoden zu erreichen.

Seit einigen Jahren wird bei DESY ein optisches Synchronisationssystem entwickelt. Dieses basiert auf der Verteilung von kurzen Laserpuls in Glasfasern in der Beschleunigeranlage. Die Laserpulse werden in einem Faserlaser bei einer Zentralwellenlänge von 1550nm mit einer Wiederholrate von $f_0=216$ MHz erzeugt. Die optische Länge der Glasfaserstrecke wird mittels Verzögerungsstrecken stabilisiert. Um die Längeänderung einer Glasfaser präzise zu vermessen, wird ein Teil der Laserpulse an dessen Ende reflektiert und zurück zum Laser gesendet. Die reflektierten Pulse werden genutzt um die Umlaufzeit in der Glasfaser zu vermessen, welche durch Temperaturänderungen, Vibrationen und Luftfeuchtigkeitsänderungen stark variieren kann.

Das etablierte Verfahren zur Umlaufzeitmessung basiert auf einem nichtlinearen optischen Prozess (Summenfrequenzgeneration). Dabei wird die Ankunftszeit der reflektierten Laserpulse durch Kreuzkorrelation mit Laserpulsen, die dem Laser direkt entnommen werden, ermittelt. Auf diese Weise können Glasfaserstrecken auf wenige Femtosekunden stabilisiert werden. Diese Methode hat jedoch drei gravierende Nachteile:

1. die Kreuzkorrelation erfordert kurze Laserpulse (< 0.5 ps FWHM) nach dem Durchlaufen der Glasfaserstrecke um ein ausreichend starkes Regelsignal zu erzeugen, was eine aufwändige Dispersionskompensation erfordert,
2. die aus dem Laser direkt emittierten Laserpulse und die Laserpulse die am Ende der Glasfaserstrecke reflektiert werden müssen sich zeitlich und räumlich exakt am nichtlinearen optischen Kristall überlagern,
3. das Zeitfenster, in dem sich die die beiden Pulse überlagern, ist so schmal, dass die Länge der Glasfaserstrecke permanent geregelt sein muss, um den Überlapp jederzeit zu gewährleisten.

Um diese Bedingungen zu erfüllen ist eine aufwändige und kostspielige Optomechanik erforderlich, so dass nur wenige optische Glasfaserverbindungen auf diese Weise stabilisiert werden können (< 20 Glasfaserstrecken).

Um Kosten zu reduzieren und einen breiteren Einsatz des optischen Synchronisationssystems zu ermöglichen ist ein weiteres Verfahren, basierend auf Hochfrequenzleistungsmessung, entwickelt worden. Die Messung der Umlaufzeit basiert hier auf der Detektion der Änderungen der Hochfrequenzleistungen von Frequenzharmonischen der Laserwiederholrate f_0 , die als elektrisches Signal mittels eines Photodetek-

tors generiert wird. Wie auch bei der optischen Kreuzkorrelationsmethode werden die aus dem Laser direkt emittierten und die reflektierten Laserpulse vereint, nun jedoch auf eine Photodiode (InGaAs, > 10GHz Bandbreite) geleitetet. Das elektrische Ausgangssignal der Photodiode besteht aus nadelförmigen Pulsen, die einen Hochfrequenzbandpassfilter durchlaufen zur Selektion z.B. einer einzelnen Frequenzharmonischen ($f = n f_0$, $n=0, 1, \dots$). Treffen die direkten und reflektierten Pulse bei gleicher Amplitude zeitlich gerade um eine halbe Laserpulsperiode T_0 ($T_0=1/f_0$) versetzt auf die Photodiode, treten die ungeraden Harmonischen ($f = (2n+1) f_0$, $n=0, 1, \dots$) durch die hiermit hervorgerufene Verdopplung der Wiederholrate nicht mehr auf. Ein kleiner zeitlicher Versatz zwischen den Laserpulszügen lässt jedoch ein Signal auch bei den ungeraden Harmonischen auftreten, welches zur Messung des zeitlichen Versatzes genutzt werden kann.

Unter Verwendung zweier benachbarter hoher Harmonischer (z.B. 9.3 GHz und 9.5 GHz) können Fehler durch Laseramplitudenschwankungen unterdrückt werden. Dieses wurde in einem leicht abgewandelten Verfahren bereits erfolgreich getestet. Kommende Untersuchungen zielen darauf ab, den Labor-Aufbau zu vereinfachen, weitere Stabilitätsmessungen durchzuführen und erste Schritte für ein optomechanisches Prototypendesign vorzubereiten und somit den Weg für eine industrielle Fertigung zu ermöglichen.

Die Arbeit ist unter Beachtung der durch die FMB herausgegebenen Richtlinien für die Erarbeitung von Studien- und Diplomarbeiten anzufertigen.

Magdeburg, 26. April 2010

Tag der Ausgabe: 03.05.2010
Tag der Abgabe: 04.10.2010



Prof. Dr.-Ing. Roland Kasper
Aufgabensteller (Erstprüfer)

Erstprüfer: Prof. Dr.-Ing. R. Kasper – IMS-MTK
Zweitprüfer: Prof. Dr.-Ing. H. Tschöke – IMS-KM
Betreuer: Dipl.-Ing. M. Felber, DESY Hamburg



Prof. Dr.-Ing. habil. Frank Palis
Vors. des Prüfungsausschusses

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbständig und ohne fremde Hilfe verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die den Werken wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Magdeburg, 15. November 2010

Vorwort

Diese Diplomarbeit wurde während des Sommers 2010 am Deutschen Elektronen Synchrotron in Hamburg angefertigt.

Die offene und produktive Arbeitsatmosphäre, die ich dort vorgefunden habe, war eine große Bereicherung und sie hat maßgeblich zum Gelingen dieser Arbeit beigetragen.

Ich danke an dieser Stelle Prof. Dr.-Ing. Roland Kasper für die Annahme und Betreuung dieses außergewöhnlichen Themas.

Mein besonderer Dank gilt Dr. Holger Schlarb für die Möglichkeit diese Diplomarbeit bei DESY anzufertigen, für die inspirierenden Ideen und die vielen hilfreichen Hinweise.

Ich bedanke mich außerdem bei Matthias Felber für die umfassende Betreuung sowie die unermüdliche Beantwortung meiner zahlreichen Fragen.

Viele weitere Kollegen haben mich in dieser Zeit begleitet, insbesondere Frank Ludwig, Sebastian Schulz, Patrick Geßler, Christian Schmidt, Marie Kristin Bock, Johann Zemella, Gerhard Grygiel, Philip Pototzki, Michael Bousonville, Matthias Hoffmann (MSK), Bernd Steffen, Sven Pfeiffer, Karl-Heinz Matthiesen, Wolfgang Reinsch, Albert Schleiermacher, Matthias Hoffmann (FLA), Bernhard Schmidt, Karol Dlugolecki, Bibiane Wendland und Izabela Malka. Auch bei ihnen bedanke ich mich herzlich für kleine und große Hilfen.

Nicht zu vergessen sind die zahlreichen helfenden Hände im Hintergrund und die übrigen Mitglieder der Gruppen FLA und MSK bei DESY.

Mein persönlicher Dank gilt meiner Partnerin Deborah Sommer, die mir den Rücken freigehalten und mich während der ganzen Zeit unterstützt hat, sowie meinen Eltern, die dieses Studium erst ermöglicht haben.

Zusammenfassung

Röntgenlichtquellen wie der Freie-Elektronen-Laser FLASH in Hamburg oder der zukünftige XFEL erzeugen Lichtblitze mit einer Dauer von wenigen zehn Femtosekunden. Um die zeitlichen Anforderungen an die Synchronisation verschiedener Komponenten auf dieser Zeitskala zu erfüllen, werden bereits erfolgreich optische Synchronisationssysteme eingesetzt.

In der vorliegenden Diplomarbeit wird ein neuartiges Photodioden-basiertes Detektionsschema zur Messung der Driften von optischen Links in einem solchen Synchronisationssystem entwickelt. Das Detektionsschema ist weitgehend driftfrei und sehr robust. Es wird gezeigt, dass die Langzeitstabilität des konstruierten Detektors über 33 h unter 5 fs (peak-to-peak), bei einer Standardabweichung von 0,86 fs, beträgt. Außerdem wird der erfolgreiche Aufbau einer aktiven Stabilisierung eines Faserlinks mit diesem Detektor geschildert.

Abstract

X-ray light sources like the free electron laser FLASH in Hamburg or the future XFEL generate light pulses with durations in the order of a few ten femtoseconds. To fulfill the requirements for the synchronisation of various components on this timescale, optical synchronisation systems are already successfully used.

In this diploma thesis a novel photodiode-based, detection principle for the measurement of drifts in the optical links of such a synchronisation system is developed. The detection principle is nearly drift-free and highly robust. It is demonstrated that the long term stability of the assembled detector over 33 h is below 5 fs (peak to peak) at a standard deviation of 0.86 fs. Furthermore, an active stabilisation of a fibre link using this detector is successfully achieved.

Inhaltsverzeichnis

1. Einleitung	1
2. Allgemeines	3
2.1. Das optische Synchronisationssystem bei FLASH	3
2.2. Kurzpulslaser	4
2.3. Modelle zur Betrachtung von Laserpulszügen	7
2.3.1. Ein Pulszug aus Dirac-Pulsen	7
2.3.2. Ein Pulszug aus Gauß-Pulsen	10
2.3.3. Ein Pulszug aus sech^2 -Pulsen	11
3. Stand der Entwicklung	13
3.1. Direkte Konversion	13
3.2. Optische Kreuzkorrelation	14
3.3. Erweiterte photodiodenbasierte Verfahren	14
4. Optischer Aufbau für den Detektor	17
4.1. Wichtige Optik-Komponenten	18
4.2. Die optische Delayline	23
4.2.1. Der Linearschlitten	24
4.2.2. Simulation der Delayline	25
4.2.3. Delayline auf Basis eines Retroreflektors	30
4.2.4. Messergebnisse und erreichbare Performance	31
4.2.5. Verbesserungsmöglichkeiten und Anwendung	32
4.3. Versuchsaufbau der Optik	34
5. Aufbau des Detektors	37
5.1. Das Messprinzip des Detektors	37
5.2. Simulation des Messprinzips	40
5.3. Komponentenauswahl	41
5.4. Aufbau des Detektors	47
5.5. Arbeitspunkteinstellung	49
5.6. Kalibration	50
5.7. Begrenzungen des Detektionsprinzips	51
5.7.1. Linearität des Messbereichs	52
5.7.2. Rauschen des Detektors	52

5.7.3.	Pulsverbreiterung durch Gruppengeschwindigkeitsdispersion . .	54
5.7.4.	Empfindlichkeit gegenüber anderen Fehlerquellen	55
6.	Passive Driftmessungen	57
6.1.	Die Langzeitmessung	59
6.2.	Die Low-Power Messung	61
6.3.	Die High-Power Messung	63
7.	Entwurf einer digitalen Regelung	65
7.1.	Hardware des digitalen Reglers	66
7.2.	Implementierung des digitalen Reglers	68
7.2.1.	Der digitale PI-Regler	68
7.2.2.	Das digitale IIR-Filter	70
7.3.	Der Aktuator	71
7.4.	Systemcharakterisierung und Reglerauslegung	74
7.4.1.	Untersuchung der offenen Strecke	76
7.4.2.	Temperaturverhalten der Faser	80
7.4.3.	Reglerparametrierung	81
7.4.4.	Führungsverhalten	83
7.4.5.	Störverhalten	84
7.5.	Messungen mit aktiver Regelung	85
8.	Zusammenfassung und Ausblick	91
	Anhang	93
A.	Informationen zum digitalen Anhang	94
B.	Quellcode zur Simulation der Delayline	95
B.1.	Die Klasse <i>beam</i>	95
B.2.	Die Klasse <i>mirror</i>	99
B.3.	Die Klasse <i>retroreflector</i>	105
B.4.	Das Simulationsscript	117
C.	Quellcode zur Simulation des Messprinzips	121
D.	Literaturverzeichnis	132

Abbildungsverzeichnis

2.1.	Schematischer Aufbau des opt. Synchronisationssystems bei FLASH	3
2.2.	Schematischer Aufbau des verwendeten Kurzpulslaser	5
2.3.	Autokorrelationsmessung der Laserpulse	6
2.4.	Modulation des Frequenzkamms für verschiedene Δt	9
3.1.	Driftmessung eines alternativen photodiodenbasierten Verfahrens	15
4.1.	Skizze des optischen Aufbaus	17
4.2.	Polarisation an $\lambda/4$ und $\lambda/2$ Wellenplatten	19
4.3.	Polarisierender Strahlteiler	19
4.4.	Strahlparameter	21
4.5.	Polarisationsdrehung durch den Faraday-Effekt	22
4.6.	Linearschlitten mit Halterung für den Retroreflektor	24
4.7.	Simulation der Delayline mit Retroreflektor	31
4.8.	Einkopplung bei verschiedenen Delaylines	33
4.9.	CAD Modell der Mechanik für OXC basierte Faserlinks	34
4.10.	Testaufbau der Optik mit eingezeichnetem Strahlverlauf	35
5.1.	Schematischer Aufbau des Faserlinks mit Detektor	38
5.2.	Screenshot der Simulation des Detektors	41
5.3.	Frequenzabhängige Sättigung einer Photodiode	42
5.4.	Sättigung einer Photodiode in Abhängigkeit von der Bias-Spannung	43
5.5.	Frequenzgang des HF-Bandpassfilters (IMCSD - Typ 936917)	45
5.6.	Frequenzgang des HF-Bandpassfilters (PROCOM BPF10G/9)	46
5.7.	Foto des aufgebauten Detektors	48
5.8.	Kalibration des Detektors	51
5.9.	Detektorrauschen	53
6.1.	33 Stunden Driftmessung	58
6.2.	Driftmessung mit niedriger Leistung	60
6.3.	Driftmessung mit hoher Leistung und nur einem Verstärker	62
7.1.	VME-Crate mit Einschüben	67
7.2.	Bedienpanel des PID Reglers im Kontrollsystem	69
7.3.	Amplitudengang des digitalen IIR-Filters, G_{IIR}	70
7.4.	Piezostretcher (Evanescence Optics Inc. - Modell 915)	72

7.5. Messschaltung für die elektronische Übertragungsfunktion des Piezostretchers	73
7.6. Bodediagramm der Übertragungsfunktion des Piezostretchers	73
7.7. Bipolarer Piezotreiber mit Netzteil	74
7.8. Die aufgebaute Regelung und das angenäherte Modell zur Simulation .	75
7.9. Sprungantwort der offenen Strecke	76
7.10. Bodediagramm der angenäherten Streckenübertragungsfunktion	77
7.11. Sprungantwort der offenen Strecke mit Filter	79
7.12. Bodediagramm der offenen Strecke mit Filter	79
7.13. Sprungantwort für Temperaturänderungen an der Linkfaser	80
7.14. Bodediagramm des Modells der offenen Strecke	82
7.15. Wurzelortskurve der geschlossenen Regelschleife	83
7.16. Systemantwort auf eine sprunghafte Änderung der Führungsgröße . .	84
7.17. Systemantwort auf eine sprunghafte Änderung der Störgröße	84
7.18. Bodediagramm der Störübertragungsfunktion	85
7.19. Temperatursprung mit aktiver Regelung	87
7.20. Langzeitmessung mit aktiver Regelung	89

Tabellenverzeichnis

2.1. Umrechnungsparameter für FWHM Pulsbreiten	5
5.1. Kalibrationskonstante K_φ des Detektors	50

Quellcodeverzeichnis

4.1. Ausschnitt aus dem Simulationsscript	29
---	----

1. Einleitung

Röntgenlichtquellen der nächsten Generation sind in der Lage, extrem kurze Lichtblitze mit gleichzeitig hoher Brillanz¹ bereitzustellen. Die Erzeugung dieser Lichtblitze mit einer Dauer von wenigen zehn Femtosekunden und Wellenlängen im kurzwelligen ultravioletten und weichen Röntgenbereich² stellt höchste technische Anforderungen an den Bau und den Betrieb der Beschleuniger, die diese Prozesse erst ermöglichen.

Beim **Freien Elektronen Laser** in **Hamburg** (FLASH) sowie auch beim derzeit im Bau befindlichen Europäischen XFEL werden in einem supraleitenden Linearbeschleuniger Elektronenpakete beschleunigt. Bei Energien von bis zu 1,25 GeV (bei FLASH) wird das komprimierte Elektronenpaket im sogenannten Undulator in einem SASE-Prozess (Self-Amplified Spontaneous Emission) zur Emission des kohärenten Röntgenlichts angeregt. Das erzeugte Licht steht dann für wissenschaftliche Experimente zur Verfügung.

Insbesondere bei der Diagnose des Elektronenstrahls und der Synchronisation der verschiedenen Experimente am Beschleuniger muss eine hohe zeitliche Auflösung erreicht werden, um einen effizienten Betrieb zu gewährleisten. Die geforderten Toleranzen liegen üblicherweise ebenfalls in einer Größenordnung von wenigen zehn Femtosekunden und sie sind mit herkömmlicher Hochfrequenztechnik nicht und vor allem nicht wirtschaftlich zu erreichen.

Aus diesem Grund werden inzwischen in solchen Einrichtungen erfolgreich optische Synchronisationssysteme entwickelt und betrieben, um periodische Synchronisationssignale mit hoher Phasengenauigkeit im Beschleuniger zu verteilen. Die Laufzeit der Glasfaserstrecken, die dort zur Übertragung der Synchronisationsinformationen dienen, wird dabei aktiv stabilisiert, was zunächst die Detektion der auftretenden Phasendriften erfordert.

Die etablierten Verfahren zur Detektion dieser Phasendriften sind, wie im Fall der optischen Kreuzkorrelation, zwar hochgenau, erfordern jedoch einen enormen technischen Aufwand, der nur den Betrieb weniger auf diese Weise stabilisierter Faserlinks erlaubt. Oder aber, sie erfüllen wie im Fall der direkten Konversion mit Photodioden und anschließender Phasendetektion im Hochfrequenzbereich nicht die gestellten Anforderungen an die Genauigkeit und Stabilität des Detektors.

¹Die Brillanz ist definiert als $B = \text{Photonenzahl}/(\text{Pulsdauer}[\text{s}] \cdot \text{Fläche}[\text{mm}^2] \cdot \text{Raumwinkel}[\text{mrad}^2] \cdot 0,1\% \text{Bandbreite})$

²Bei FLASH zwischen 44 nm und 4,1 nm

1. Einleitung

In der folgenden Diplomarbeit wird deshalb ein neuartiger Detektor entwickelt und getestet, der diese Nachteile umgeht. Der Detektor vereint die Vorteile photodiodenbasierter Detektoren, nämlich ihr vergleichsweise einfacher Aufbau und ihre geringeren Kosten mit denen der optischen Kreuzkorrelation, die eine wesentlich genauere Messung erlaubt.

Zur Entwicklung dieses Detektors wird zunächst der optische Aufbau des Detektors entworfen und optimiert. Anhand eines im Rahmen der Diplomarbeit gebauten Prototypen wird untersucht, welche Genauigkeit mit diesem neuen Detektionsschema tatsächlich erreicht werden kann. Schlussendlich wird die aktive Regelung eines Faserlinks auf Basis dieses Detektors implementiert.

2. Allgemeines

Im optischen Synchronisationssystem sind die Synchronisationsinformationen in der hoch genauen Repetitionsrate eines Laserpulszuges kodiert. Um einen Überblick über die Funktionsweise optischer Synchronisationssysteme zu erhalten, wird zunächst exemplarisch das bei FLASH implementierte und bereits betriebene System vorgestellt. Im Hinblick auf den später entwickelten, neuen Detektor wird anschließend kurz auf das Funktionsprinzip der eingesetzten Faserlaser eingegangen, um dann mathematisch zu untersuchen, welche Auswirkungen die Überlagerung zweier Pulszüge im Frequenzbereich hat.

2.1. Das optische Synchronisationssystem bei FLASH

Abbildung 2.1 zeigt den schematischen Aufbau des optischen Synchronisationssystems bei FLASH. Rot beziehungsweise lila sind optische Faserlinks eingezeichnet, die bereits betrieben werden, projektierte Links sind grau dargestellt. Man kann erkennen, dass am Beschleuniger selbst viele Komponenten auf den Betrieb von Lasern verschiedenster Art (blau) angewiesen sind.

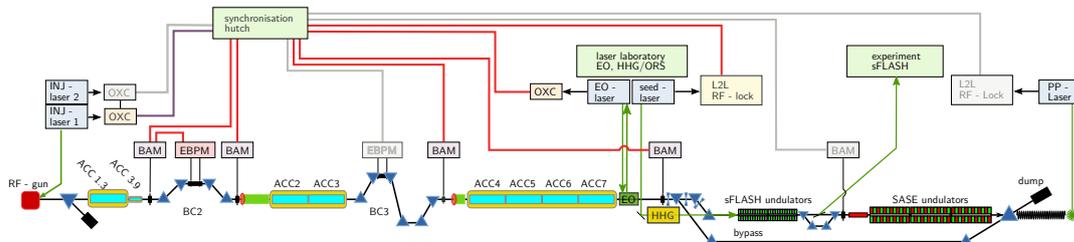


Abbildung 2.1.: Schematischer Aufbau des optischen Synchronisationssystems bei FLASH [modifiziert aus BFG+10, S. 2]

Das optische Synchronisationssystem ist auch deshalb so hervorragend zur Synchronisierung dieser Komponenten mit dem Master Oszillator des Beschleunigers geeignet, weil es möglich ist, diese Laser direkt in einem optischen Verfahren an das Synchronisationssignal anzubinden. Hier kommt beispielsweise ein two-color optical cross correlator zum Einsatz, der in einem nichtlinearen optischen Prozess ein Messsignal für die Phasenverschiebung zwischen den Lasern erzeugt, mit dem dann eine sogenannte Phase

Locked Loop (PLL) aufgebaut werden kann. Andere Applikationen wie beispielsweise die Bunch Arrivaltime Monitore (BAM, lila) verwenden das gepulste Laserlicht des Synchronisationssystems direkt zum hoch genauen Messen eines vom Elektronenstrahl erzeugten Hochfrequenzsignals (HF-Signals).

Um diesen Endgeräten eine exakte Referenz in Form des bereits erwähnten Laserpulszuges zur Verfügung stellen zu können, muss die Laufzeit in der Übertragungsstrecke für das Synchronisationssignal, nämlich der Glasfaser, aktiv stabilisiert werden. Dazu wird am Anfang eines jeden Links ein Detektionssystem aufgebaut, das die Phasenverschiebung zwischen einem Referenzpulszug direkt aus dem Master Laser Oszillator (MLO) des optischen Synchronisationssystems und einem Pulszug, der am Linkende vom Nutzsignal abgetrennt und zurück reflektiert wurde, bestimmt. Der zurück reflektierte Pulszug durchläuft insgesamt zweimal die Glasfaser des Links. Die Laufzeit wird durch von außen einwirkende Temperaturänderungen beeinflusst. Wenn die Phasenbeziehung diese Pulszuges zum Referenzpulszug durch aktive Regelung konstant gehalten wird, kann angenommen werden, dass auch die Phasenbeziehung des Linkendes zum MLO konstant ist.

Die Komponenten des Synchronisationssystems werden bei FLASH in einer mit großem Aufwand klimatisierten und gegen elektromagnetische Störungen geschützten Umgebung, der sogenannten „synchronization hutch“, betrieben, um äußere Einflüsse auf die Performance des Synchronisationssystems soweit wie möglich zu eliminieren.

2.2. Kurzpulslaser

Der Laserpulszug zur Übertragung der Synchronisationsinformationen im optischen Synchronisationssystem wird von einem sogenannten Kurzpuls- oder Femtosekunden-Laser erzeugt. Bei solchen Lasern bildet sich innerhalb des Laserresonators eine große Zahl longitudinaler Moden aus. Die Überlagerung der Moden ergibt einen extrem kurzen Laserpuls, wenn durch Modenkopplung eine starre Phase zwischen ihnen hergestellt wird. In diesem stationären Zustand kann bei jedem Umlauf ein Teil des Pulses als nutzbares, gepulstes Laserlicht ausgekoppelt werden. Mit solchen Lasern können bei hohen Spitzenleistungen typischerweise Pulslängen von wenigen zehn Femto- bis hin zu einigen Pikosekunden erreicht werden [vgl. ST91, S. 535].

Die Pulsform kann aus der nichtlinearen Wellengleichung für den verwendeten Resonator abgeleitet werden. Eine mögliche Lösung ist das Soliton mit dem zeitlichen Leistungsprofil einer sech^2 -Funktion. Die Intensität eines Laserpulses ist zur Leistung proportional.

$$\mathcal{P}_{\text{sech}^2}(t) = \mathcal{P}_p \cdot \text{sech}^2\left(\frac{t}{\sigma_s}\right) \quad (2.1)$$

Zur Vereinfachung wird oft ein gaußförmiger Puls angenommen, der sich nur minimal von einem *sech*²-Puls unterscheidet.

$$\mathcal{P}_{Gauss}(t) = \mathcal{P}_p \cdot e^{-\left(\frac{t}{\sigma_g}\right)^2} \quad (2.2)$$

Die Pulsbreite wird in der Laserphysik meist als sogenanntes Full Width at Half Maximum (FWHM) angegeben. Die in den beiden oben genannten Formeln genannte Pulsbreite σ kann entsprechend Tabelle 2.1 in eine FWHM-Pulsbreite τ_p umgerechnet werden.

	Faktor	FWHM τ_p
sech²-Puls	$2 \cdot \operatorname{acosh}(\sqrt{2})$	$1,763 \sigma_s$
Gauss-Puls	$2 \cdot \sqrt{\ln(2)}$	$1,665 \sigma_g$

Tabelle 2.1.: Umrechnungsparameter für FWHM Pulsbreiten

Für das Synchronisationssystem von FLASH wurde inzwischen aus Gründen der Ausfallsicherheit ein kommerzieller Origami-15 Laser des Schweizer Herstellers onefive als Master Laser Oszillator (MLO) installiert. Das Synchronisationssystem ist für eine Wellenlänge von 1560 nm (C-Band der Telekommunikationsindustrie) ausgelegt und der MLO arbeitet bei einer Repetitionsrate von 216,6 MHz, also der sechsten Subharmonischen der Beschleunigerfrequenz von 1,3 GHz.

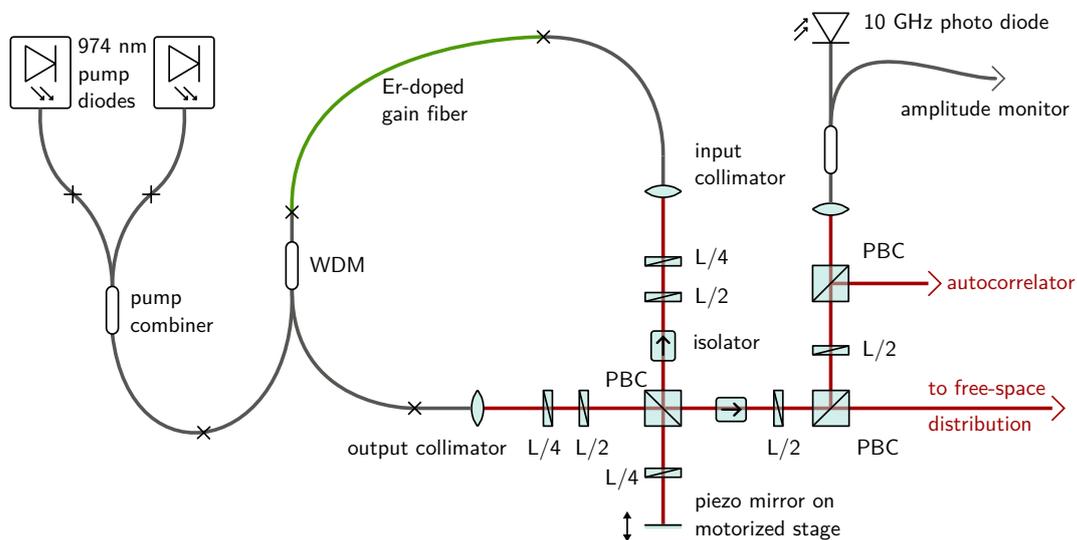


Abbildung 2.2.: Schematischer Aufbau des verwendeten Kurzpuls laser [modifiziert aus Sch10, S. 7]

Abbildung 2.2 zeigt eine schematische Übersicht über den im Folgenden verwendeten, passiv modengekoppelten Erbium-dotierten Faserlaser. Es handelt sich hierbei um die industrialisierte Version eines bei DESY entwickelten Lasers mit den gleichen Parametern was Wellenlänge und Repetitionsrate betrifft. Die erbiumdotierte Gainfaser wird durch zwei Laserdioden bei einer Wellenlänge von 974 nm gepumpt. Die Wellenplatten im Laser sind motorisiert. Sie werden benötigt, um einen Modelock-Zustand einzustellen. Die Laseramplitude wird über die Pumpleistung und einen Amplitudenmonitor stabilisiert. Die Repetitionsrate kann über einen motorisierten Spiegel, der auf einem Piezoaktuator befestigt ist, eingestellt werden.

Allgemein weist die Repetitionsrate solcher Kurzpulslaser eine sehr gute Kurzzeitstabilität auf, die Phase driftet jedoch über größere Zeiträume im Bereich von einigen Pikosekunden. Aus diesem Grund werden solche Laser üblicherweise mit einer Phase Locked Loop (PLL) an einen langzeitstabilen Quarzoszillator angebunden. Bei FLASH wird das optische Synchronisationssystem auf den Mastersoszillator des Beschleunigers gelockt. Für den oben beschriebenen Laborlaser steht ein kommerzieller Dielektrischer Resonator Oszillator (DRO) der Firma Poseidon Scientific Instruments zur Verfügung, der an einen beheizten Quarz mit hervorragender Langzeitstabilität angebunden ist.

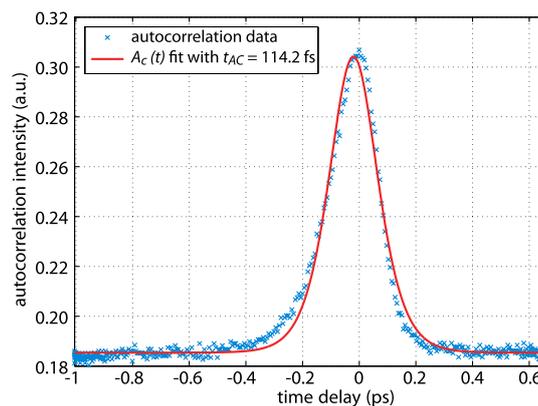


Abbildung 2.3.: Autokorrelationsmessung der Laserpulse

Die Pulsform dieses Lasers wird für spätere Berechnungen mit Hilfe eines Autokorrelators bestimmt. Das Messergebnis ist in Abbildung 2.3 zu sehen. Die leichte Schiefe des Pulses wurde vermutlich durch unpräzise Ansteuerung des Autokorrelators verursacht. Die angegebene Pulsbreite von $t_{AC} = 114,2$ fs muss noch über einen, aus der Autokorrelationsfunktion für die spezifische Pulsform bestimmten, Umrechnungsfaktor skaliert werden [DR06, S. 477]. Die tatsächliche Pulsbreite des Lasers beträgt unter Annahme eines sech^2 -Pulses

$$\tau_p = 114,2 \text{ fs} / 1,543 = 74 \text{ fs}$$

2.3. Modelle zur Betrachtung von Laserpulszügen

Zur qualitativen Simulation des Messprinzips werden zunächst die spektralen Eigenschaften zweier überlagerter Laserpulszüge mathematisch betrachtet. Mit den Ergebnissen dieser Untersuchung wird dann in Matlab eine Simulation entworfen. Im Messaufbau werden die überlagerten Pulszüge später mit schnellen 10 GHz Photodioden ausgewertet.

2.3.1. Ein Pulszug aus Dirac-Pulsen

Vereinfacht kann man einen Laserpulszug zunächst als eine unendliche Folge zeitverschobener Dirac-Pulse betrachten. Die Dirac-Pulse beschreiben hier das zeitliche Leistungsprofil der Laserpulse. Eine Betrachtung des elektrischen Feldes ist nicht notwendig, da die Pulse durch ihren zeitlichen Versatz nicht miteinander interferieren können. Die Pulse werden mit der mittleren Leistung P skaliert, die Periodendauer T_0 folgt aus der Repetitionsrate f_0 des Lasers, in diesem Fall 216,6 MHz. Zur Untersuchung von Dirac-Pulsen siehe auch [Zem08, S. 11 f.].

Die Formel für eine Folge aus verschobenen Dirac-Pulsen lautet:

$$f(t) = \sum_{n=-\infty}^{\infty} P \cdot \delta(t - nT_0) \quad (2.3)$$

Mit Hilfe der Fouriertransformation [siehe HSZ03, S. 412 ff.] kann diese Folge in den Frequenzbereich transformiert werden. Sei $\hat{f}(j\omega) = \mathcal{F}\{f(t)\}(j\omega)$ die Fouriertransformierte von $f(t)$ so ergibt sich $\hat{f}(j\omega)$ unter Berücksichtigung des Verschiebungssatzes zu

$$\begin{aligned} \hat{f}(j\omega) &= \frac{P}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \delta(t - nT_0) \cdot e^{-j\omega t} dt \\ &= \frac{P}{\sqrt{2\pi}} \sum_{n=-\infty}^{\infty} e^{-j\omega nT_0} \end{aligned} \quad (2.4)$$

Überlagert man nun zwei Pulszüge mit einem zeitlichen Versatz von Δt und nimmt weiter an, dass beide Pulse die exakt gleiche Repetitionsrate T_0 haben, erhält man folgende mathematische Beschreibung für die überlagerten Pulszüge im Zeitbereich

$$f_m(t) = \sum_{n=-\infty}^{\infty} \left[P_1 \cdot \delta(t - nT_0) + P_2 \cdot \delta(t - nT_0 + \Delta t) \right] \quad (2.5)$$

Das Spektrum der überlagerten Pulszüge erhält man durch Transformation von $f_m(t)$ in den Frequenzbereich. Die Fouriertransformierten der Pulszüge können aufgrund der

Linearität der Fouriertransformation getrennt berechnet werden.

$$\begin{aligned}
 \hat{f}_m(j\omega) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \left[P_1 \cdot \delta(t - nT_0) + P_2 \cdot \delta(t - nT_0 + \Delta t) \right] \cdot e^{-j\omega t} dt \\
 &= \frac{P_1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \delta(t - nT_0) \cdot e^{-j\omega t} dt \\
 &\quad + \frac{P_2}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \delta(t - nT_0 + \Delta t) \cdot e^{-j\omega t} dt \\
 &= \frac{1}{\sqrt{2\pi}} \sum_{n=-\infty}^{\infty} \left(P_1 + P_2 \cdot e^{j\omega \Delta t} \right) \cdot e^{-j\omega nT_0} \tag{2.6}
 \end{aligned}$$

Der durch die überlagerten Pulszüge in der Photodiode erzeugte Photostrom ist proportional zur optischen Leistung. Die Spannungsamplitude der erzeugten Harmonischen im Zeitbereich ist also ebenfalls proportional zur optischen Leistung. Von Interesse für die Simulation ist allerdings nicht die hier betrachtete komplexe Leistung, sondern zunächst nur ihr Betrag, mit dem sich das Frequenzspektrum zeichnen lässt. Die Intensität oder Amplitude einer Frequenzlinie $I(\omega_n)$ ist also

$$I(\omega_n) \propto \left| \hat{f}_m(\omega_n) \right| \tag{2.7}$$

Allgemein lässt sich die Leistung in einer Frequenzharmonischen des Laserlichts also durch

$$I(\omega_n) \propto \sqrt{\frac{P_1^2 + 2P_1P_2 \cos(\Delta t\omega_n) + P_2^2}{2\pi}} \tag{2.8}$$

ausdrücken, wobei $\omega_n = 2\pi \cdot n \cdot f_0$ ist. Man erhält im Frequenzbereich einen Kamm mit der Grundfrequenz bei der Repetitionsrate f_0 , sowie allen höheren Harmonischen. Die Frequenzlinien werden durch die Überlagerung der Pulszüge moduliert. Zur anschaulichen Betrachtung bietet sich die Annahme, dass die Leistung der beiden Pulszüge gleich ist, an. Die Formel vereinfacht sich mit $P_1 = P_2 = P$ zu

$$I(\omega_n) \propto P \sqrt{\frac{2}{\pi}} \left| \cos\left(\frac{\Delta t\omega_n}{2}\right) \right| \tag{2.9}$$

Der Kosinus wird bei einer Harmonischen ω_n genau dann zu Null, wenn sein Argument ein ungerades, ganzzahliges Vielfaches von $\pi/2$ beträgt. Diese Bedingung tritt für alle ungeraden Harmonischen ein, wenn $\Delta t = T_0/2$ beträgt. Mit anderen Worten wird jede ungerade Harmonische der Repetitionsrate ausgelöscht, wenn die Leistung der beiden Pulszüge gleich ist und sie einen Phasenversatz zueinander von ihrer halben Periodendauer haben. Im Zeitbereich hat sich die Repetitionsrate in diesem Fall verdoppelt, die Grundrepetitionsrate ist verschwunden.

Wenn die Pulse nicht exakt die gleiche Leistung haben oder der zeitliche Versatz zwischen den Pulszügen nicht genau der halben Periodendauer entspricht, tritt eine komplexere Modulation des Frequenzkamms auf. Die Einhüllende dieser Modulation ist definiert durch Gleichung 2.8, beziehungsweise im vereinfachten Fall durch Gleichung 2.9 wenn man die Funktionen nicht nur für ω_n , sondern für kontinuierliche ω betrachtet.

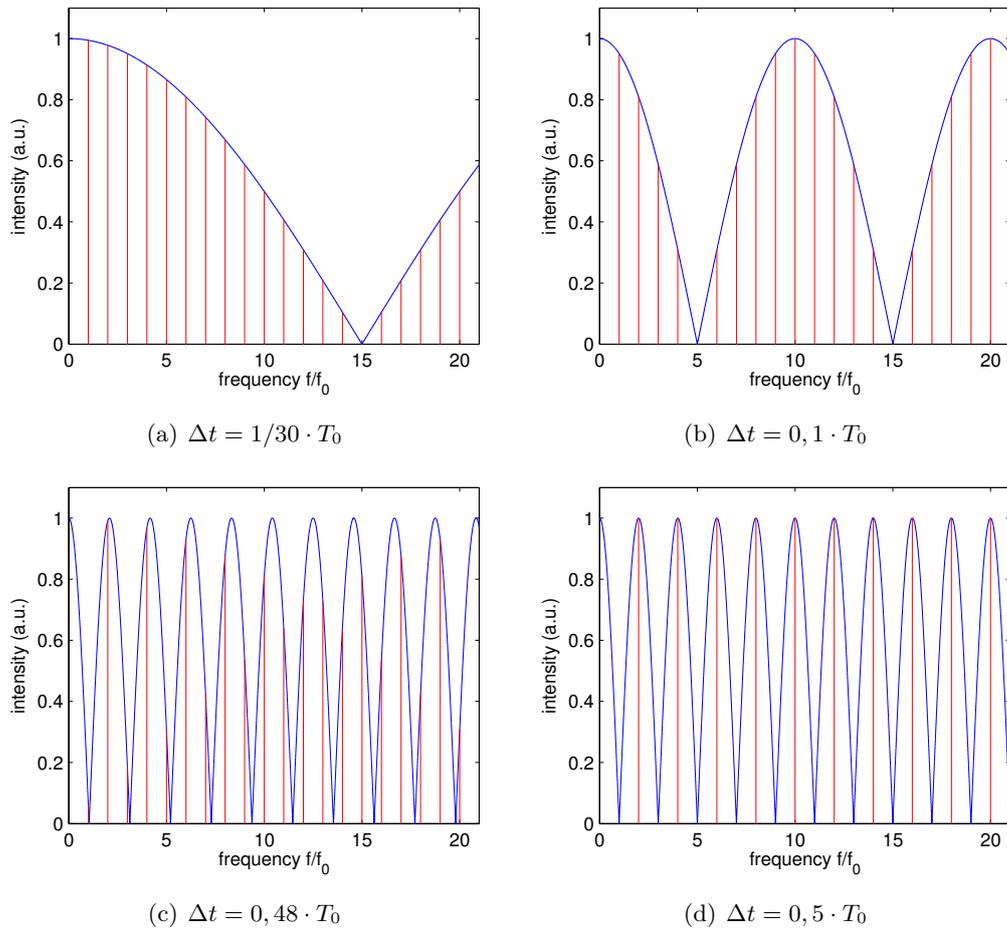


Abbildung 2.4.: Modulation des Frequenzkamms für verschiedene Δt

Abbildung 2.4 zeigt für vier verschiedene Zeitverschiebungen Δt den vereinfachten Fall, dass die Leistung der Pulszüge gleich ist. Die Frequenzharmonischen sind rot eingezeichnet, die Einhüllende blau. Die Intensität oder auch Amplitude der Harmonischen ist normiert.

Vergleicht man beispielsweise die 20. Harmonische in den beiden unteren Abbildungen, erkennt man eine große Änderung der Amplitude, obwohl die Laserpulse nur um 2% von T_0 gegeneinander verschoben wurden. Je höher die Frequenz der Harmonischen, die man

betrachtet ist, desto häufiger wird sie moduliert wenn man die Pulse um eine Periode T_0 gegeneinander verschiebt. Selektiert man zur Messung des Zeitversatzes eine möglichst hochfrequente Harmonische aus dem Frequenzkamm, ist die Amplitudenmodulation bei dieser hohen Frequenz sensitiver gegenüber Timingänderungen. In der Abbildung rechts unten ist darüber hinaus ein möglicher Arbeitspunkt für die 20. Harmonische abgebildet, da sie komplett ausgelöscht wird.

Zur Detektion dieser Amplitudenmodulation gibt es verschiedene Ansätze. Ein Ansatz wird in [Zem08] beschrieben. Diese Messmethode wird in Abschnitt 3.3 kurz vorgestellt und die Ergebnisse dieser Untersuchung werden zusammengefasst. Hier wird ein anderer Ansatz verfolgt, der in den nächsten Kapiteln ausführlich erläutert wird.

Was bisher außer acht gelassen wurde, ist die Tatsache, dass das zeitliche Profil realer Laserpulse nicht einem Dirac-Puls entspricht, da reale Laserpulse keine verschwindende zeitliche Ausdehnung aufweisen. Die in Ultrakurzpulslasern erzeugten sech^2 -Pulse werden in der Literatur oft durch Gauß-Pulse angenähert (siehe auch Abschnitt 2.2).

Untersucht man einen Pulszug aus Gauß- oder sech^2 -Pulsen, so stellt man fest, dass die Amplitude der hohen Harmonischen mit steigender Pulsbreite abnimmt. Um beurteilen zu können ob dieser Effekt bei den hier durchgeführten Untersuchungen eine Rolle spielt, werden die Fouriertransformierten von Pulszügen mit diesen beiden Pulsformen mittels Mathematica bestimmt. Die Ausgangsgleichungen weichen von den in Abschnitt 2.2 genannten ab, da die Pulse hier normiert werden, um die Energie auch bei Veränderungen der Pulsbreite konstant zu halten.

2.3.2. Ein Pulszug aus Gauß-Pulsen

Für zwei zueinander verschobene Kämmen mit Gauß-Pulsen gilt unter Berücksichtigung der normierten mittleren Leistung

$$f_{m,Gauss}(t) = \sum_{n=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} \left[\frac{P_1}{\sigma_1} \cdot e^{-\frac{1}{2} \left(\frac{t-nT_0}{\sigma_1} \right)^2} + \frac{P_2}{\sigma_2} \cdot e^{-\frac{1}{2} \left(\frac{t-nT_0+\Delta t}{\sigma_2} \right)^2} \right] \quad (2.10)$$

Die Fouriertransformierte kann wie zuvor bei den Dirac-Pulsen bestimmt werden.

$$\begin{aligned} \hat{f}_{m,Gauss}(j\omega) &= \frac{1}{2\pi} \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \left[\frac{P_1}{\sigma_1} \cdot e^{-\frac{1}{2} \left(\frac{t-nT_0}{\sigma_1} \right)^2} + \frac{P_2}{\sigma_2} \cdot e^{-\frac{1}{2} \left(\frac{t-nT_0+\Delta t}{\sigma_2} \right)^2} \right] \cdot e^{-j\omega t} dt \\ &= \frac{1}{2\pi} \cdot \frac{P_1}{\sigma_1} \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} e^{-\frac{1}{2} \left(\frac{t-nT_0}{\sigma_1} \right)^2} \cdot e^{-j\omega t} dt \\ &\quad + \frac{1}{2\pi} \cdot \frac{P_2}{\sigma_2} \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} e^{-\frac{1}{2} \left(\frac{t-nT_0+\Delta t}{\sigma_2} \right)^2} \cdot e^{-j\omega t} dt \end{aligned}$$

$$\hat{f}_{m,Gauss}(j\omega) = \frac{1}{\sqrt{2\pi}} \sum_{n=-\infty}^{\infty} \left(P_1 \cdot e^{-1/2(\sigma_1\omega)^2} + P_2 \cdot e^{-1/2(\sigma_2\omega)^2} e^{j\omega\Delta t} \right) \cdot e^{-j\omega n T_0} \quad (2.11)$$

Die Leistung einer Harmonischen beträgt nach Gleichung 2.7 und Gleichung 2.11

$$I_{Gauss}(\omega_n) \propto \sqrt{\frac{1}{2\pi} \left[P_1^2 \cdot e^{-(\sigma_1\omega_n)^2} + P_2^2 \cdot e^{-(\sigma_2\omega_n)^2} \dots \right.} \\ \left. + 2P_1P_2 \cdot e^{-(\sigma_1^2+\sigma_2^2)\omega_n^2} \cos(-\Delta t\omega_n) \right]} \quad (2.12)$$

Für $\sigma \rightarrow 0$ ergibt sich exakt das gleiche Ergebnis wie in Gleichung 2.8. Mit steigender Pulsbreite und steigender Frequenz der betrachteten Harmonischen nimmt ihre Leistung ab. Dadurch wird auch die Messempfindlichkeit bei den eigentlich bevorzugten, hohen Harmonischen reduziert.

2.3.3. Ein Pulszug aus sech²-Pulsen

Die Modellierung mit sech²-Pulsen führt zur folgenden Gleichung:

$$f_{m,sech^2}(t) = \sum_{n=-\infty}^{\infty} \left[\frac{P_1}{\sigma_1} \operatorname{sech}^2\left(\frac{t-nT_0}{\sigma_1}\right) + \frac{P_2}{\sigma_2} \operatorname{sech}^2\left(\frac{t-nT_0+\Delta t}{\sigma_2}\right) \right] \quad (2.13)$$

Die Fouriertransformierte kann analog zu den anderen Pulszügen berechnet werden.

$$\begin{aligned} \hat{f}_{m,sech^2}(j\omega) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \left[\frac{P_1}{\sigma_1} \operatorname{sech}^2\left(\frac{t-nT_0}{\sigma_1}\right) \right. \\ &\quad \left. + \frac{P_2}{\sigma_2} \operatorname{sech}^2\left(\frac{t-nT_0+\Delta t}{\sigma_2}\right) \right] \cdot e^{-j\omega t} dt \\ &= \frac{1}{\sqrt{2\pi}} \cdot \frac{P_1}{\sigma_1} \cdot \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \operatorname{sech}^2\left(\frac{t-nT_0}{\sigma_1}\right) \cdot e^{-j\omega t} dt \\ &\quad + \frac{1}{\sqrt{2\pi}} \cdot \frac{P_2}{\sigma_2} \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \operatorname{sech}^2\left(\frac{t-nT_0+\Delta t}{\sigma_2}\right) \cdot e^{-j\omega t} dt \\ &= \sum_{n=-\infty}^{\infty} \sqrt{\frac{\pi\omega^2}{2}} \left[P_1\sigma_1 \operatorname{csch}\left(\frac{1}{2}\pi\sigma_1\omega\right) \right. \\ &\quad \left. + P_2\sigma_2 \operatorname{csch}\left(\frac{1}{2}\pi\sigma_2\omega\right) e^{j\omega\Delta t} \right] e^{-j\omega n T_0} \end{aligned} \quad (2.14)$$

Die Leistung einer Harmonischen beträgt nach Gleichung 2.7 und Gleichung 2.14

$$I_{sech^2}(\omega_n) \propto \sqrt{\frac{\pi\omega^2}{2} \left[P_1^2\sigma_1^2 \cdot \operatorname{csch}^2\left(\frac{\pi\sigma_1\omega_n}{2}\right) + P_2^2\sigma_2^2 \cdot \operatorname{csch}^2\left(\frac{\pi\sigma_2\omega_n}{2}\right) \dots \right.} \\ \left. + 2P_1P_2\sigma_1\sigma_2 \cdot \cos(-\omega_n\Delta t) \cdot \operatorname{csch}\left(\frac{\pi\sigma_1\omega_n}{2}\right) \cdot \operatorname{csch}\left(\frac{\pi\sigma_2\omega_n}{2}\right) \right]} \quad (2.15)$$

3. Stand der Entwicklung

Optische Synchronisationssysteme sind derzeit bereits bei FLASH und an anderen Beschleunigern in der ganzen Welt im Betrieb. In den folgenden Abschnitten werden drei gebräuchliche Schemata zur Auswertung der Synchronisationssignale erläutert. Anschließend wird ein Verfahren vorgestellt, das eine gute Performance zeigt, jedoch in dieser Diplomarbeit maßgeblich abgeändert und weiterentwickelt wird.

3.1. Direkte Konversion

Mit direkter Konversion wird die Erzeugung eines HF-Signals direkt aus dem optischen Pulszug bezeichnet. Das optische Synchronisationssignal wird hierzu in einen elektrischen Pulszug umgewandelt und mit Hilfe eines Bandpassfilters wird die Frequenzkomponente herausgefiltert, die für die entsprechende Anwendung benötigt wird. Die Vorteile dieser Vorgehensweise liegen auf der Hand, da für diesen Aufbau zunächst nur zwei einfache Bauteile benötigt werden und man bis zur Bandbreite der Photodiode harmonische Signale bei allen Vielfachen der Grundfrequenz, also der Repetitionsrate, extrahieren kann.

Die Repetitionsrate des Lasers ist so gewählt, dass sich fast alle im Beschleuniger relevanten Frequenzen aus ihr ohne weitere Frequenzumsetzung auf der HF-Seite erzeugen lassen. Man kann diese Methode auch zur Stabilisierung von Faserlinks einsetzen, indem man einen HF-Mischer als Phasendetektor betreibt und ein mit diesem Verfahren aus dem im Link reflektierten Pulszug gewonnenes Signal mit einem direkt vom Master Laser Oszillator abgeleiteten Referenzsignal mischt.

Der gravierende Nachteil dieser Lösung ist ihre für viele Anwendungen nicht ausreichende Rausch- und Driftperformance. Hier spielen verschiedene Effekte eine Rolle. Den größten Einfluss hat der sogenannte AM/PM¹ Konversion in der Photodiode. Dieser Effekt beschreibt das Phasenverhalten der Photodiode bei Leistungsänderungen. In Untersuchungen wurde gezeigt, dass dieser Einfluss etwa $1,3 \text{ ps/mW}$ beträgt, gemessen wurde bei einer Frequenz von $1,3 \text{ GHz}$ [vgl. LLS+07]. Dieser Wert variiert für verschiedene Photodioden.

Die Temperaturempfindlichkeit von Photodioden konnte in der gleichen Veröffentlichung zu 150 fs/K bestimmt werden. In [LML+07] wird die Temperaturabhängigkeit

¹ Amplituden-Modulation/Phasen-Modulation

der ganzen Kette aus Photodiode, Bandpassfilter und einem HF-Verstärker auf ihre Temperaturempfindlichkeit hin untersucht und Temperaturkoeffizienten in der Größenordnung von 330 fs/K gemessen. Die AM/PM Effekte bleiben hier natürlich noch unberücksichtigt.

Der einfache und günstige Aufbau dieses Detektionsschemas wird also auf Kosten der Performance erkauft, die den hier gestellten Anforderungen für die Linkstabilisierung nicht gerecht wird.

3.2. Optische Kreuzkorrelation

Die optische Kreuzkorrelation bezeichnet ein Messverfahren zur Bestimmung des Zeitversatzes zweier optischer Pulszüge, wie sie in den bereits beschriebenen Faserlinks auftreten. Es ist das derzeit bei DESY verwendete Verfahren zur Messung des Timings in optischen Links. Der Zeitversatz wird hier mit einem nichtlinearen Kristall gemessen. Hinter dem Kristall werden die durch Summenfrequenzgeneration erzeugten höherfrequenten Harmonischen der überlagerten Pulszüge mit zwei Detektoren verglichen.

Im Gegensatz zu den anderen vorgestellten Verfahren muss hier eine exakte Überlagerung der Pulszüge gewährleistet werden, um ein messbares Signal zu erzeugen. Diese Bedingung ist, bei der für das Verfahren erforderlichen Pulsbreite von weniger als 500 fs (FWHM), nur durch einen anspruchsvollen und kostspieligen optomechanischen Aufbau zu erreichen.

In [LSC+07] wird über einen Zeitraum von 12 h die Stabilisierung eines Faserlinks mit 25 fs (peak-to-peak) bei einer Standardabweichung von 7,5 fs gezeigt.

3.3. Erweiterte photodiodenbasierte Verfahren

Neben den beiden vorgestellten Verfahren, die derzeit hauptsächlich eingesetzt werden, wurde bereits 2008 ein anderes Verfahren auf der Basis von Photodioden vorgeschlagen [vgl. Zem08]. Hier wird die in Abschnitt 2.3 bereits erläuterte Überlagerung zweier Pulszüge verwendet. Diese Pulszüge werden mit einer Photodiode gemessen und es werden mit Bandpassfiltern zwei benachbarte Harmonische aus dem Frequenzkamm selektiert. Der Arbeitspunkt wird jetzt so gewählt, dass für kleine Timingänderungen die Amplitude einer der beiden Harmonischen durch die Modulation kleiner wird, die Amplitude der anderen größer. Amplitudenschwankungen wirken sich auf beide Pulszüge gleichermaßen aus und werden deshalb unterdrückt. Um eine hohe Sensitivität gegenüber Timingänderungen zu gewährleisten, werden die 44. und 45. Harmonischen bei 9,53 GHz und 9,75 GHz ausgewählt.

In einem ersten Aufbau wurden diese Harmonischen zunächst getrennt verstärkt und dann in einem Differenzverstärker ausgewertet. Außerdem wurde hier der optische Aufbau komplett in Faser vorgenommen, was das Setup extrem kompliziert macht, da die Längen der zusammengespleißten Fasern sehr exakt abgemessen werden müssen, um den Arbeitspunkt zu treffen. Die erreichte Performance ist deutlich besser als die anderer photodiodenbasierter Detektionschemata. Zwischen zwei unabhängigen Detektoren wurde über einen Zeitraum von 8 h die Differenz gebildet. Die Standardabweichung dieses Messwertes beträgt 23 fs, wobei die beiden Detektorsignale peak-to-peak Abweichungen von etwa 350 fs aufweisen. Unterschiedliche Driften der beiden Verstärker und starkes Messrauschen sind die überwiegenden Probleme dieses Aufbaus.

Anschließend wurde ein zweiter Anlauf mit einer neuen (Freistrah-)Optik und einer anderen Auswertung der Detektorsignale vorgenommen. Aus dem Signal der Photodiode werden zunächst mit einem breitbandigen Bandpassfilter zwei Harmonische selektiert, die dann zusammen in nur einem Verstärker verstärkt werden. Aus dem anschließend aufgesplitteten Signal können dann mit den gleichen schmalbandigen Filtern wie im ursprünglichen Aufbau jeweils einzeln die beiden zu messenden Harmonischen selektiert werden. Die eigentliche Messung erfolgt nun durch kompakte, biasfreie Leistungsdetektoren auf der Basis von Schottky-Dioden. Dieser Aufbau führte zu den in [ZAB+09] publizierten Ergebnissen. Hier konnte über 50 h eine Standardabweichung des Differenzsignals zwischen zwei Detektoren von unter 5 fs und eine peak-to-peak Abweichung von 20 fs erzielt werden. Allerdings ist auf den Messwerten weiterhin ein hohes Rauschniveau erkennbar. Abbildung 3.1 zeigt die dort aufgenommene Messung.

In Anlehnung an dieses vielversprechende Verfahren wird in den folgenden Kapitel ein neues Messprinzip entwickelt, um diese Resultate weiter zu verbessern.

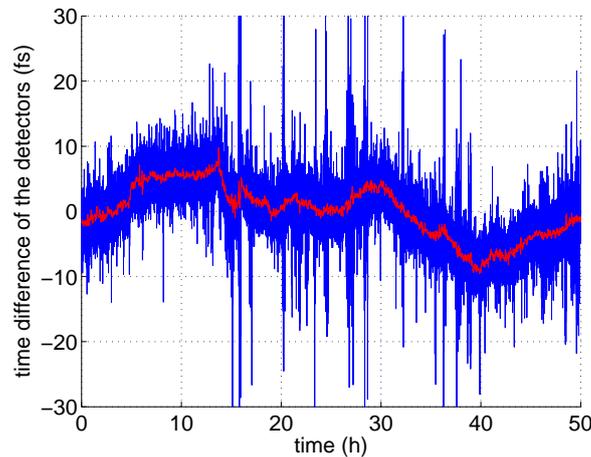


Abbildung 3.1.: Driftmessung eines alternativen photodiodenbasierten Verfahrens [aus ZAB+09]

4. Optischer Aufbau für den Detektor

In Abbildung 4.1 ist eine schematische Übersicht des optischen Aufbaus abgebildet. Dieser besteht aus Freistrah- (rot) und Faseroptik (blau und grau). Um die optischen Verluste zu minimieren, sind die Faserkomponenten zusammengespießt.

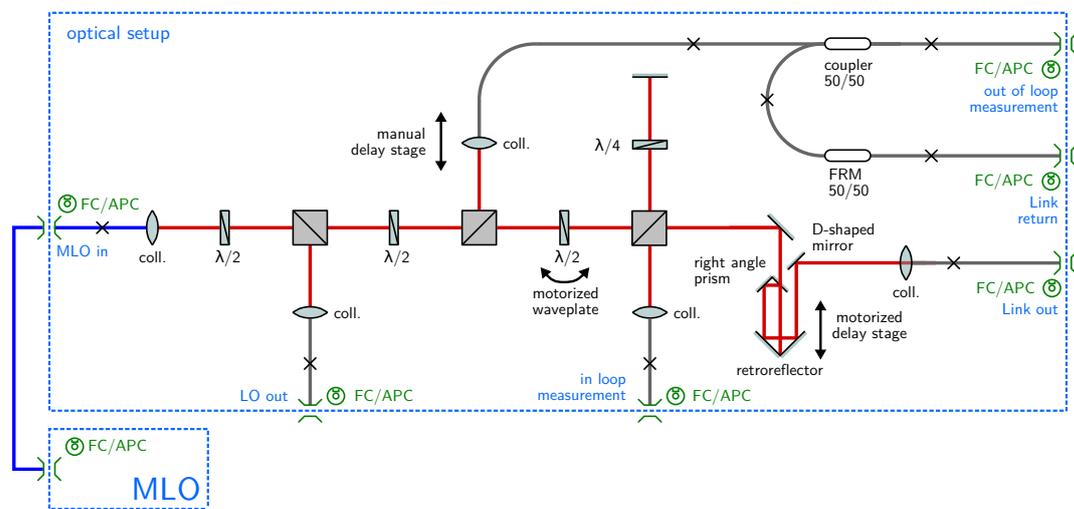


Abbildung 4.1.: Skizze des optischen Aufbaus

Zunächst wird das im Master Laser Oszillator (MLO) erzeugte Licht durch eine polarisationserhaltende Faser (blau) zum Aufbau transportiert und über einen Stecker eingespeist. Dort wird es kollimiert aus der Faser ausgekoppelt und mittels polarisierenden Strahlteilern auf die folgenden Ausgänge aufgeteilt:

- LO-Erzeugung für die HF-Mischer (**LO out**, siehe Abschnitt 5.3)
- Abgriff des Messsignals (**in-loop measurement**)
- Speisung des optischen Links (**Link out**)
- Abgriff der out-of-loop-Messung (**out-of-loop measurement**)

Das Linkende wird mit dem **Link return**-Port verbunden. Die Wellenplatten ($\lambda/2$) vor den Strahlteilern dienen jeweils der Einstellung des Aufteilungsverhältnisses und somit der optischen Leistung auf dem jeweiligen Port. Das Teilungsverhältnis zwischen Linkpulsen und Referenzpulsen kann mit einer motorisierten Wellenplatte auch elektronisch

eingestellt werden. Die Freistrahlsignale werden über Kollimatoren wieder in Fasern eingekoppelt und über optische Verbinder aus dem Aufbau geführt. Im Freistrahlteil befindet sich auch die optische Verzögerungsstrecke zur langsamen Nachführung des Timings der Lichtpulse, die in den Link eingekoppelt werden.

Die out-of-loop Messung wird nur für den Testaufbau, nicht aber in der realen Anwendung benötigt, da sie lediglich zur Verifikation der erzielten Stabilität verwendet wird.

Das am Linkende im Faraday Rotating Mirror (FRM) reflektierte Licht durchläuft erneut den Link und wird dann am rechten polarisierenden Strahlteiler mit dem Referenzpuls kombiniert und auf den Messausgang geführt. Das Licht hinter dem Faraday Rotating Mirror am Linkende wird über einen 50/50 Koppler mit dem zuvor am mittleren Strahlteiler erzeugten Referenzpulszug für die out-of-loop Messung kombiniert und anschließend ebenfalls auf den entsprechenden Messausgang geführt. Der Kollimator für die out-of-loop Messung ist auf einem per Mikrometerschraube verstellbaren Schlitten montiert, um für die out-of-loop Messung einen separaten Arbeitspunkt einstellen zu können. Eine genaue Beschreibung des Messprinzips und der erzeugten Signale ist in Kapitel 5 zu finden.

4.1. Wichtige Optik-Komponenten

In den folgenden Abschnitten werden kurz die wichtigsten Eigenschaften der verwendeten Optikelemente und ihre Funktion skizziert.

$\lambda/2$ und $\lambda/4$ Wellenplatten

Bei sogenannten Wellen- oder Verzögerungsplatten handelt es sich um anisotrope optische Kristalle, die für unterschiedliche Polarisationsrichtungen verschiedene Brechungsindizes aufweisen. Wenn die Dicke dieser Kristalle in einem bestimmten Verhältnis zur Wellenlänge des verwendeten Lichts steht, kann dessen Polarisation gezielt beeinflusst werden. Die Brechungsindizes entlang der beiden Achsen werden entsprechend der Ausbreitungsgeschwindigkeit des Lichts mit n_{slow} und n_{fast} bezeichnet. Die Phasenverschiebung $\Delta\varphi$ zwischen den beiden Polarisationsrichtungen lässt sich abhängig von der Wellenlänge λ und der Dicke d der Wellenplatte wie folgt berechnen [vgl. ST91, S. 232]

$$\Delta\varphi = \frac{2\pi}{\lambda} \cdot d \cdot (n_{slow} - n_{fast}). \quad (4.1)$$

Wenn die Phasenverschiebung genau die halbe oder viertel Wellenlänge des Lichts beträgt, spricht man von $\lambda/2$ oder $\lambda/4$ Wellenplatten. Diese Wellenplatten haben spezielle Eigenschaften bezüglich des verwendeten Lichts (siehe auch Abbildung 4.2).

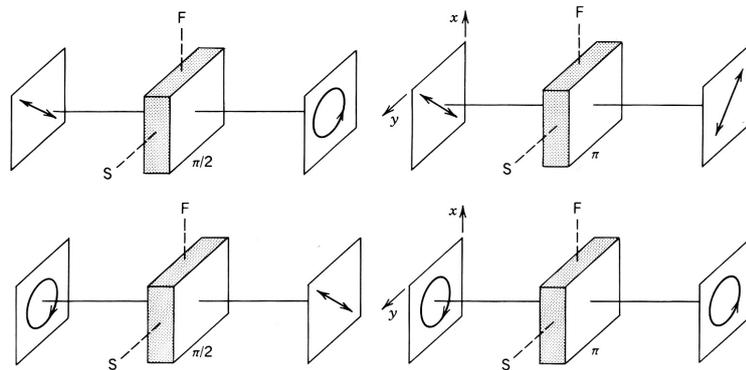


Abbildung 4.2.: Polarisation an $\lambda/4$ und $\lambda/2$ Wellenplatten (hier im Bezug auf die Wellenlänge als $\pi/2$ und π bezeichnet) [aus ST91, S. 201]

Von besonderer Relevanz ist die Möglichkeit bei bereits linear polarisiertem Licht durch die Rotation einer $\lambda/2$ Wellenplatte dessen Polarisationsrichtung zu drehen (Abbildung 4.2, rechts oben).

Liegt zirkular oder elliptisch polarisiertes Licht vor, so kann es in linear polarisiertes Licht transformiert werden, wenn es ein im richtigen Winkel zur Polarisation ausgerichtetes $\lambda/4$ Plättchen durchläuft (Abbildung 4.2, links unten).

Polarisierende Strahlteiler

Die verwendeten polarisierenden Strahlteiler (Beamcubes) teilen das einfallende Licht anhand seiner Polarisation in eine senkrechte (I_s) und eine parallele Komponente (I_p) auf. Abbildung 4.3 zeigt einen solchen Strahlteiler. Die Grenzfläche zwischen

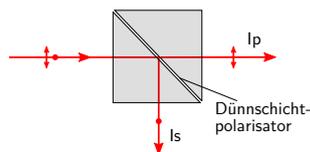


Abbildung 4.3.: Polarisierender Strahlteiler [vgl. Dem09, S. 255]

den Prismen ist meist mit einem sogenannten Dünnschichtpolarisator beschichtet, der polarisationsabhängige Reflektions- und Transmissionseigenschaften aufweist. Unabhängig davon, wie das Licht vor dem Durchgang durch den polarisierenden Strahlteiler polarisiert ist, erhält man an seinen Austrittsflächen linear polarisiertes Licht. Das Teilungsverhältnis kann über die Polarisation eingestellt werden.[vgl. Dem09, S. 255 f.]

Faserkomponenten

Um die Laserpulse im optischen Synchronisationssystem zu übertragen werden ausschließlich Singlemode-Fasern verwendet. Durch den deutlich geringeren Kerndurchmesser von Singlemode- gegenüber Multimodefasern kann sich nur die Grundmode entlang der Faser ausbreiten. Die Ausbreitungsgeschwindigkeit und der zurückgelegte Weg sind für alle Pulse gleich.

In Multimodefaser hingegen wird das Licht durch einen deutlich dickeren Kern geführt. Am Rand der Faser wird durch eine Änderung des Brechungsindex erreicht, dass das Licht den Faserkern nicht verlassen kann sondern zurückreflektiert (Stufenindexfaser) oder zur Kernmitte hin abgelenkt wird (Gradientenindexfaser). Dadurch ergeben sich für die verschiedenen Lasermoden in Gradientenindexfaser unterschiedliche Ausbreitungswege und in der Stufenindexfaser zusätzlich eine über den Weg nicht konstante Ausbreitungsgeschwindigkeit. Dieser Effekt führt dazu, dass die verschiedenen Moden das Faserende zu unterschiedlichen Zeitpunkten erreichen und so der Laserpuls verbreitert wird. Die Modendispersion kann durch die Verwendung von Singlemodefaser vermieden werden. [vgl. ST91, S. 299 f.]

Die verwendete Glasfaser vom Typ SMF-28 ist eine Singlemodefaser für Laserlicht mit einer Wellenlänge von 1550 nm. Um trotz des geringen Kerndurchmessers von etwa 9 μm in diese Faser effizient Licht einzukoppeln, werden sogenannte Kollimatoren verwendet. Die sauber abgeschnittene Faser ist hier in einem kleinen zylindrischen Gehäuse mit einer, in diesem Fall, asphärischen Linse fixiert. Ein Datenblatt der verwendeten Kollimatoren ist unter [Oz05] zu finden.

Bei bestimmten Anforderungen an die Faser werden Spezialfasern verwendet, die hier nur kurz erwähnt werden sollen.

Polarisationserhaltende Faser (PMF) Durch einen speziellen Aufbau der Faser kann in die richtige Achse eingekoppeltes, linear polarisiertes Licht ohne zusätzliche Polarisationsänderung transportiert werden.

Phasenstabilisierte Faser (PSOF) Diese Faser wird mit einem Polymer mit negativem Temperaturkoeffizienten beschichtet, was in einem bestimmten Temperaturbereich zu einer, im Vergleich zu normaler Singlemodefaser, deutlich kleineren Temperaturempfindlichkeit der Faser führt.

Dispersionskompensierende Faser (DCF) Mit dieser Faser können die Auswirkungen der chromatischen Dispersion, die in gewöhnlicher SMF-28 auftritt, kompensiert werden. Sie muss um den gewünschten Effekt zu erzielen in der richtigen Länge in den Übertragungsweg eingefügt werden. Da diese Art von Faser schwierig zu spleißen ist, ist eine Dispersionskompensation immer auch mit einer relativ großen optischen Dämpfung verbunden.

Fasergebundene Splitter oder Koppler sind Komponenten zur Aufspaltung oder Zusammenführung von Laserlicht in einem festgelegten Verhältnis. Der hier verwendete 50/50 Koppler kombiniert die beiden Laserpulse zur out-of-loop Messung zu gleichen Teilen.

Um die Verluste möglichst gering und die Fasern kurz zu halten, wurden die meisten Faserkomponenten passend zurechtgeschnitten und gespleißt. Dazu werden die exakt rechtwinklig abgeschnittenen (gecleavten) Fasern in ein Spleißgerät eingelegt, genau aufeinander ausgerichtet und zusammengeschoben. Ein kurzer Lichtbogen verschweißt die beiden Faserenden. Der bruchempfindliche Spleiß muss anschließend durch einen Spleißprotector geschützt werden.

Strahlparameter

Die vereinfachte Betrachtung der Strahlausbreitung in der geometrischen Optik ist unter Berücksichtigung der Wellennatur des Lichts nicht mehr gültig. Stattdessen kann die Gaußsche Strahlenoptik zur Beschreibung der Lichtausbreitung unter Berücksichtigung der Wellenoptik verwendet werden. Der Strahlradius bleibt hier über große Entfernungen nicht konstant, sondern weist ein durch eine Gaußkurve beschriebenes transversales Profil auf.

Mit Radius wird hier der Abstand bezeichnet, an dem die radiale Intensität auf $1/e^2$ gegenüber dem Strahlzentrum abgefallen ist. In Abbildung 4.4 ist ein solches gaußsches Profil zu sehen. W_0 bezeichnet den kleinsten Strahlradius an der sogenannten

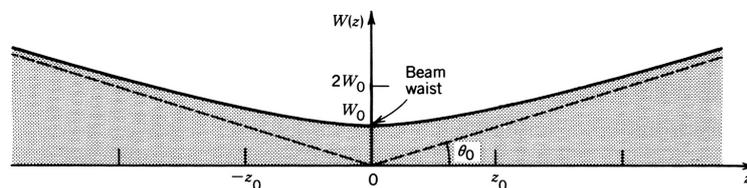


Abbildung 4.4.: Strahlparameter [aus ST91, S. 86]

Strahltaile. Der Winkel Θ_0 beschreibt die Divergenz des Strahls für große z , bei denen der Strahldurchmesser sich näherungsweise linear mit der Ausbreitung vergrößert. Die Formel für die Divergenz lautet dann,

$$\Theta_0 = \frac{2}{\pi} \cdot \frac{\lambda}{2 \cdot W_0} \quad (4.2)$$

wobei λ die Wellenlänge des Lichts bezeichnet. Man kann erkennen, dass die Divergenz wächst, wenn die Strahltaile verkleinert wird. Das bedeutet im Umkehrschluss, je kleiner der Strahldurchmesser gewählt wird, desto stärker weitet sich der Strahl für große Entfernungen auf. [vgl. ST91, S. 83 ff.]

4. Optischer Aufbau für den Detektor

Der sogenannte „full divergence angle“ entspricht der zweifachen Divergenz. Mit Hilfe des Datenblattes zu den hier verwendeten Kollimatoren [siehe Oz05] und der dort genannten Formel kann die Divergenz aus der Brennweite f der integrierten Linse (in mm) und dem Durchmesser a der verwendeten Glasfaser (in μm) wie folgt berechnet werden.

$$\Theta_0 = \frac{a}{2 \cdot f} = \frac{9 \mu\text{m}}{2 \cdot 3,9 \text{ mm}} = 1,15 \text{ mrad} \quad (4.3)$$

Der Strahldurchmesser direkt nach dem Kollimator beträgt laut Datenblatt 0,9 mm. Man erhält nach einer Wegstrecke von beispielsweise 0,5 m mit den gegebenen Daten einen Strahl von etwa 2 mm Durchmesser, der nur mit Verlusten erneut in einen Kollimator eingekoppelt werden kann.

Faraday Rotating Mirror (FRM)

Der Faraday Effekt beschreibt die Eigenschaft von Materialien die Polarisation von durchlaufendem Licht auf Grund eines äußeren Magnetfeldes zu drehen. Der Parameter $\rho = \theta/l$ beschreibt den Rotationswinkel der Polarisation pro Längeneinheit. Er ist über die Verdet-Konstante proportional zur Stärke des angelegten Magnetfeldes B . [vgl. ST91, S. 225 f.]

$$\rho = V \cdot B \quad (4.4)$$

Die Verdet-Konstante wiederum bezeichnet eine von der Wellenlänge abhängige Materialkonstante. In Gleichung 4.5 wird die sogenannte Becquerel Formel zur Berechnung der Verdet-Konstante angegeben. [vgl. Dam04, S. 133 f.]

$$V = -\frac{e \cdot \lambda}{2 \cdot m \cdot c} \left(\frac{dn}{d\lambda} \right) \quad (4.5)$$

Die Naturkonstanten e , m und c bezeichnen wie üblich die Elementarladung, Elektronenmasse und die Lichtgeschwindigkeit. λ steht für die Wellenlänge des Lichts, für die der Effekt berechnet werden soll, und $dn/d\lambda$ für die Abhängigkeit des Brechungsindex von der Wellenlänge, also die Dispersion.

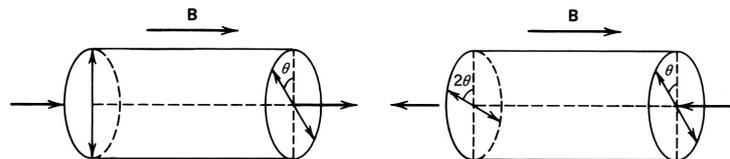


Abbildung 4.5.: Polarisationsdrehung durch den Faraday-Effekt [aus ST91, S. 226]

Wichtig ist, dass die Richtung der Polarisationsdrehung unabhängig von der Ausbreitungsrichtung des Lichts ist und stattdessen von der Richtung des angelegten Magnetfeldes abhängt (siehe Abbildung 4.5).

Bei dem im Folgenden als Linkende verwendeten, fasergekoppelten Faraday Rotating Mirror (FRM) wird das Magnetfeld durch einen integrierten Permanentmagneten erzeugt. Hinter dem Faraday-Rotator ist ein halb durchlässiger Spiegel integriert, der die Hälfte des Laserlichts als Nutzsignal durchlässt und die andere Hälfte als Messsignal zurück zum Linkanfang reflektiert. Das reflektierte Licht durchläuft dabei ein weiteres Mal den Faraday Rotator. Dieser ist so dimensioniert, dass das Licht bei der Nennwellenlänge insgesamt um 90 Grad gedreht wird.

4.2. Die optische Delayline

Als langsames Stellglied in Faserlinks und anderen optischen Anwendungen werden üblicherweise Delaylines eingesetzt. Neben der Forderung nach einem großen Hub ist die wichtigste Anforderung an solche Delaylines, mit möglichst geringen, am Besten über den Stellbereich konstanten, optischen Verlusten zu operieren.

Die einfachste Konstruktion einer Delayline besteht aus einem auf einem motorisierten Schlitten befestigten und einem festen Kollimator. Durch die Position des Schlittens kann der Abstand und somit auch die Laufzeit variiert werden. Diese Art der Verzögerungsstrecke ist allerdings auf extrem präzise Schlitten angewiesen, da die Einkopplung des Laserstrahls in den Kollimator sehr empfindlich auf Positions- und Winkelfehler reagiert. Außerdem muss auf eine günstige Führung der an den Kollimator anschließenden Glasfaser geachtet werden, um von der Position des Schlittens abhängige Biegeverluste zu vermeiden. Darüber hinaus verschlechtert sich die Einkopplung bei größer werdendem Abstand zwischen zwei Kollimatoren durch die Strahldivergenz. Dieses prinzipielle Problem kann durch den Einbau von Linsen mit entsprechenden Abbildungseigenschaften oder Teleskopen, also Systemen aus mehreren Linsen, minimiert werden.

Bei komplexeren Schlitten wird üblicherweise mit einem bewegten Planspiegel gearbeitet. Diese Konstruktion hat den Vorteil, dass die transversale Position des Spiegels zum Strahl keine Rolle spielt, solange Strahl und Spiegel genau senkrecht aufeinander stehen. Winkelfehler führen jedoch zu einem, von der Position des Schlittens abhängigen, unerwünschten Strahlversatz. Allerdings wird bei der Nutzung eines Planspiegels die nutzbare Strecke im Vergleich zum einfachen Schlitten mit Kollimator verdoppelt. Einfallender und reflektierter Strahl können mit Hilfe von Wellenplatten und polarisierenden Strahlteilern separiert werden. Eine solche Delayline kann nur von einem Strahl durchlaufen werden, da die Laserstrahlen sonst nicht mehr getrennt werden können. Für das in dieser Arbeit vorgestellte Schema zur Stabilisierung eines Faserlinks ist es jedoch erforderlich, den hinlaufenden und den rücklaufenden Puls gemeinsam zu verzögern um so das Timing am Linkende stabil zu halten. Die vom Detektor gemessene Timingänderung setzt sich ebenfalls aus diesen beiden Anteilen zusammen.

4. Optischer Aufbau für den Detektor

Um das zu erreichen, wird im Folgenden eine Delayline mit Retroreflektor entwickelt und vorgestellt.

4.2.1. Der Linearschlitten

Die Winkeltoleranzen kommerziell erhältlicher Motorschlitten mit den benötigten Stellwegen von etwa 60 mm betragen üblicherweise wenige hundert μrad [vgl. PI08]. Toleranzen in dieser Größenordnung verursachen, unter Verwendung eines Planspiegels, Schwankungen in der Einkoppeleffizienz von mehr als 20 % (siehe Abbildung 4.8). Schlitten mit dem geforderten Stellweg sind bei diesen vorgegebenen Toleranzen zudem meist Sonderanfertigungen und nicht im Sortiment der Hersteller.

In der industrialisierten Version der auf optischer Kreuzkorrelation (OXC) basierenden Faserlinkstabilisierungseinheiten werden bei DESY entwickelte Linearschlitten verwendet. Diese Schlitten weisen ähnliche Toleranzen auf, sind jedoch in der Herstellung preisgünstiger. Für die hier vorgestellte Delayline kommt eine Sonderanfertigung des Herstellers PI, mit einem Fahrweg von 56 mm, zum Einsatz. Angetrieben wird der Schlitten von einem Schrittmotor. Der Stellbereich des Motors beträgt 4.896.569 Schritte, was zu einer Schrittgröße von theoretisch 11,44 nm führt. Hier ist allerdings zu beachten, dass der Schlitten eine Hysterese von etwa 200 Schritten aufweist, was ungefähr $2,3 \mu\text{m}$ entspricht. Diese Hysterese wird vermutlich durch Fertigungstoleranzen bei der Antriebsspindel verursacht. Abbildung 4.6 zeigt den verwendeten Schlitten mit der bereits montierten Halterung für den Retroreflektor. Die Ansteuerung des

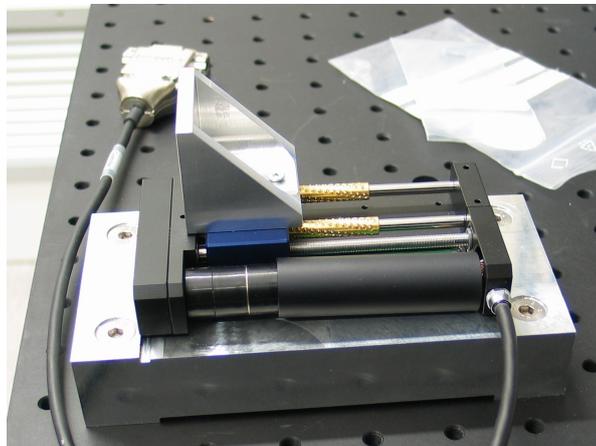


Abbildung 4.6.: Linearschlitten mit Halterung für den Retroreflektor

Schlittens erfolgt über eine SPS mit Schrittmotorklemme, die in das Beschleunigerkontrollsystem eingebunden ist und somit die Voraussetzung für das in Abschnitt 7.3 erwähnte langsame Feedback bildet.

Der PI Schlitten besitzt integrierte Hall-Endschalter, die mit einer Spannung von 5 V betrieben werden müssen. Um diese Endschalter an der Schrittmotorklemme zu verwenden, die für mechanische Endschalter ausgelegt ist und diese mit 12 V beaufschlagt, wird eine externe Schaltung zur Pegelanpassung verwendet. Diese Schaltung wird zwischen Motorklemme und Schrittmotor gesteckt. Ein Festspannungsregler erzeugt 5 V Versorgungsspannung für die Hall-Endschalter. Diese schalten zunächst Bipolartransistoren, die wiederum an die Eingänge der Motorklemme angeschlossen sind.

4.2.2. Simulation der Delayline

Im Rahmen dieser Diplomarbeit wird ein neues Design für eine optische Delayline entworfen, das einige der zuvor genannten Probleme umgeht. Die initiale Idee hinter dieser Neuentwicklung ist die Verwendung eines sogenannten Retroreflektors. Es handelt sich hierbei um eine Optikkomponente mit drei Spiegelflächen, die exakt senkrecht zueinander ausgerichtet sind. Diese Kombination aus Spiegeln hat gegenüber einem Planspiegel den großen Vorteil, dass Licht unabhängig vom Einfallswinkel immer parallel zum einfallenden Strahl reflektiert wird. Diese Reflektion setzt sich aus drei Einzelreflektionen an den Spiegelflächen zusammen.

Zur Abwägung der Vor- und Nachteile von verschiedenen Kombinationen aus Planspiegeln, Prismen und Retroreflektoren wird eine Simulation dieser Komponenten in Matlab implementiert, die sich die Möglichkeiten der objektorientierten Programmierung zu Nutze macht. Diese Implementierung beinhaltet drei Klassen zur Simulation von Laserstrahlen, Spiegeln und Retroreflektoren. Prismen müssen derzeit noch aus einzelnen Spiegeln zusammengesetzt werden. Objekte dieser Klassen können mit sehr wenigen Matlab-Befehlen erzeugt, manipuliert und dreidimensional geplottet werden. Der Quellcode zu den einzelnen Klassen und das Script zur Simulation sind in Anhang B zu finden. Eine ausführbare Version liegt im digitalen Anhang bei, weitere Information hierzu in Anhang A.

Jede der Klassen besitzt ein eigenes Interface. Die entsprechende Schnittstelle wird im Folgenden vorgestellt. Wenn von Vektoren gesprochen wird, sind immer Spaltenvektoren mit drei Elementen gemeint, die Punkte oder Richtungen in einem kartesischen Koordinatensystem bezeichnen. Alle Elemente werden innerhalb eines globalen kartesischen Koordinatensystems positioniert.

Die Klasse *beam*

Die Klasse *beam* beschreibt einen Laserstrahl. Sie verwaltet intern einen Stütz- und einen Richtungsvektor, jedoch keinen Start- oder Endpunkt. Laserstrahlen sind in diesem Modell zunächst unendlich ausgedehnt, wobei der Richtungsvektor die positive Ausbreitungsrichtung vorgibt. Φ bezeichnet den Winkel zwischen dem Richtungsvektor und dem positiven Teil der X-Achse, Θ den Winkel zwischen dem positiven Teil der

4. Optischer Aufbau für den Detektor

Z-Achse und dem Richtungsvektor. Objekte vom Typ *beam* unterstützen die folgenden Methoden:

beam()

Der Konstruktor erwartet als Argument entweder zwei Vektoren, also Stütz- und Richtungsvektor oder ein Objekt vom Typ *beam*. Wird er ohne oder mit falschen Argumenten aufgerufen, wird ein Objekt mit leeren Vektoren erzeugt.

beam.display()

erzeugt eine textuelle Ausgabe der momentanen Strahlparameter

beam.is_valid()

liefert **true** wenn Stütz- und Richtungsvektor gesetzt und keine Nullvektoren sind, sonst **false**

beam.set_t(vec), beam.set_base(vec)

setzt einen neuen Stützvektor

beam.set_base_rel(vec)

setzt einen neuen Stützvektor relativ zum alten

beam.set_v(vec), beam.set_dir_kart(vec)

setzt einen neuen Richtungsvektor in kartesischen Koordinaten

beam.set_dir_kart_rel(vec)

setzt einen neuen Richtungsvektor in kartesischen Koordinaten relativ zum alten

beam.set_dir_spher(phi, theta)

setzt einen neuen Richtungsvektor in Kugelkoordinaten

beam.set_dir_spher_rel(phi, theta)

setzt einen neuen Richtungsvektor in Kugelkoordinaten relativ zum alten

Die Klasse *mirror*

Die Klasse *mirror* beschreibt einen einfachen Planspiegel anhand eines Stütz- und zweier Spannvektoren. Der Spiegel ist in seiner Ausdehnung im Raum nicht beschränkt. Es werden die folgenden Methoden unterstützt:

mirror()

Der Konstruktor erwartet als Argument entweder drei Vektoren, also einen Stütz- und zwei Spannvektoren, die den Spiegel aufspannen oder ein Objekt vom Typ *mirror*. Wird er ohne oder mit falschen Argumenten aufgerufen, wird ein Objekt mit leeren Vektoren erzeugt.

mirror.display()

erzeugt eine textuelle Ausgabe der momentanen Parameter

mirror.is_valid()
 liefert **true** wenn Stütz- und Spannvektoren gesetzt und keine Nullvektoren sind,
 sonst **false**

mirror.set_t(vec), mirror.set_base(vec)
 setzt einen neuen Stützvektor

mirror.set_base_rel(vec)
 setzt einen neuen Stützvektor relativ zum alten

mirror.set_v1(vec), mirror.set_v2(vec)
 setzt neue Spannvektoren in kartesischen Koordinaten

mirror.set_beam_in(beam)
 weist dem einfallenden Strahl ein Objekt vom Typ *beam* zu

mirror.update_all()
 berechnet den Schnittpunkt des Spiegels mit dem einfallenden Strahl, den Normalenvektor auf dem Spiegel, den reflektierten Strahl und ob der Schnittpunkt ein positives Vielfaches der Richtungsvektoren ist

Nach ihrer Berechnung mittels **mirror.update_all()** kann auf alle vier Ausgangsgrößen direkt zugegriffen werden. Es handelt sich hier um die Eigenschaften:

mirror.normal
 Normalenvektor auf dem Spiegel

mirror.intersection
 Koordinaten des Schnittpunktes des Laserstrahls mit der Spiegelfläche

mirror.is_positive
 ist true, wenn der Schnittpunkt ein positives Vielfaches der Spannvektoren ist,
 sonst false

mirror.beam_out
 beinhaltet den reflektierten Strahl in Form eines Objekts vom Typ *beam*

Alle Ausgangsgrößen werden gelöscht wenn eine Eingangsgröße, beispielsweise ein Spannvektor, geändert wird. Sie müssen dann mittels **mirror.update_all()** neu berechnet werden.

Die Klasse *retroreflector*

Die Klasse *retroreflector* ist die umfangreichste der entwickelten Klassen. Sie beschreibt einen Retroreflektor im dreidimensionalen Raum anhand eines Stützvektors, zweier Winkel, die angeben, wie die optische Achse des Retroreflektors ausgerichtet ist, eines Winkels, der die Rotation des Retroreflektors um seine optische Achse angibt, und seines Radius. Ψ bezeichnet den Rotationswinkel des Retroreflektors um sich selbst, Φ

4. Optischer Aufbau für den Detektor

den Winkel zwischen dem Richtungsvektor und dem positiven Teil der X-Achse und Θ den Winkel zwischen dem positiven Teil der Z-Achse und dem Richtungsvektor. Das Interface gestaltet sich wie folgt:

retroreflector()

Der Konstruktor erwartet als Argument entweder den Stützvektor, den Radius und die drei Winkel, die die Position im Raum festlegen oder ein Objekt vom Typ *retroreflector*. Wird er ohne oder mit falschen Argumenten aufgerufen, wird ein Objekt mit leeren Parametern erzeugt.

retroreflector.display()

erzeugt eine textuelle Ausgabe der momentanen Parameter

retroreflector.set_t(vec), retroreflector.set_base(vec)

setzt einen neuen Stützvektor

retroreflector.set_base_rel(vec)

setzt einen neuen Stützvektor relativ zum alten

retroreflector.set_r(radius)

setzt den Radius

retroreflector.rotate_absolute(psi, phi, theta)

setzt einen neuen Richtungsvektor in Kugelkoordinaten

retroreflector.rotate_relative(psi, phi, theta)

dreht den Richtungsvektor relativ zum alten

retroreflector.set_beam_in(beam)

weist dem einfallenden Strahl ein Objekt vom Typ *beam* zu

retroreflector.update_all()

berechnet die Schnittpunkte des Retroreflektors mit dem einfallenden Strahl, die Normalenvektoren auf den drei Spiegeflächen und den reflektierten Strahl

retroreflector.draw()

plottet den Retroreflektor

retroreflector.draw_normal_vectors(bool)

setzt eine boolesche Variable, die angibt, ob beim Plotten die Normalenvektoren gezeichnet werden oder nicht

Der Retroreflektor wird durch den Radius in seiner Ausdehnung begrenzt. Gibt es zum Beispiel keinen Schnittpunkt zwischen einfallendem Strahl und Retroreflektor, so bleibt die Liste der Schnittpunkte leer. Der ausgehende Strahl entspricht in diesem Fall dem einfallenden. Auch für den Retroreflektor gilt, was bereits über den Spiegel gesagt wurde: Die Ausgangsgrößen werden bei der Veränderung einer Eingangsgröße zurückgesetzt und müssen dann erneut durch den Befehl **retroreflector.update_all()** berechnet werden. Die Klasse hat die folgenden beiden Ausgangsgrößen:

retroreflector.intersections

Koordinaten der Schnittpunkte des Laserstrahls mit der Spiegelfläche

retroreflector.beam_out

beinhaltet den reflektierten Strahl in Form eines Objekts vom Typ *beam*

Die Simulation

Die in den vorhergehenden Abschnitten vorgestellten Klassen ermöglichen das Erstellen einer Simulation mit wenigen Befehlen. Der in Quellcode 4.1 gezeigte Ausschnitt verdeutlicht diesen Sachverhalt. Den meisten Raum nimmt die Definition der Winkel und Vektoren ein.

```

1 % parameters for beam
2 t_beam = [0;0;70.5]; % base
3 v_beam = [1;0;0];   % initial direction
4 theta_beam = 0;     % up/down
5 phi_beam = 0;       % left/right
6
7 % parameters for retroreflector
8 t_retro = [85;0;65]; % base 35 - 95
9 r_retro = 12.7;      % radius of reflector
10 psi_retro = 90;     % rotate
11 theta_retro = 0;   % up/down
12 phi_retro = 180;   % left/right
13
14 % incoming beam
15 beam_in = beam(t_beam, v_beam);
16 beam_in.set_dir_spher_rel(phi_beam, theta_beam);
17
18 % reflection at retroreflector_1
19 reflector_1 = retroreflector(t_retro, r_retro, psi_retro, theta_retro, phi_retro  ↔
    );
20 reflector_1.set_beam_in(bean_in);
21 reflector_1.update_all;
22
23 beam_intermediate_1 = reflector_1.beam_out;
24
25 % draw retroreflector
26 reflector_1.draw_normal_vectors(true);
27 reflector_1.draw;

```

Quellcode 4.1: Ausschnitt aus dem Simulationsscript

Die Simulation erlaubt nicht nur die optische Kontrolle des Strahlverlaufs, sondern sie ermöglicht auch die Bestimmung seiner exakten Position und die der einzelnen Elemente wie Retroreflektor und Prisma. Die Ausgabe des Simulationsscripts ist in Abbildung 4.7 zu sehen. Die Angaben im Script sind maßstabsgerecht wenn man alle

Längen in der gleichen Einheit (hier mm) eingibt. Diese Informationen können genutzt werden um Halterungen für die entsprechenden Elemente zu konstruieren. Einfallender und reflektierter Strahl sind im vorliegenden Beispiel nur um 6 mm separiert, was eine Konstruktion auf engstem Raum erfordert.

Außerdem wird im vorliegenden Script zur Beurteilung der Fehlertoleranz des Aufbaus ein Kollimator simuliert. Der Benutzer kann den Kollimator an der von ihm gewünschten Stelle positionieren und erhält dann auf der Kommandozeile Informationen darüber, in welchem Winkel und mit welchem transversalen Versatz der Strahl auf den Kollimator auftrifft. So können schnell verschiedene Winkel und Positionsabweichungen des Retroreflektors bei verschiedenen Schlittenpositionen getestet werden und man erhält bei jeder Simulation direkt Angaben über den entstandenen Fehler am Kollimator. Auf Basis dieser Simulation werden die im nächsten Abschnitt geschilderten Entscheidungen getroffen.

4.2.3. Delayline auf Basis eines Retroreflektors

Der hier vorgestellte Entwurf wurde aus einer Vielzahl von Möglichkeiten ausgewählt. Zur Reflektion der Laserpulse auf dem motorisierten Schlitten wird nun ein sogenannter replizierter, metallischer Retroreflektor verwendet. Die maximale Ungenauigkeit des reflektierten Strahls beträgt für den verwendeten Retroreflektor zwei Bogensekunden. Sie setzt sich aus den Winkeltoleranzen der drei Spiegelflächen zusammen. Die Winkeltoleranzen des Motorschlittens haben somit keinen Einfluss mehr auf die Reflektionswinkel des Laserlichts.

Der Retroreflektor hat allerdings gegenüber dem Planspiegel den Nachteil, sensitiv gegenüber transversalen Positionsfehlern zu sein. Eine Verschiebung des Retroreflektors senkrecht zum Laserstrahl verursacht eine Parallelverschiebung des reflektierten Strahls um den doppelten Betrag in die gleiche Richtung. Um diesen Effekt zumindest in vertikaler Richtung zu kompensieren wird der Retroreflektor so auf dem Motorschlitten montiert, dass das Licht anschließend durch ein Prisma erneut auf den Retroreflektor gelenkt und von diesem ein zweites Mal reflektiert wird. Abbildung 4.7 zeigt die Simulation dieses Strahlverlaufs, ein Foto der aufgebauten Delayline zusammen mit der übrigen Optik ist in Abbildung 4.10 zu finden.

Das Prisma wird hier in einer festen Position montiert, während der Retroreflektor auf dem Schlitten befestigt wird. Es ist deutlich zu erkennen, dass einfallender und reflektierter Strahl räumlich getrennt sind. Die Delayline kann also einfach vor dem Kollimator der Linkfaser in den Strahlverlauf eingesetzt werden. Um sicherzustellen, dass bei der Montage kein Winkelfehler zwischen Motorachse und Strahlrichtung entsteht, werden zur Justage Spiegel vorgesehen, mit denen der Laserstrahl exakt parallel zur Motorachse ausgerichtet wird. Ein Winkelfehler an dieser Stelle würde, wenn man den Schlitten verfährt, zu einem relativen Positionsfehler führen, der durch die Optik

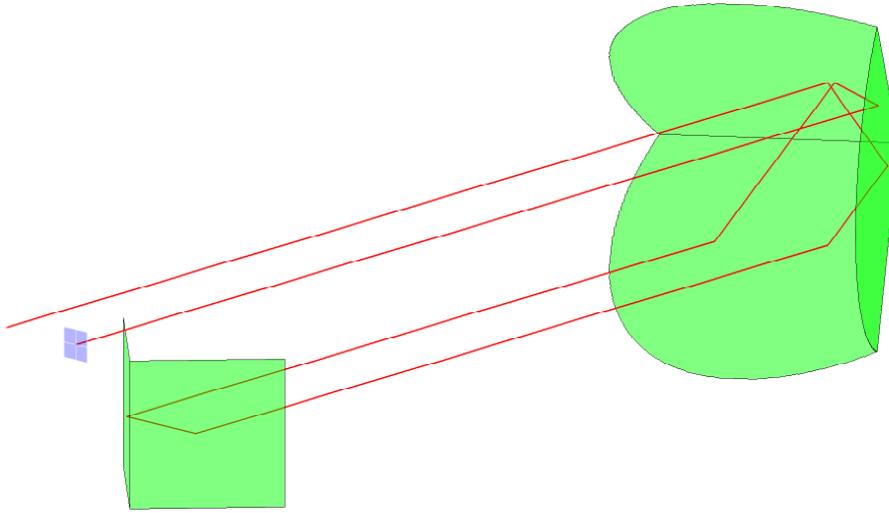


Abbildung 4.7.: Simulation der Delayline mit Retroreflektor

nicht kompensiert werden kann. Die Justage des Laserstrahls ist letztlich einfacher als eine Ausrichtung des Schlittens mit Passstiften, zumal dadurch die Möglichkeit zur Fehlerkorrektur und Nachjustage besteht. Zusätzlich wird die Halterung des Retroreflektors so entworfen, dass seine Spitze sich möglichst nah am Drehpunkt des Schlittens befindet. Dies soll eine Übersetzung von Winkelfehlern in transversale Abweichungen, die durch eine Position außerhalb des Drehpunktes begünstigt würde, minimieren.

Diese Kombination ist, wenn man die Strahlachse perfekt parallel zur Motorachse ausrichtet, nur sensitiv gegenüber transversalen Toleranzen des Schlittens in horizontaler Richtung.

4.2.4. Messergebnisse und erreichbare Performance

Um die tatsächliche Performance zu evaluieren wird die Einkoppeleffizienz nach dem Durchlaufen der Delayline untersucht. Die Leistung des in eine Faser eingekoppelten Laserstrahls wird mit einer Photodiode ausgewertet. Die zur Leistung proportionale Spannung, die durch den Photostrom über einem $50\ \Omega$ Widerstand erzeugt wird, wird zunächst mit einem Low Noise Amplifier (LNA, siehe Abschnitt 5.3) um den Faktor 10 verstärkt und anschließend mit einem Tiefpass gefiltert. Die so verstärkte Spannung wird mit einem digitalen Speicheroszilloskop aufgezeichnet, während der Schlitten mit konstanter Geschwindigkeit von der Anfangs- in die Endlage verfahren wird. Zur Auswertung wird die Zeitinformation der aufgezeichneten Daten wieder in eine

Schlittenposition umgerechnet. Die maximale gemessene Spannung wird auf 100 % normiert. Außerdem werden für Auslenkungsfenster verschiedener Größe die normierten Leistungsschwankungen relativ zum Maximalwert und zum Mittelwert dieses Fensters in einem jeweils eigenen Plot dargestellt. Abbildung 4.8 zeigt die gewonnenen Messwerte, sowie zum Vergleich die Messung einer Delayline mit Planspiegel.

Die Referenzmessung wurde während der Anfertigung dieser Diplomarbeit bei DESY aufgenommen. Sie zeigt die bisher besten, mit einem Planspiegel gewonnenen, Messwerte. Es ist deutlich zu erkennen, dass sich hier zwei verschiedene Effekte überlagern. Mechanische Spannungen in den Endlagen verursachen Einkoppelverluste, die im Wesentlichen von der Position des Schlittens abhängen. Hier wurde die Einkopplung für eine zentrale Position optimiert und sie fällt für kleinere und größere Entfernungen ab, da sich der Winkel des Spiegels relativ zum Laserstrahl mit der Position ändert. Diesem Effekt überlagert sind Oszillationen mit Verlusten bis zu 20 %, die durch Toleranzen an der Antriebsspindel und somit kleine alternierende Winkelfehler verursacht werden. Einkoppelverluste durch Divergenz spielen hier kaum eine Rolle, da bei dieser Messung bereits ein Teleskop verwendet wurde.

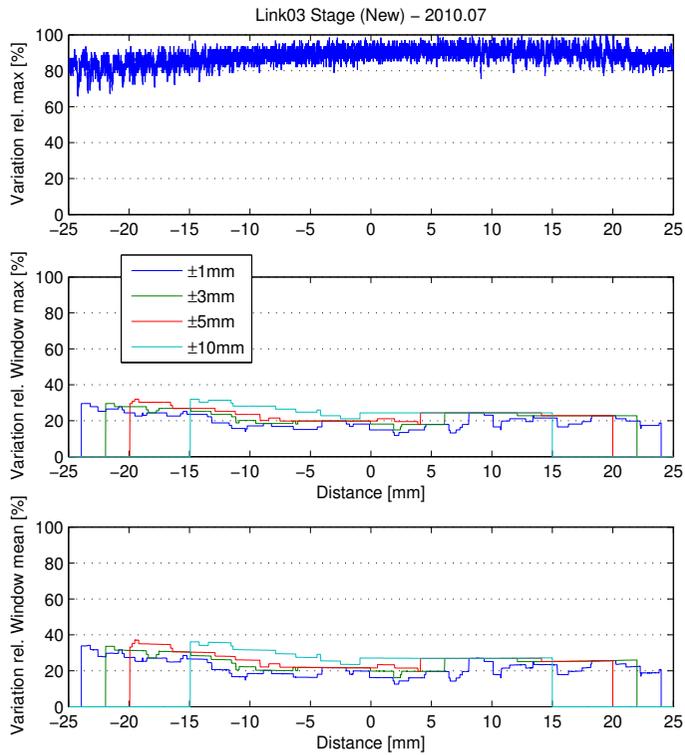
Die Konstruktion mit Retroreflektor zeigt keine der zuvor genannten Effekte, da sie unempfindlich gegenüber Winkelfehlern am motorisierten Schlitten ist. Die 25 % relative Einkopplungsverluste, die im Messergebnis zu erkennen sind, werden durch die Strahldivergenz verursacht. Hier muss beachtet werden, dass die Einkopplung gegen die Position des Schlittens geplottet wurde. Der Laserstrahl wird jedoch bei Verwendung des Retroreflektors zweifach gefaltet, was zu einer Verdopplung des Weges gegenüber der anderen Messung führt. Es ist zu erwarten, dass die Einkoppeleffizienz unter Berücksichtigung der im folgenden Abschnitt aufgelisteten Verbesserungsvorschläge von der aktuellen Schlittenposition nahezu unabhängig wird.

Abgesehen von den wegabhängigen Einkopplungsverlusten an der Delayline müssen auch konstante Reflektionsverluste berücksichtigt werden. Diese Verluste von 20 % sind im Wesentlichen auf Reflektionsverluste am Retroreflektor zurückzuführen.

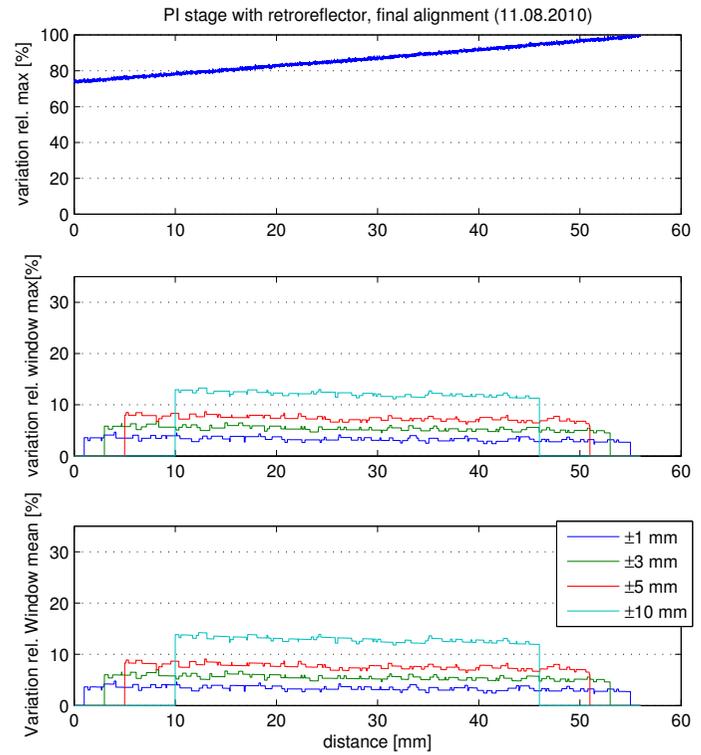
4.2.5. Verbesserungsmöglichkeiten und Anwendung

Die hier vorgestellte Delayline auf der Basis eines Retroreflektors bietet weitere Möglichkeiten zur Optimierung. Im Rahmen dieser Diplomarbeit wurde nur der vorgestellte Prototyp aufgebaut und untersucht. Für eine Serienfertigung sind zwei Verbesserungen unbedingt zu empfehlen.

Zum einen sollte der metallische Retroreflektor durch ein Tripelprisma ersetzt werden, das eine verlustärmere Reflektion ermöglicht. Bei der Laserwellenlänge von 1560 nm entstehen beispielsweise an Gold Reflektionsverluste von 1–2 %, wohingegen die Reflektionsverluste an einem Prisma mit einer schmalbandigen Antireflexbeschichtung an der vorderen Durchgangsfläche vernachlässigbar klein sind.



(a) DESY Schlitten mit Planspiegel



(b) PI Schlitten mit Retroreflektor

Abbildung 4.8.: Einkopplung bei verschiedenen Delaylines

4. Optischer Aufbau für den Detektor

Da das Laserlicht pro Durchgang durch die Delayline sechs mal im Retroreflektor gespiegelt wird, kann so die Gesamteffizienz um 5–10 % gesteigert werden. Zum anderen sollten die bereits erwähnten, wegabhängigen Verluste durch Strahldivergenz möglichst durch eine, auf den Stellbereich der Delayline angepasste, Optik kompensiert werden.

Aufgrund dieser vielversprechenden Ergebnisse ist im aktuellen optomechanischen Design der OXC basierten Faserlinkstabilisierung bereits eine Delayline auf Basis eines Retroreflektors vorgesehen und die vorgeschlagenen Verbesserungen wurden berücksichtigt. In Abbildung 4.9 ist ein CAD Modell dieser Mechanik zu sehen. Ein Prototyp befindet sich derzeit in der Produktion. Eine dauerhafte Verwendung auch für den hier vorgestellten diodenbasierten Detektor ist beabsichtigt.

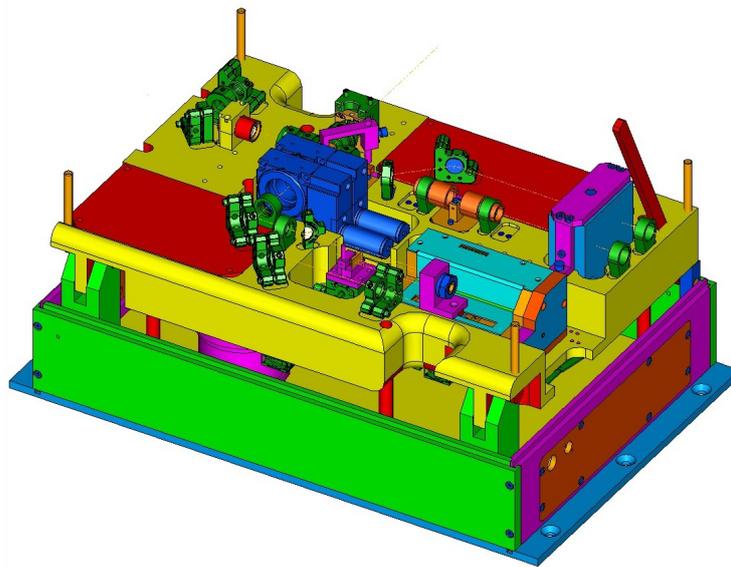


Abbildung 4.9.: CAD Modell der Mechanik für OXC basierte Faserlinks, Motorschlitten in hellblau [Grafik: DESY]

4.3. Versuchsaufbau der Optik

Alle Komponenten werden auf einer Lochrasterplatte montiert und zum Schutz mit einem Gehäuse versehen. Es wird eine Halterung entworfen, die die platzsparende Aufnahme der Strahlteiler und der Wellenplatten ermöglicht. Auch die eigens entworfene Delayline wird hier eingebaut. Die Fasern und Faserkomponenten werden alle mit Klebeband auf der Lochrasterplatte fixiert. So wird vermieden, dass die Fasern später bewegt werden. Durch die thermische Anbindung an die Platte und den optischen Tisch wird die größtmögliche passive Temperaturstabilität erreicht. Ein Foto des fertigen Aufbaus ist in Abbildung 4.10 zu sehen.

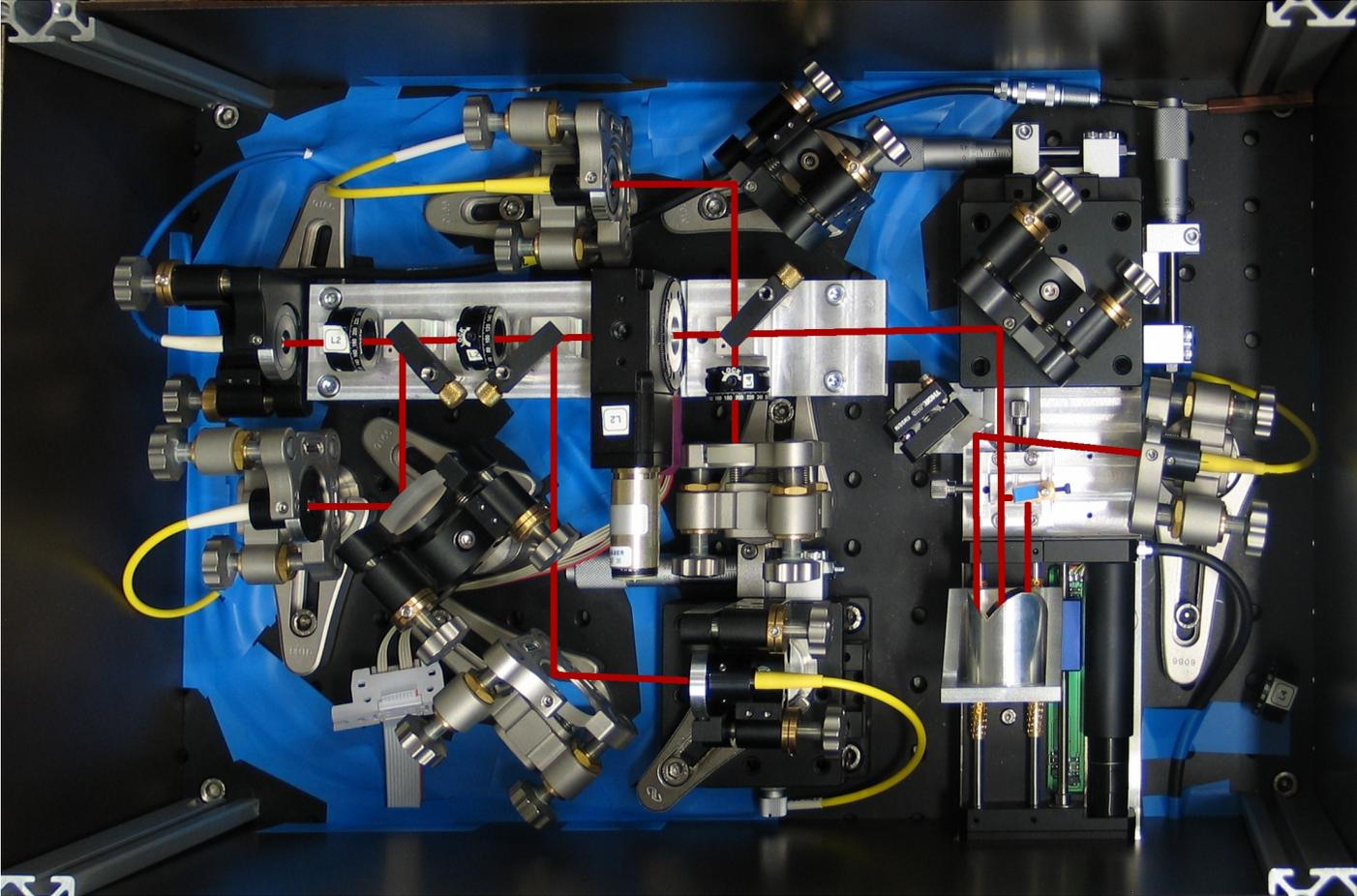


Abbildung 4.10.: Testaufbau der Optik mit eingezeichnetem Strahlverlauf

5. Aufbau des Detektors

In den bisherigen Kapiteln wurde bereits beschrieben, wie sich zwei überlagerte Pulszüge des MLO im Frequenzbereich verhalten und es wurde die zur Detektion der Laufzeitunterschiede benötigte Optik vorgestellt. Im folgenden Kapitel wird das Messprinzip erläutert und in einem Versuchsaufbau implementiert. Es wird eine Simulation für das Messprinzip entwickelt und die zur Messung verwendeten Elektronikkomponenten werden vorgestellt. Am Ende des Kapitels wird eine Einschätzung der Leistungsfähigkeit und Fehlertoleranz des vorgestellten Detektors vorgenommen.

5.1. Das Messprinzip des Detektors

Auf die Eigenschaften der Modulation im Frequenzbereich, die durch die Überlagerung zweier Laserpulszüge entsteht, wurde bereits eingegangen. Dieser modulierte Frequenzkamm wird mit Hilfe einer Photodiode (Bandbreite > 10 GHz) gemessen. Das Hochfrequenzsignal, das am Ausgang der Photodiode anliegt, enthält innerhalb ihrer Bandbreite alle spektralen Bestandteile der überlagerten Pulszüge.

Eine einzelne Frequenzharmonische wird durch die Verwendung schmalbandiger HF-Filter aus diesem Frequenzspektrum selektiert. Amplitude und Phase dieser hochfrequenten harmonischen Schwingung werden durch die Phasenverschiebung zwischen den beiden Laserpulszügen moduliert. Im perfekten Arbeitspunkt des Detektors wird die Amplitude der Harmonischen durch die Modulation zu Null.

Eine relative Phasenverschiebung der beiden Pulszüge führt zu einer Veränderung der Modulation und somit auch der Amplitude der betrachteten Harmonischen. Auf diese Weise wird die zu detektierende Phaseninformation in eine Amplitudeninformation umgewandelt. Diese Amplitudenänderung in Abhängigkeit von der Phase beschreibt mathematisch eine gerade Funktion, die im Symmetriepunkt betrachtet wird, es fehlt also die Richtungsinformation der Phasenänderung.

Um auch die Richtung der Phasenverschiebung zu detektieren, wird das bereits aus dem Frequenzkamm herausgefilterte Signal in einem Hochfrequenzmischer mit einem weiteren Referenzsignal gemischt. Auch dieses als Lokaloszillator (LO) bezeichnete Signal wird aus dem Pulszug des MLO mit einer Photodiode und HF-Filtern gewonnen. Es wird die gleiche Harmonische der Repetitionsrate wie für die Messung verwendet.

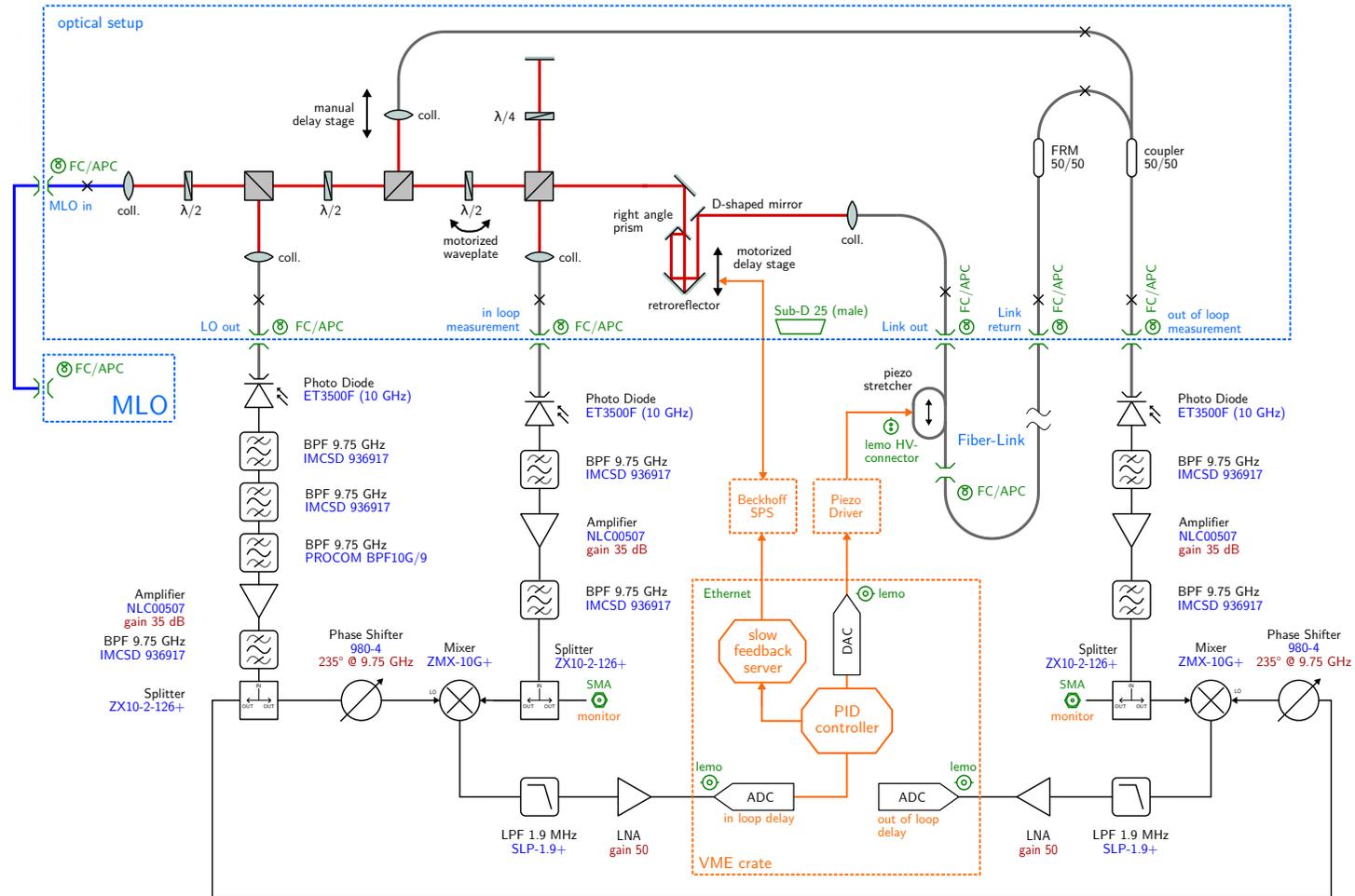


Abbildung 5.1.: Schematischer Aufbau des Faserlinks mit Detektor

Der Hochfrequenzmischer erzeugt nun aus diesen beiden Signalen das eigentliche Messsignal, in das Phase und Amplitude der modulierten Harmonischen eingehen. Abbildung 5.1 zeigt eine vollständige Übersicht über den hier entwickelten Faserlink mit Detektor.

Die konkrete Auswahl der Harmonischen zur Messung der Phasenverschiebung wird zunächst dadurch eingegrenzt, dass die Empfindlichkeit der Messung mit steigender Frequenz steigt. Die Amplitudenmodulation und der Mischprozess sind beide auf trigonometrische Funktionen zurückzuführen, deren maximaler Anstieg im Nulldurchgang liegt. Mit steigender Frequenz steigt dort die zeitliche Empfindlichkeit. Die Verwendung einer Harmonischen zwischen neun und zehn GHz wird angestrebt, da die Empfindlichkeit in diesem Bereich ausreichend hoch ist, und noch eine große Auswahl an HF-Bauteilen und Photodioden am Markt verfügbar ist. Die konkret verwendete Harmonische wird durch die Auswahl der Bandpassfilter bestimmt.

Um für Auslenkungen aus dem Arbeitspunkt empfindlich zu sein, wird bereits im Hochfrequenzteil der Schaltung um 35 dB verstärkt. Eine direkte Detektion der Amplitude an dieser Stelle wurde bereits in [Zem08] und [ZAB+09] mit der Hilfe von verschiedenen Leistungsdetektoren untersucht. Die Ergebnisse dieser Messungen zeigen allerdings keine ausreichende Stabilität (siehe auch Abschnitt 3.3).

Bei dem hier vorgestellten Messverfahren wird zur Detektion nicht auf die zuvor erwähnten Leistungsdetektoren zurückgegriffen. Stattdessen werden hier die Eigenschaften eines Hochfrequenzmischers ausgenutzt. Vereinfacht gesagt wird an einem solchen Mischer immer die Summen- und die Differenzfrequenz der aufgeschalteten Eingangsschwingungen gebildet. Wird der Mischer mit zwei Signalen gleicher Frequenz f betrieben, erzeugt er einerseits die doppelte Grundfrequenz $2f$. Andererseits wird die Differenzfrequenz der Eingangssignale in diesem Fall zu Null. Die doppelte Grundfrequenz wird hinter dem Mischer mit einem Tiefpassfilter blockiert. Das Ausgangssignal ist demzufolge eine Gleichspannung, deren Größe von der Amplitude und der relativen Phase der beiden gemischten Signale abhängt. Für den vorliegenden Detektor wird das hochfrequente Messsignal mit dem zuvor erzeugten LO-Signal gemischt. Die relative Phase dieser beiden Signale kann mit Hilfe eines jedem Mischer vorgeschalteten Phasenschiebers eingestellt werden. Das Ausgangssignal wird zur besseren Aussteuerung der ADCs, mit denen das Signal digitalisiert wird, noch mit einem LNA um Faktor 50 verstärkt. Auf den Mischprozess wird in Abschnitt 5.3 genauer eingegangen.

Das erzeugte Signal folgt der Zeitverschiebung zwischen den Laserpulszügen in Form einer Winkelfunktion, die für kleine Auslenkungen um den Nulldurchgang als linear angenommen werden kann. Da die zu messenden Zeitänderungen klein gegenüber der Periodendauer sind ist diese Annahme für den betrachteten Messbereich zutreffend. Eine Abschätzung dieses systematischen Fehlers wird in Unterabschnitt 5.7.1 vorgenommen.

Die inzwischen schon oft erwähnte Auswahl des Arbeitspunktes wird in Abschnitt 5.5 näher beleuchtet.

5.2. Simulation des Messprinzips

Um eine erste Fehlerabschätzung vornehmen zu können und um einen Arbeitspunkt zu finden, bei dem die Messeinrichtung möglichst unempfindlich gegenüber äußeren Einflüssen ist, wird für das Messprinzip eine Simulation in Matlab erstellt. Abbildung 5.2 zeigt einen Screenshot der grafischen Oberfläche (GUI) der Simulation. Die Simulation wurde mit einer GUI entworfen, um direkt den Einfluss der verschiedenen Parameter auf das Messsignal beobachten zu können. Sie gliedert sich in zwei Teilbereiche.

Im oberen Bereich ist der bei der Modulation von zwei überlagerten Pulszügen entstandene Frequenzkamm und seine Einhüllende zu sehen. Es können Dirac-, Gauß- oder *sech*²-Pulse zur Simulation ausgewählt werden. Außerdem kann die Leistung und die Breite der einzelnen Pulse variiert und die Veränderung des Frequenzspektrums bei allen Einstellungen direkt beobachtet werden. Wichtigster Parameter ist jedoch die Zeitverschiebung zwischen den beiden Pulszügen. Das Frequenzspektrum wird bis zur 45. Harmonischen bei 9,75 GHz dargestellt, die später auch im Versuchsaufbau zur Messung genutzt wird. Bei der Simulation muss beachtet werden, dass die Amplituden der Frequenzspektren alle auf Eins normiert sind.

Im unteren Bereich ist parallel zu sehen, wie das eigentliche Messsignal bei den gewählten Simulationseinstellungen am Ausgang des Mischers aussehen würde. Als weitere Parameter können hier die Amplitude des LO-Signals sowie seine Phase relativ zum Referenzpulszug eingestellt werden. Da die Ausgangsgröße proportional zur LO-Amplitude ist, kann über diese Einstellung auch untersucht werden, wie sich Veränderungen der Verstärkung im HF-Zweig auf das Ausgangssignal auswirken. Als Fehlergröße kann außerdem ein ungewollter DC-Offset am Mischerausgang simuliert werden.

Der Plot im unteren Bereich der GUI zeigt immer in der Mitte des Plots das aktuelle Ausgangssignal. Zu seinen beiden Seiten kann man im Zeitbereich sehen, wie das Ausgangssignal sich ändern würde, wenn das Timing in die entsprechende Richtung verschoben wird. Im Hintergrund sieht man zur Orientierung die einzelnen Anteile, aus denen sich das Messsignal zusammensetzt. Das sind konkret die Amplitude der modulierten Harmonischen, ihre Phase und der Kosinus dieses Phasenwinkels, der dann in die Berechnung des Ausgangssignals eingeht. Auch hier sind wieder alle Amplituden normiert, trotzdem wird zur Orientierung der momentane Messwert auch als Zahlenwert außerhalb des Plots angezeigt.

Die Simulation wird verwendet, um einen geeigneten Arbeitspunkt auszuwählen. Die Auswahl des Arbeitspunktes wird in Abschnitt 5.5 erläutert. Außerdem werden anhand

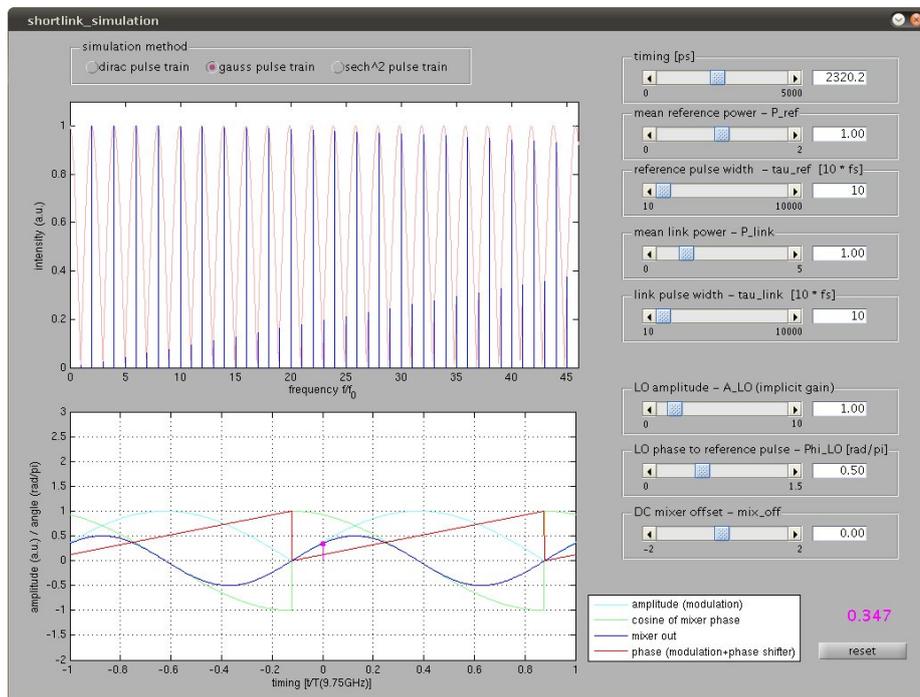


Abbildung 5.2.: Screenshot der Simulation des Detektors

der Simulation die Auswirkungen der Pulsbreite auf die Messempfindlichkeit abgeschätzt (siehe dazu Unterabschnitt 5.7.3).

Der Quellcode der Simulation befindet sich in Anhang B, eine ausführbare Version im digitalen Anhang, weitere Informationen hierzu stehen in Anhang A.

5.3. Komponentenauswahl

Zum Aufbau der Messschaltung werden einige sorgfältig ausgewählte Elektronikkomponenten benötigt. Über ihre wesentlichen Eigenschaften wird im Folgenden ein kurzer Überblick gegeben.

Photodioden

Die zur Detektion der Laserpulszüge verwendeten InGaAs-Photodioden vom Typ ET-3500F des Herstellers EOT besitzen laut Datenblatt eine Bandbreite von über 10 GHz, sie sind also bestens für die hier gemessene Frequenzharmonische von 9,75 GHz geeignet. Vor Beginn der Messungen wird das Sättigungsverhalten aller Photodioden mit einem Rohde&Schwarz Spectrum Analyzer (FSV13) untersucht. Abbildung 5.3 zeigt anhand einer Diode exemplarisch das Ergebnis dieser Messungen.

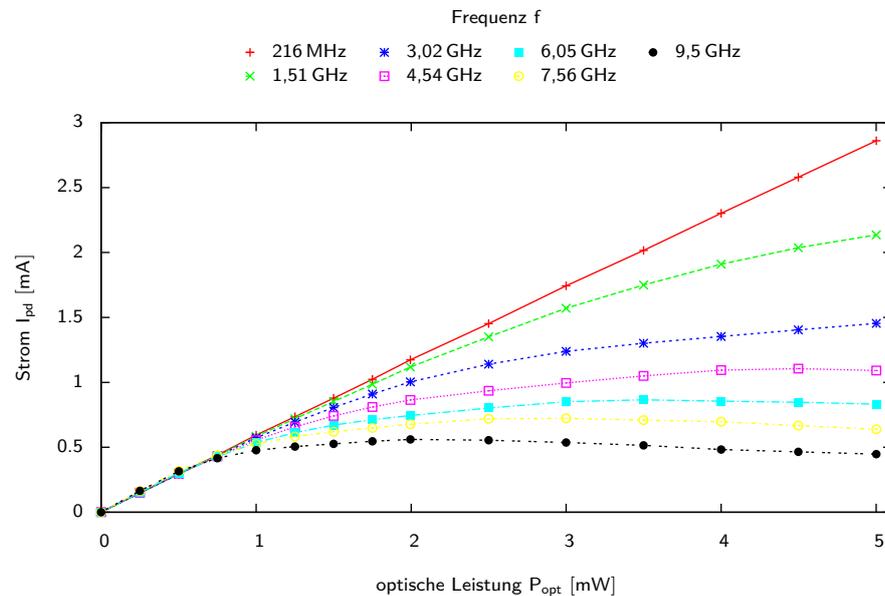


Abbildung 5.3.: Frequenzabhängige Sättigung einer Photodiode (ET-3500F, Seriennummer: 2106NIF)

Für verschiedene optische Leistungen, die zur Messung mit einem variablen optischen Abschwächer eingestellt werden, wird am Spectrum Analyzer der Anteil der verschiedenen Frequenzkomponenten am Gesamtsignal untersucht. Man erkennt deutlich, dass die Grundharmonische im betrachteten Leistungsbereich nicht sättigt, während mit zunehmender Frequenz bei immer kleineren Leistungen Sättigungseffekte zu beobachten sind. Bei der höchsten untersuchten Frequenz von 9,5 GHz sättigt die Diode bereits ab Leistungen von 1,5 mW, was auch für zwei Pulszüge dieser Leistung noch ausreichend weit von der Zerstörschwelle der Photodiode entfernt ist. Diesen Effekt kann man ausnutzen um kleine Leistungsschwankungen, die sonst zu Messfehlern führen würden, zu unterdrücken. Er tritt bei allen drei untersuchten Dioden in identischer Form auf. Die Photodiode zur LO-Erzeugung wird bei allen Messungen prinzipiell mit 2,5 mW gesättigt.

Kann man auch Sensitivitätsverluste in Kauf nehmen, besteht außerdem die Möglichkeit die Bias-Spannung der Photodiode abzusenken. Mit dem sinkenden Ausgangspegel der Photodiode sinkt auch die Leistung, ab der die Photodiode sättigt. Abbildung 5.4 zeigt diesen Effekt bei einer Frequenz von 9,5 GHz. Da der Ausgangspegel der Photodiode für die Empfindlichkeit der Messungen eine große Rolle spielt, wird bei allen Versuchen eine Bias-Spannung von 15 V gewählt. Bei den später erreichten Leistungen auf der Photodiode von etwa 1,3 mW ist die Diode zwar noch nicht völlig in der Sättigung, die Kurve aber schon sehr flach, sodass der Einfluss von Leistungsschwankungen zumindest minimiert wird.

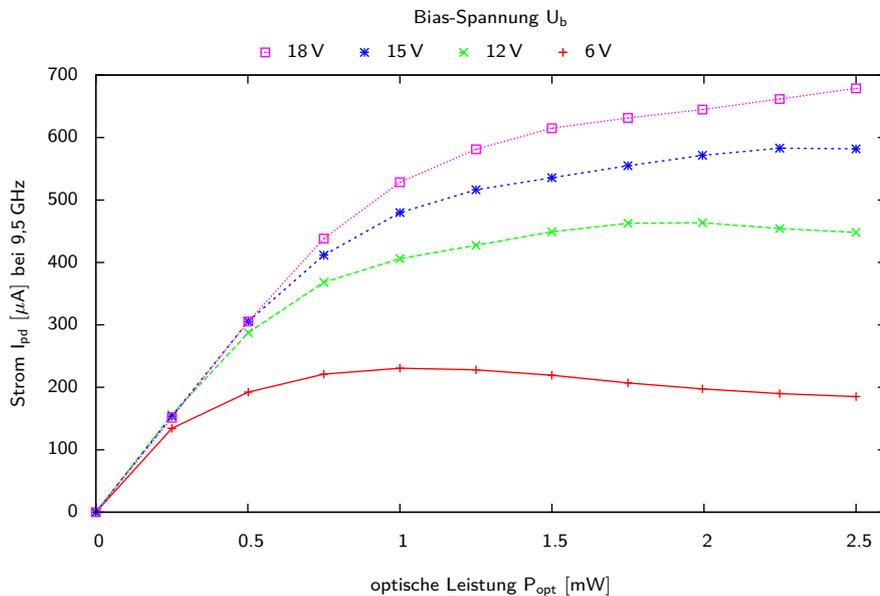


Abbildung 5.4.: Sättigung einer Photodiode in Abhängigkeit von der Bias-Spannung (ET-3500F, Seriennummer: 2106NIF)

HF-Mischer

Ein zentrales Bauelement der Messkette ist der Mischer, der die wichtige Aufgabe übernimmt, das hochfrequente Messsignal in einen Frequenzbereich zu transformieren, in dem es ohne großen Schaltungsaufwand messbar ist. Außerdem wird der gemessenen absoluten Phasenverschiebung am Mischer wieder ein Vorzeichen hinzugefügt, das unbedingt benötigt wird, um die Richtung der Timingänderung am Link detektieren zu können.

Ein idealer Mischer kann mathematisch als Multiplizierer betrachtet werden. Er hat zwei Eingänge, den sogenannten LO-Eingang, über den die internen Dioden geschaltet werden, und den RF-Eingang, an dem das Messsignal eingespeist wird. Dessen Frequenz wird durch den Mischprozess um die LO-Frequenz verschoben und dann am IF-Ausgang ausgegeben. Es gilt nach [Maa93, S. 4]

$$U(t) = A_1 \sin(\omega_1 t) \cdot A_2 \sin(\omega_2 t + \varphi) \quad (5.1)$$

$$U(t) = \frac{A_1 A_2}{2} [\cos((\omega_2 - \omega_1) \cdot t + \varphi) - \cos((\omega_2 + \omega_1) \cdot t + \varphi)] \quad (5.2)$$

Die Summenfrequenz kann vernachlässigt werden, da sie in der vorliegenden Anwendung von einem Tiefpassfilter aus dem Ausgangssignal ausgefiltert wird. Wenn die beiden

5. Aufbau des Detektors

Signale die gleiche Frequenz $\omega_2 = \omega_1 = \omega$ haben vereinfacht sich der Term aus Gleichung 5.2 zu

$$U = \frac{A_1 A_2}{2} \cos(\varphi_0 + \Delta\varphi) \quad (5.3)$$

Aus Gleichung 5.3 folgt, dass die Ausgangsspannung eines idealen Mischers unter den oben genannten Bedingungen keine Funktion der Zeit sondern nur der Phasenverschiebung zwischen den beiden Signalen ist. Diese Phasenverschiebung wurde sozusagen in das Basisband heruntergemischt. Der Mischer wird als Phasendetektor betrieben. Für kleine Winkel um den Arbeitspunkt bei $\varphi_0 = \pi/2$ kann angenommen werden, dass $U \propto \Delta\varphi$ ist.

Zunächst wird ein Mischer des Herstellers Minicircuits vom Typ ZMX-10G verwendet. Es stellt sich jedoch heraus, dass dieser Mischer am Ausgang einen relativ hohen DC-Offset zeigt. Dieser Offset kann am Besten beobachtet werden, wenn man den RF-Eingang mit einem $50\ \Omega$ -Widerstand abschließt. Verschiebt man jetzt am Phasenschieber im LO-Zweig die LO-Phase verändert sich die Gleichspannung am Ausgang des Mischers, die bei einem idealen Mischer Null ist. Der DC-Offset wird offensichtlich durch Leistungsreflexionen verursacht, die je nach Phasenwinkel miteinander konstruktiv oder destruktiv interferieren. Dieser Offset verhindert die exakte Einstellung des Arbeitspunktes, da er nicht konstant ist. Von den vorhandenen vier Mischern konnten die beiden mit dem kleinsten Offset selektiert werden, was allerdings noch immer keine zufriedenstellende Einstellung des Arbeitspunktes erlaubte.

Das Voltage Standing Wave Ratio (VSWR) kann als Maß für die an einem Bauteil reflektierte Leistung interpretiert werden. Es hängt beim vorliegenden Mischer von der Frequenz und der LO-Leistung ab. Der Mischer weist mit den gegebenen Parametern ein VSWR von drei am LO-Eingang und von fünf am RF-Eingang auf, was als relativ hoch betrachtet werden muss. Um die Auswirkungen dieser schlechten Anpassung zu reduzieren, werden an den einzelnen Anschlüssen des Mischers Abschwächer eingebaut. Durch einige Untersuchungen konnte eine für diese Schaltung optimale Konfiguration von einem 5 dB Abschwächer am LO-Eingang und einem 1 dB Abschwächer am IF-Ausgang gefunden werden. Vor dem RF-Eingang ist ein Splitter eingebaut um einen Messpunkt bereitzustellen, sodass ein separater Abschwächer nicht nötig ist. Der Splitter hat bereits eine Dämpfung von 3 dB.

Vor den Optimierungen beträgt der Offset des unempfindlichsten Mischers hinter dem LNA (Verstärkung 50) etwa 200–500 mV, beim schlechtesten Mischer bis zu 2 V, abhängig von der Stellung des Phasenschiebers. Das entspricht je nach Kalibrationskonstante einer Abweichung vom Arbeitspunkt von 100 fs bis zu 1 ps. Nach der Optimierung kann dieser Wert für den besten Mischer auf 5–10 mV reduziert werden, allerdings auf Kosten der Signalamplitude.

Um eine bessere Lösung für diese Probleme zu finden, werden Mischer der Firma Macom (Typ M14A) bestellt, die ein wesentlich geringeres VSWR von unter zwei aufweisen. Sie wurden für die Messungen allerdings nicht rechtzeitig geliefert. Tests

zeigen ferner, dass auch diese Mischer einen DC-Offset am Ausgang aufweisen. Er ist fast konstant und nicht von der Einstellung des Phasenschiebers abhängig, sodass er bei der Einstellung des Arbeitspunktes als konstanter Fehler berücksichtigt werden kann. Dieser Offset wird dadurch verursacht, dass der Mischer nicht perfekt symmetrisch ist. Um einen generellen Einsatz dieser Mischer zu empfehlen, sind weitere Untersuchungen nötig.

HF-Filter

Direkt hinter der Photodiode wird mit HF-Filtern die untersuchte Harmonische aus dem Frequenzkamm herausgefiltert. Die verwendeten Bandpassfilter sind wegen der speziellen Anforderungen Sonderanfertigungen. Zur Selektion einer Harmonischen wird ein Filter mit einer Bandbreite von unter 300 MHz benötigt, das eine möglichst hohe Unterdrückung im Sperrbereich aufweist, und diese auch für weit von der Resonanzfrequenz entfernte Frequenzen aufrecht hält.

Um die extrem langen Lieferzeiten solcher Filter zu umgehen, werden bei DESY bereits vorhandene Filter eingesetzt. Ziel der Messung ist es, eine Harmonische zwischen neun und zehn GHz zu messen. In diesem Frequenzbereich stehen zwei verschiedene Filtertypen zur Auswahl, die im Folgenden beide eingesetzt werden. Die 45. Harmonische bei 9,75 GHz wurde also speziell für die Messungen ausgewählt, weil für sie Filter direkt verfügbar sind.

Das schmalbandigere Filter ist ein elliptisches Filter des Herstellers IMCSD. Der Frequenzgang des Filters ist in Abbildung 5.5 zu sehen. Die benachbarten Harmonischen bei $\pm 216,6$ MHz werden zwar mit 35 dB unterdrückt, der Frequenzgang steigt jedoch anschließend wieder an, um dann auf beiden Seiten der Resonanzfrequenz nur langsam abzufallen.

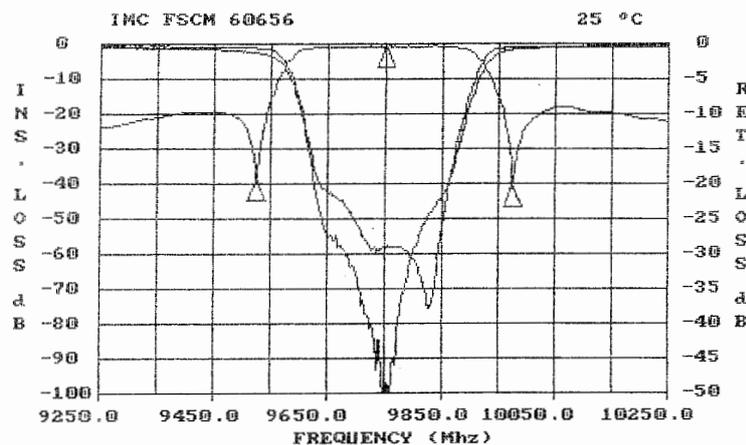


Abbildung 5.5.: Frequenzgang des HF-Bandpassfilters (IMCSD - Typ 936917)

5. Aufbau des Detektors

Das breitbandigere Filter ist so dimensioniert, dass sein Durchlassbereich zwei Harmonische abdeckt, was ungünstig für die vorliegende Anwendung ist. Allerdings weist das Filter im Sperrband eine extrem hohe Dämpfung von 90 dB auf. Es handelt sich hier um ein sogenanntes Interdigitalfilter mit einer Bandbreite von 600 MHz.

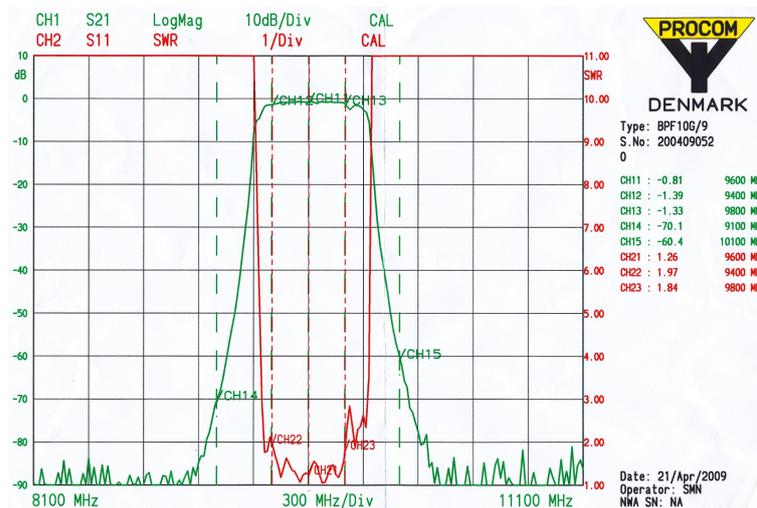


Abbildung 5.6.: Frequenzgang des HF-Bandpassfilters (PROCOM BPF10G/9)

Eine gute Dämpfung der unerwünschten Harmonischen ist für die Messung von größter Wichtigkeit, da alle verbleibenden Frequenzanteile am HF-Mischer ebenfalls gemischt werden und als Fehler das eigentliche Messsignal überlagern. Wie bereits im Abschnitt über HF-Mischer erläutert, wird nur der in das Basisband heruntergemischte Anteil der Messgröße betrachtet. Wenn am Mischer allerdings weitere Harmonische sowohl am RF- als auch am LO-Eingang auftauchen, wird jede einzelne von ihnen ins Basisband heruntergemischt und überlagert additiv das Messsignal. Es ist zu beachten, dass für ein nahezu perfekt amplitudenmoduliertes Messsignal von einer Leistung von etwa -70 dBm ausgegangen werden kann, während die benachbarten Harmonischen eine Leistung von etwa -20 dBm haben. Die Leistung der benachbarten Harmonischen ist also bei der Verwendung nur eines Filters immer noch wesentlich höher als die Leistung des amplitudenmodulierten Signals. Um diesen Fehler zu minimieren, werden im Messaufbau in jedem Messzweig zwei IMCSD-Filter verwendet, im LO-Zweig sogar drei und zusätzlich ein PROCOM Bandpass. Die Idee dahinter ist, den LO möglichst perfekt zu filtern, um die parasitären Mischprodukte so klein wie möglich zu halten.

Verstärker

Zur Verstärkung der Messsignale werden zwei verschiedene Verstärker eingesetzt. In jedem der beiden Messzweige ist dauerhaft ein rauscharmer Verstärker des Herstellers nextec-RF (Typ NLC00507) im Einsatz. Dieser Verstärker für Frequenzen von 9–10 GHz

zeichnet sich durch seine hohe Verstärkung von 35 dB bei einer gleichzeitig sehr kleinen Rauschzahl von 1,3 dB (typisch) aus. Die Ausgangsleistung im 1 dB Kompressionspunkt beträgt 10 dBm.

Für Messungen mit sehr kleinen optischen Leistungen wird noch ein zweiter Verstärker des Herstellers Minicircuits (Typ ZX60-14012L+) verwendet. Dieser Verstärker ist für Frequenzen in einem größeren Bereich von 0,3–14 GHz geeignet. Er hat bei einer Ausgangsleistung im 1 dB Kompressionspunkt von 11 dBm nur eine Verstärkung von 12 dB. Auch die Rauschzahl ist mit 5,5 dB im genutzten Frequenzbereich deutlich schlechter als bei dem zuerst vorgestellten Verstärker. Er wird deshalb, wenn er verwendet wird, prinzipiell hinter dem rauscharmen nextec-RF Verstärker eingebaut, um das Gesamttrauschen zu minimieren.

Phasenschieber

Als Phasenschieber werden mechanische Phasenschieber des Herstellers Aeroflex Weinschel ausgewählt. Das eingesetzte Modell 980-4 ist für Frequenzen bis 12 GHz geeignet. Es erlaubt eine ausreichend große Phasenverschiebung von 235 Grad bei 9,75 GHz. Um die Hochfrequenztauglichkeit sicherzustellen sollten spezielle Phasenschieber für die verwendeten Frequenzen und keine Standardbauteile verwendet werden. Die Phasenschieber haben zudem den Vorteil, sehr kompakt gegenüber Phasenschiebern für niedrige Frequenzen zu sein.

Low Noise Amplifier (LNA)

Der sogenannte Low Noise Amplifier (LNA) ist ein bei DESY gebauter und häufig verwendeter, rauscharmer DC-Spannungsverstärker auf Basis des JFET-Operationsverstärkers AD797 des Herstellers Analog Devices. Der Operationsverstärker ist in erster Linie als Impedanzwandler geschaltet, wobei über einen DIP-Schalter eine moderate Verstärkung von 10, 20, 50 oder 100 ausgewählt werden kann. Er wird üblicherweise als rauscharmer Messverstärker eingesetzt. Sein äquivalentes Eingangsrauschen beträgt etwa $1 \text{ nV}/\sqrt{\text{Hz}}$ und bei der verwendeten Verstärkung von 50 erreicht der Verstärker eine Bandbreite von 1 MHz.

5.4. Aufbau des Detektors

Der Zusammenbau der beschriebenen Komponenten erfolgt wie in Abbildung 5.1 bereits skizziert wurde. Zusätzlich zu den eingezeichneten Komponenten wird für einige Messungen der bereits vorgestellte zweite Verstärker eingesetzt, weil hier mit besonders niedrigen optischen Leistungen gearbeitet wird. Bei den Messungen mit höheren optischen Leistungen, bei denen nur ein Verstärker verwendet wird, werden zusätzliche Abschwächer in den jeweiligen Messzweig eingefügt, um so die Leistung und dadurch auch den Messbereich einstellen zu können.

5. Aufbau des Detektors

Alle Komponenten, die einen großen Temperaturkoeffizienten haben, werden mit Halteklammern am optischen Tisch fixiert, um eine gute Wärmeübertragung zu gewährleisten und um sie an die große thermische Masse des Tisches anzubinden. Das betrifft konkret die Photodioden, die Mischer und die Low Noise Amplifier. Zusätzlich werden die Mischer und Photodioden mit Luftpolsterfolie gegen schnelle Temperaturänderungen durch den Einfluss der Lufttemperatur abgeschirmt. Die Faserkomponenten, die aus dem optischen Aufbau kommen, sind mit Klebeband am optischen Tisch befestigt und zusätzlich mit Schaumstoff abgedeckt, genauso der Faserstretcher.

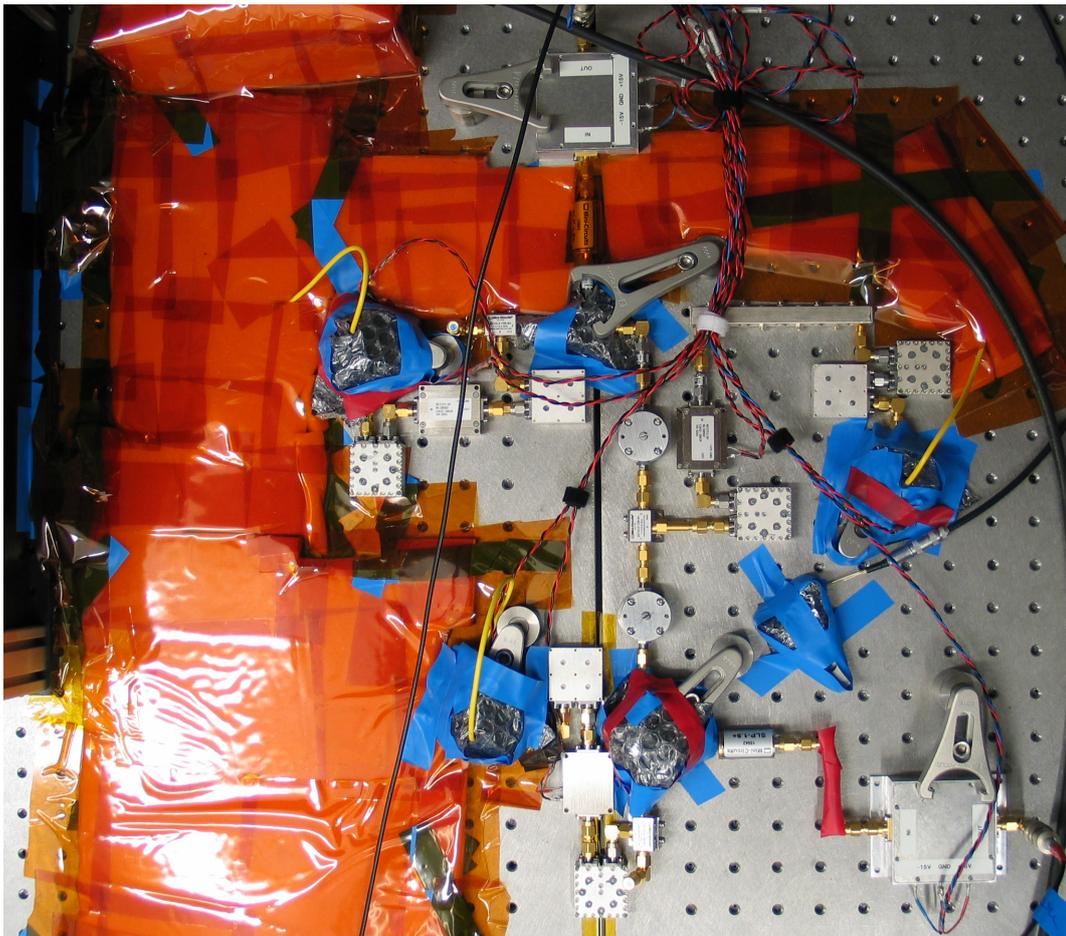


Abbildung 5.7.: Foto des aufgebauten Detektors

In Abbildung 5.7 ist die aufgebaute Messschaltung abgebildet. Oben links, in Schaumstoff eingepackt, kann man den Piezostretcher sehen, daneben den LNA der out-of-loop Messung. Direkt darunter ist mit einer Klammer der HF-Mischer auf dem optischen Tisch fixiert. Links daneben ist anhand der gelben Faser die Position der dazugehörigen Photodiode zu erkennen. In der Mitte des Bildes ist ein Splitter mit den beiden

runden Phasenschiebern zu sehen. Dort wird das LO-Signal auf die beiden HF-Mischer aufgeteilt. Am unteren Rand des Bildes ist analog zum oberen Bereich die in-loop Messung aufgebaut. Darüber im Bereich rechts oben wird das LO-Signal erzeugt. Gut zu erkennen ist der lange HF-Filter des Herstellers PROCOM. Ganz am linken Rand des Bildes ist ein Teil des Gehäuses für den optischen Aufbau zu sehen.

5.5. Arbeitspunkteinstellung

Der Arbeitspunkt der Messeinrichtung beinhaltet zwei wichtige Bedingungen. Zum einen muss die Amplitude der Harmonischen, des aus dem Link zurückkommenden Laserpulszuges exakt der Amplitude der Harmonischen des Referenzpulszuges entsprechen, zum anderen muss der Phasenwinkel am Mischer exakt $\pi/2$ zwischen LO- und Referenzpulszug betragen. Diese Einstellungen führen dazu, dass der Messwert am Mischerauszug zu Null wird.

Dieser Arbeitspunkt wurde deshalb ausgewählt, weil es zwei voneinander unabhängige Bedingungen gibt, die das Messsignal bei konstantem Linktiming auf Null halten. Treten beispielsweise Amplitudenschwankungen im Link auf, bleibt das Ausgangssignal trotzdem Null, da immer noch die Phasenbedingung am Mischer erfüllt ist. Driftet im Gegenzug die LO-Phase und die Amplitude der Pulse ist stabil, so wird das Ausgangssignal über die Amplitudenbedingung auf Null gehalten. Schließlich ist das Timing des Links unverändert geblieben. Diese Fehler treten in realen Applikationen durchaus auf, spielen jedoch im optimalen Arbeitspunkt keine Rolle. Die Fehlerunterdrückung greift allerdings nur, wenn der Link geregelt wird. Entfernt er sich beispielsweise bei Driftmessungen ohne aktivierte Regelung aus dem Arbeitspunkt, dann werden diese Fehler nicht mehr unterdrückt.

Zum Einstellen des Arbeitspunktes stellt man zunächst den Zeitversatz so ein, dass die Amplitude der modulierten Frequenzharmonischen minimal wird. Der Zeitversatz zwischen dem Referenzpulszug und dem vom Link zurückkommenden Pulszug beträgt dann genau $T_{9,75\text{ GHz}}/2$. Es ist von Vorteil, wenn man gleichzeitig darauf achtet, dass der Arbeitspunkt, den man eingestellt hat, in der Nähe von $T_{216,6\text{ MHz}}/2$ liegt, da die Pulse auf der Photodiode dann den maximalen Abstand haben. Ist dieser Punkt gefunden, kann man durch Einstellung des Teilungsverhältnisses am zentralen Strahlteiler die Leistung der beiden Pulszüge so anpassen, dass die Amplituden der Harmonischen gleich sind. Die Amplitude der Modulation verringert sich dann weiter, bis zumindest in der Theorie die Frequenzkomponente bei 9,75 GHz und alle anderen ungeraden Harmonischen verschwinden. Der Arbeitspunkt kann genauer eingestellt werden, wenn dieses Vorgehen iterativ wiederholt wird.

Jetzt muss noch die korrekte Phase am Mischer eingestellt werden. Hierzu blockiert man den Laserstrahl, der in den Link geht, sodass nur noch der Referenzpulszug auf die Photodiode fällt. Mit dem manuellen Phasenschieber kann nun die LO-Phase

so lange verschoben werden, bis am Mischer keine Ausgangsspannung mehr anliegt. Die Phasenverschiebung zwischen Referenzpulszug und LO-Signal beträgt jetzt exakt 90 Grad. Nun kann die Blockade des Laserpulszuges aus dem Link aufgehoben werden und der Aufbau befindet sich im optimalen Arbeitspunkt.

Für die out-of-loop Messung muss genauso verfahren werden, allerdings empfiehlt es sich, sie erst nach der in-loop Messung einzustellen, da das out-of-loop Signal am Linkende durch die Einstellung des in-loop Arbeitspunktes verändert wird. Während die in-loop Messung durch die motorisierten Aktuatoren sehr leicht justiert werden kann, ist es bei der out-of-loop Messung nicht immer möglich, den Arbeitspunkt per Hand perfekt einzustellen.

5.6. Kalibration

Zur Kalibrierung des Detektors wird der Umrechnungsfaktor K_φ zwischen der Spannung am Ausgang des Detektors und dem aktuellen Timing bestimmt. Da der genaue Zusammenhang zwischen der Schrittzahl des Schrittmotors und dem zurückgelegten Weg bekannt ist (vgl. Unterabschnitt 4.2.1) und der Zeitversatz, den eine Bewegung des Schlittens auslöst, aus dem Weg berechnet werden kann, wird der Schlitten zur Kalibrierung eingesetzt. Vor und nach einer Messung wird jeweils eine Kalibrierung durchgeführt und dann über beide Werte gemittelt. Es konnte kein Fall beobachtet werden, bei dem die bestimmten Kalibrationskonstanten signifikant voneinander abweichen. Zur Bestimmung der Kalibration werden 21 Messwerte äquidistant in einem Bereich von 2000 Schritten aufgenommen, was einem Bereich von 305 fs am out-of-loop Detektor und dem doppelten Bereich am in-loop Detektor entspricht. Abbildung 5.8 zeigt zwei typische Kalibrationen für den in-loop und out-of-loop Detektor. Die Kalibration wird absichtlich so gewählt, dass der out-of-loop Detektor empfindlicher als der in-loop Detektor ist, um etwa den gleichen Messbereich mit dem ADC abdecken zu können.

K_φ entspricht dem Anstieg der jeweiligen Geraden, ϵ bezeichnet den Korrelationskoeffizienten von Messwerten und Regression. Die nach der Methode der kleinsten Quadrate berechneten Werte lauten

	$K_{\varphi,1}[\text{mV/fs}]$	ϵ_1	$K_{\varphi,2}[\text{mV/fs}]$	ϵ_2
in-loop	2.532	0.999769	2.559	0.999733
out-of-loop	4.653	0.999759	4.727	0.999733

Tabelle 5.1.: Kalibrationskonstante K_φ des Detektors

Die Veränderung der beiden Kalibrationen beträgt nur etwa 1 % bei der in-loop Messung

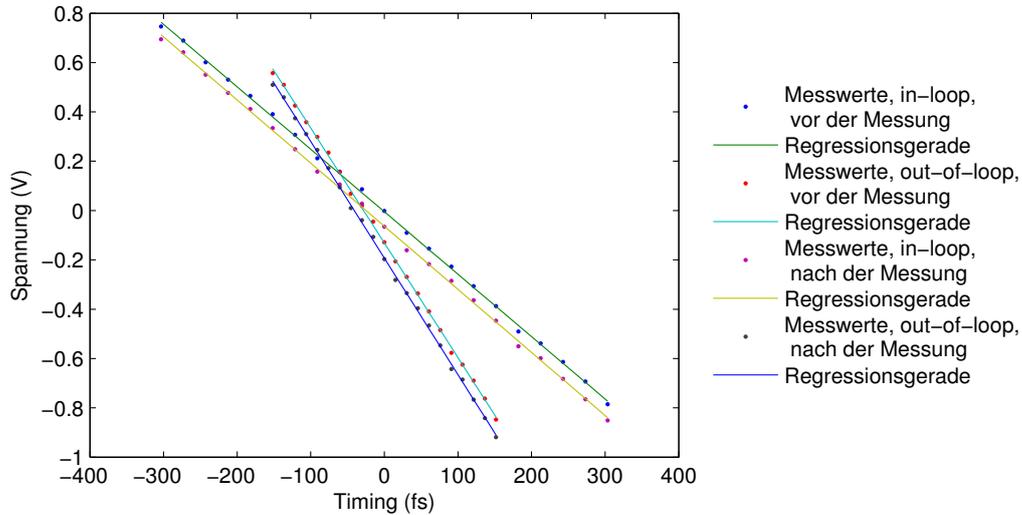


Abbildung 5.8.: Kalibration des Detektors

und 1,6 % bei der out-of-loop Messung über 10 h. Sie kann beispielsweise durch langsame Leistungsdriften des Lasers verursacht werden, die einen Einfluss auf die Empfindlichkeit haben.

Es stellt sich heraus, dass zusätzlich bei jeder Kalibration ein systematischer Fehler gemacht wird. Bei Auslenkungen aus dem Arbeitspunkt ist der Detektor nicht mehr unempfindlich gegen Amplitudenänderungen im Link. Die Amplitude der Linkpulse ändert sich allerdings minimal durch die Strahldivergenz, auch wenn der Schlitten nur in einem kleinen Bereich bewegt wird. Dieser Fehler ist linear und überlagert deshalb die eigentliche Kalibration ohne ihre Linearität zu beeinflussen. Es gibt kein Stellglied um diesen Fehler quantitativ zu untersuchen, da man am Detektor immer Timing- und Amplitudenänderung zusammen misst. Deshalb wird zur Unterdrückung des Fehlers zunächst eine weitere Verbesserung der Delayline (siehe Unterabschnitt 4.2.5) beabsichtigt.

5.7. Begrenzungen des Detektionsprinzips

Die folgenden Abschnitte geben einen kurzen Überblick über die Grenzen des Detektionschemas und es werden einige Überlegungen zur Fehlertoleranz angestellt, bevor im nächsten Kapitel die Messergebnisse der passiven Driftmessungen präsentiert werden.

5.7.1. Linearität des Messbereichs

Es wurde bereits erläutert, dass das durch die Messschaltung erzeugte Signal eine Sinusform hat. Diese Form begrenzt den linearen Messbereich um den Arbeitspunkt. Bei $f_m = 9,75$ GHz und einem ADC der einen Eingangsspannungsbereich von $\hat{U} = \pm 5$ V messen kann beträgt die absolute Abweichung ξ der linearen Interpolation vom tatsächlichen, sinusförmigen Messsignal am Skalenendwert abhängig von K_φ

$$\xi = \frac{\hat{U}}{K_\varphi} - \frac{\arcsin\left(\frac{\hat{U}}{K_\varphi} \cdot 2\pi f_m\right)}{2\pi f_m} \quad (5.4)$$

Der absolute Linearitätsfehler für die gegebenen Parameter und die im vorherigen Abschnitt bestimmte, repräsentative Kalibration beträgt -2,4 fs bei einem Messbereich von 2 ps für die in-loop Messung und -0,8 fs bei einem Messbereich von 2,1 ps für die out-of-loop Messung. Der in-loop Wert wurde halbiert, da am in-loop Detektor auch das zweifache Timing gemessen wird. Für kleine Auslenkungen um den Arbeitspunkt, wie sie im geregelten Fall auftreten, verschwindet der Fehler fast vollständig und selbst die hier berechneten Extremwerte liegen weit unterhalb der geforderten Toleranz von 50 fs. Der Messbereich deckt einen Bereich ab, der groß genug für Driftmessung über mehrere Stunden in einem Labor mit guter Temperaturstabilität ist. Die Auflösung eines 14-Bit ADCs mit dem genannten Eingangsspannungsbereich beträgt $610 \mu\text{V/bit}$, also etwa $0,3 \text{ fs/bit}$ für den schlechter aufgelösten in-loop Detektor. Sie liegt im sub-Femtosekundenbereich und ist somit ausreichend. Die Auflösung verbessert sich weiter, wenn später zur Regelung ADCs mit einem Eingangsspannungsbereich von $\hat{U} = \pm 1$ V bei gleicher Bitbreite verwendet werden.

5.7.2. Rauschen des Detektors

Bei abgenommener Glasfaser kann mit einem Signal Source Analyzer (Agilent, Modell E5052B) das Rauschen des Detektors charakterisiert werden. Es werden sowohl der in-loop als auch der out-of-loop Detektor untersucht. Zur Messung sind beide Verstärker im Detektor eingebaut, was dem Fall mit den meisten Rauschquellen entspricht. Das Rauschen des Detektors wird im Wesentlichen durch die Photodioden und die Verstärker beigesteuert, wobei das Rauschen der Photodiode außerdem durch die Verstärker angehoben wird. In Abbildung 5.9 ist deutlich der Einfluss des Tiefpassfilters hinter dem HF-Mischer zu erkennen, der eine Grenzfrequenz von 1,9 MHz hat. Sein Zweck ist es, hinter dem Mischer die Grundharmonische und die am Mischer erzeugte Summenfrequenz zu unterdrücken. Der obere Plot zeigt die gemessene spektrale Rauschleistungsdichte des in-loop und des out-of-loop Detektors. Der out-of-loop Detektor zeigt gegenüber dem in-loop Detektor im Frequenzbereich von 100 Hz bis 1 kHz ein erhöhtes Rauschen,

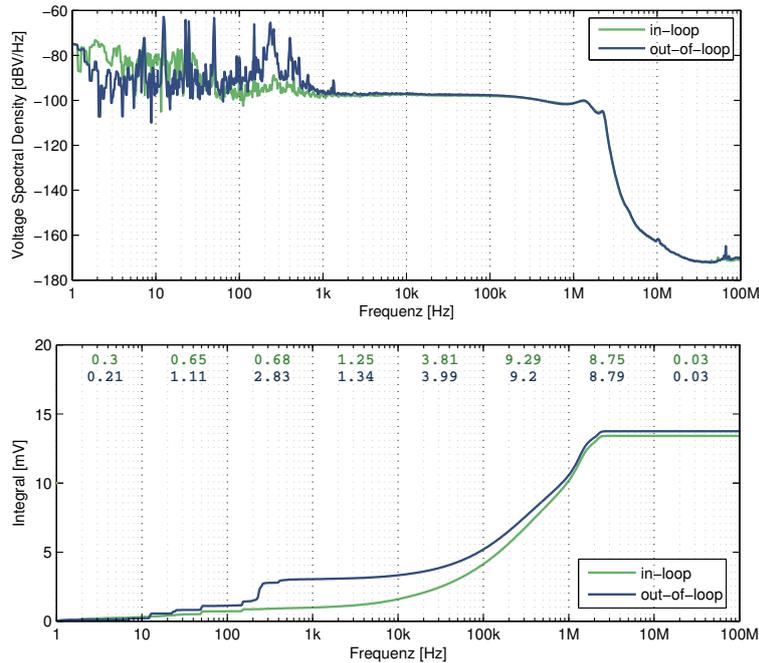


Abbildung 5.9.: Detektorrauschen

was auf eine im Rahmen der Serienstreuung schlechtere Rauschzahl des Verstärkers zurückzuführen ist. Im unteren Teil der Abbildung wird die Wurzel aus dem von kleineren zu größeren Frequenzen hin gebildeten Integral über die Rauschleistungsdichte gezeigt. Man erhält so das RMS-Spannungsrauschen in diesem Frequenzbereich. Am oberen Rand sind die Werte des Integrals jeweils für eine Frequenzdekade vermerkt. So kann abgeschätzt werden, bis zu welcher Frequenz eine Absenkung der Bandbreite sinnvoll ist, wenn das Rauschen des Detektors effektiv reduziert werden soll.

Das RMS-Spannungsrauschen beträgt beispielsweise im Frequenzbereich von 1 Hz bis 10 kHz 1,29 mV für den in-loop Detektor und 3,33 mV für den out-of-loop Detektor. Man erhält es durch quadratische Addition der Anteile in den einzelnen Dekaden. Bei typischen Kalibrationskonstanten im Bereich von wenigen mV/f_s entspricht dieses Rauschen größenordnungsmäßig 1 fs, abhängig von der jeweiligen Kalibration. Liegt die geforderte Messgenauigkeit höher, muss der Signal-zu-Rausch Abstand verbessert werden. Dies kann einerseits durch Reduzierung des Rauschens durch weitere Einschränkung der Bandbreite geschehen oder andererseits indem die Leistung auf der Photodiode erhöht wird.

5.7.3. Pulsverbreiterung durch Gruppengeschwindigkeitsdispersion

Um eine Abschätzung des Einflusses der Pulsbreite auf das Messergebnis vorzunehmen, kann die Breite eines Laserpulses, der eine dispersive Glasfaserstrecke der Länge z durchlaufen hat, abgeschätzt werden. Nach [Agr01, S. 67 ff.] gelten bei Gauß-Pulsen die folgenden Näherungen für die Pulsbreite $\sigma_g(z)$.

$$\sigma_g = \frac{\tau_p}{2 \cdot \sqrt{\ln(2)}} \quad (5.5)$$

$$\sigma_g(z) = \sigma_{g,0} \cdot \sqrt{1 + \left(\frac{z}{L_D}\right)^2} \quad (5.6)$$

$$L_D = \frac{\sigma_{g,0}^2}{|\beta_2|} \quad (5.7)$$

Für standard Telekommunikationsfasern und eine Wellenlänge von 1550 nm beträgt der Faserparameter $\beta_2 = -20 \text{ ps}^2/\text{km}$.

Bei einer maximalen Fasergesamtlänge von 35 m ergibt sich für Hin- und Rückweg eine durchlaufene Glasfaserstrecke von 70 m. Sie setzt sich aus der Länge der Linkfaser, der Länge der Faser auf dem Piezostretcher und den Anschlussfasern an der Optik zusammen. Die Pulsbreite des verwendeten Lasers beträgt $\tau_p = 74 \text{ fs}$. Nach dieser Glasfaserstrecke hat sich die Breite der Laserpulse theoretisch auf etwa $\tau_p = 52 \text{ ps}$ (FWHM) vergrößert.

Auch ohne weitere Informationen über den Detektor zu haben, kann der Einfluss dieser Pulsverbreiterung auf die Amplitude der 45. Harmonischen bei 9,75 GHz in der Simulation (siehe Abschnitt 5.2) betrachtet werden. Die Amplitude der Harmonischen fällt auf etwa ein Drittel ihres ursprünglichen Wertes ab und die Modulationstiefe des Frequenzkamms reduziert sich etwa um die Hälfte.

Das Messsignal wird im Wesentlichen durch die Reduzierung der Maximalamplitude beeinflusst, die direkten Einfluss auf die Empfindlichkeit der Messung hat. In der Simulation ist deutlich zu sehen, dass das Messsignal einer Winkelfunktion entspricht. Die Empfindlichkeit ist durch den Anstieg im Arbeitspunkt, also im Nulldurchgang gegeben, der für kleine Auslenkungen um diesen Punkt proportional zur Maximalamplitude ist. Sinkt die Maximalamplitude, wird auch der Anstieg reduziert und die Empfindlichkeit sinkt, was eine Verschlechterung des Signal/Rausch Abstandes bedeutet, die auch durch weitere Verstärkung nicht kompensiert werden kann. Eine Erhöhung der Leistung auf der Photodiode ist nur bis zur Zerstörschwelle möglich und die Signalanhebung durch die Sättigung begrenzt.

Praktisch wurde beobachtet, dass der Einfluss der Dispersion der hier untersuchten 35 m Faser an einem ADC mit einem Eingangsspannungsbereich von $\pm 1 \text{ V}$, einer Verstärkung von etwa 35 dB und der optischen Gesamtleistung auf der Photodiode von

2,6 mW gerade noch tolerierbar ist, um einen Messbereich von etwa 1 ps zu erreichen. Soll bei den gleichen Eckdaten mit einer wesentlich längeren Faser gearbeitet werden, was im Produktivbetrieb sehr wahrscheinlich ist, wird empfohlen die Dispersion zu kompensieren. Dazu kann beispielsweise die in Abschnitt 4.1 bereits erwähnte dispersionskompensierende Faser (DCF) eingesetzt werden, die zur Kompensation lediglich in der richtigen Länge in den Link eingefügt werden muss.

Wird die Verwendung von dispersionskompensierender Faser in Erwägung gezogen und steht genug Leistung zur Verfügung, um trotz optischer Verluste im Link die Photodiode auszusteuern, wird die maximale Linklänge nur noch durch den Stellbereich des Aktuators begrenzt.

5.7.4. Empfindlichkeit gegenüber anderen Fehlerquellen

Es gibt einige typische Fehlerquellen, die bei anderen Detektorprinzipien zur Messung der Timingänderung von Laserpulszügen mit Photodioden oft die Performance begrenzen. Ihr Einfluss auf den hier entwickelten Detektor wird im Folgenden kurz qualitativ beleuchtet.

Temperaturdriften der Photodiode

Photodioden zeigen üblicherweise eine Phasenabhängigkeit von der Temperatur. Der vorgestellte Detektor misst an der Photodiode die Phase zweier Laserpulszüge relativ zueinander in Form einer Modulation. Da die Phasendriften der Photodiode auf beide Pulszüge den gleichen Einfluss haben, bleibt die Amplitude der gemessenen Harmonischen von ihnen unbeeinflusst.

Leistungsschwankungen im Link

Optische Leistungsschwankungen verursachen an Photodioden sogenannte AM/PM-Effekte. Selbst kleine Leistungsschwankungen (Amplitudenmodulation, AM) werden an Photodioden intrinsisch in Phasenschwankungen (Phasenmodulation, PM) umgesetzt. Treten solche Leistungsschwankungen im Link auf, beeinflussen sie die Phase dieses Pulszuges obwohl sich das Timing möglicherweise gar nicht geändert hat. Dieser Fehler wird durch das Messprinzip nicht kompensiert. Die Amplitudenänderung selbst beeinflusst zusätzlich die Modulation, eine reine Amplitudenänderung wird allerdings durch die Phasenbedingung am Mischer unterdrückt. Die Linkfaser ist im Betrieb fest verlegt und keinen dynamischen Belastungen ausgesetzt, daher werden keine Leistungsschwankungen ausschließlich aus dem Link erwartet. Viel wahrscheinlicher sind Leistungsschwankungen, die direkt aus dem Laser kommen.

Leistungsschwankungen des Lasers

Leistungsschwankungen des Lasers verursachen ebenfalls AM/PM Effekte, allerdings sind in diesem Fall beide Pulszüge auf der Photodiode betroffen. Es gilt für die dadurch induzierten Phasendriften sinngemäß das gleiche Verhalten wie für Temperaturdriften der Photodiode. Die durch Leistungsschwankungen verursachten Amplitudenänderungen, spielen für die Modulation keine Rolle, da die Amplitude beider Pulszüge beeinflusst wird. Zusätzlich wird die Photodiode in einem Bereich betrieben, in dem sie bei der gemessenen Harmonischen bereits in Sättigung und deshalb unempfindlich gegenüber Leistungsänderungen ist.

Driften am Mischer

Im LO-Zweig werden Phasendriften nicht wie im Messzweig ausgeglichen, da nur ein Pulszug auf die Photodiode fällt. Trotzdem bleibt das Ausgangssignal des Detektors im geregelten Fall Null, wenn weiterhin die Bedingung für die Amplitudenmodulation erfüllt ist. Optische Leistungsschwankungen werden durch Sättigung der Photodiode unterdrückt. Außerdem spielt die Empfindlichkeit des Detektors, die auch von der LO-Leistung abhängt, im geregelten Fall nur eine untergeordnete Rolle, da der Arbeitspunkt einen Fixpunkt des Detektionsprinzips, unabhängig von der konkreten Empfindlichkeit, darstellt.

6. Passive Driftmessungen

Um die Zuverlässigkeit des Detektors zu prüfen und verschiedene Konfigurationen zu testen, wurden zahlreiche Messungen durchgeführt. Im Folgenden werden exemplarisch drei Messungen gezeigt, die einen guten Überblick über den abgedeckten Parameterraum und die erreichte Performance geben.

Die gezeigten Driftmessungen werden alle am unregulierten Faserlink aufgenommen. Die Messwerte aus dem Detektor werden mit einem 14-Bit ADC aufgezeichnet. Da für diese Messungen nur Langzeiteffekte von Interesse sind, wird jede Sekunde ein Messwert gespeichert. Um höherfrequentes Messrauschen zu minimieren, wird als Messwert der Mittelwert aus 2048 vom ADC mit einer Abtastrate von 100 kHz aufgenommenen Werten berechnet.

Der gesamte Versuchsaufbau ist im Labor Temperaturschwankungen von im Mittel $\Delta T = 0,2 \text{ K}$ ausgesetzt. Eine detaillierte Untersuchung der Temperaturabhängigkeit des Detektors getrennt von der Linkfaser wird erst mit einem kompakteren Design, das derzeit entwickelt wird, möglich sein. Unter den gegebenen Umständen kann keine quantitative Aussage über die Empfindlichkeit des Detektors getroffen werden, sondern nur über seine Stabilität. Die im Folgenden gezeigten Messwerte wurden unter diesen Umgebungsbedingungen aufgenommen, die auch im Beschleunigerbetrieb für den Detektor gewährleistet werden können.

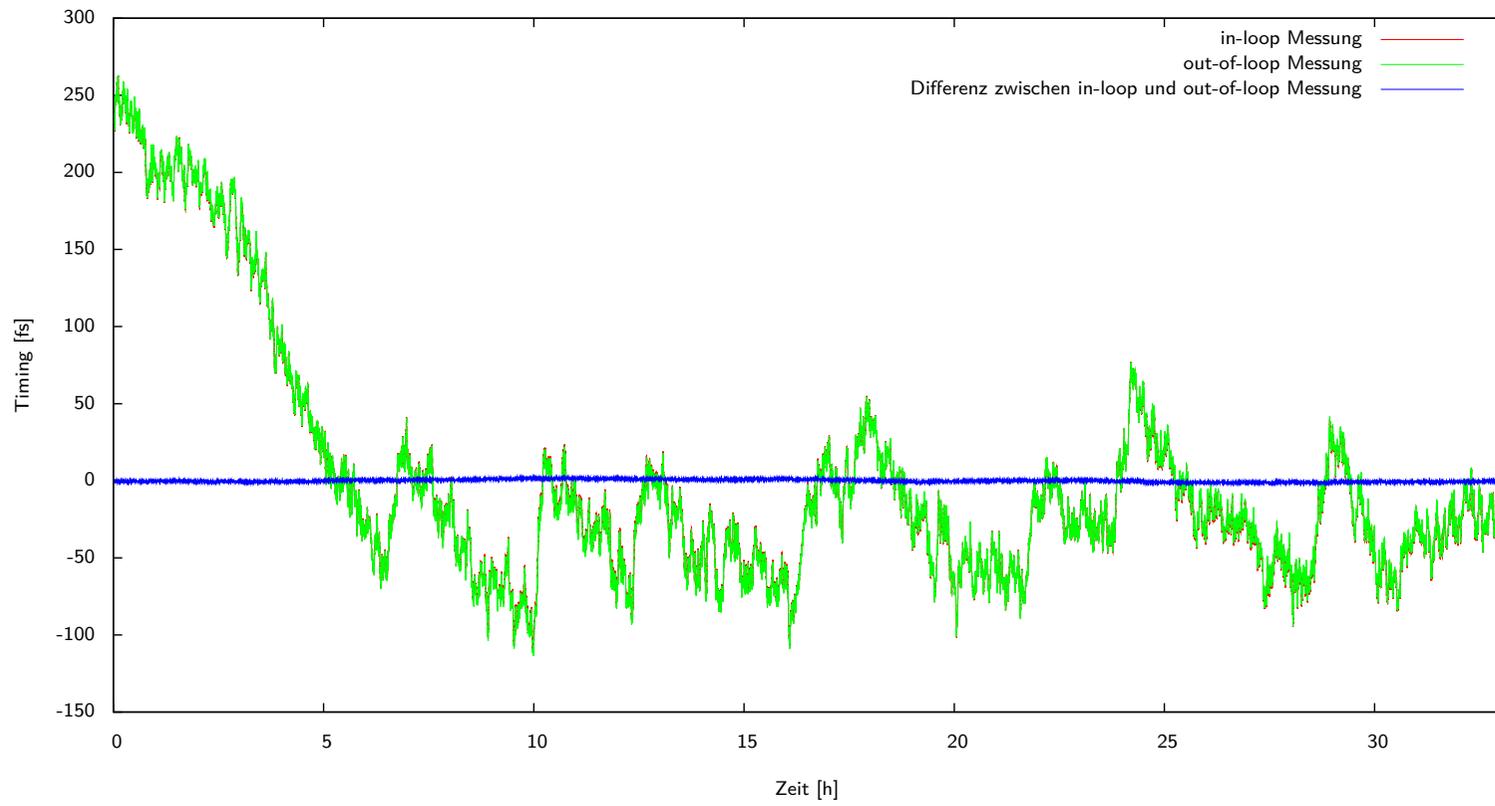


Abbildung 6.1.: 33 Stunden Driftmessung

6.1. Die Langzeitmessung

Die Messung, die in Abbildung 6.1 gezeigt wird, stellt die bisher längste aufgenommene Messung dar. Die Messwerte von in-loop und out-of-loop Detektor werden über 33 h aufgezeichnet. Sie sind im Plot jeweils grün beziehungsweise rot dargestellt. Die angegebenen Zeiten entsprechen den Timingänderungen am Linkende, das bedeutet, dass die Messwerte der in-loop Messung zur Darstellung halbiert werden mussten. In blau wird die Differenz zwischen den beiden Detektoren dargestellt. Sie beschreibt den Messfehler und für diesen langen Messzeitraum ist sie ein Maß für die Langzeitstabilität des Detektors.

Als Linkfaser wird ein 20 m langes SMF-28 Patchcord verwendet. Vor und nach der Messung wird eine Kalibration des Aufbaus durchgeführt. Die gemittelten Kalibrationskonstanten betragen $K_{\varphi,in} = 2,17 \text{ mV/fs}$ und $K_{\varphi,out} = 4 \text{ mV/fs}$, womit sich ein in-loop Messbereich von 2,29 ps und ein out-of-loop Messbereich von 2,5 ps, begrenzt durch den ADC-Eingangsbereich, ergeben.

Die Linkfaser liegt offen im Labor, driftet also abhängig von der Lufttemperatur, während für den Detektor die in Abschnitt 5.4 beschriebenen Maßnahmen zur Entkopplung von der Umgebung ergriffen wurden. Der zweite HF-Verstärker des Herstellers Minicircuits ist zu diesem Zeitpunkt in der Messschaltung eingebaut. Die optische Leistung auf den Photodioden beträgt etwa 0,28 mW, um den erwähnten Messbereich zu gewährleisten.

Der starke Abfall am Anfang der Messung zeigt, wie sich das Versuchslabor nach dem Verlassen abgekühlt hat, bis ein nahezu stabiles thermisches Gleichgewicht mit Schwankungen von etwa 0,1 K erreicht wird. Die Erwärmung erfolgte vor der Messung durch Arbeiten am Aufbau und die Einstellung des Arbeitspunktes.

Hätten die beiden Detektoren verschiedene Temperaturkoeffizienten, müssten die Messwerte über den Messzeitraum auseinanderdriften. Selbst wenn die beiden Detektoren, wenn sie den gleichen Temperaturkoeffizienten hätten, betragsmäßig genau um den gleichen Betrag driften würden, würde diese absolute Phasendrift trotzdem zu einer Abweichung zwischen den Messergebnissen führen, da die Messwerte des in-loop Detektors zur Darstellung halbiert werden.

Die gemessene Abweichung (peak-to-peak) zwischen den Detektoren beträgt 4,8 fs, die Standardabweichung beträgt 0,86 fs. Dieses hervorragende Messergebnis zeigt die Stabilität des Detektors über 33 h und unterstreicht die gute Unterdrückung der möglichen Fehlerquellen.

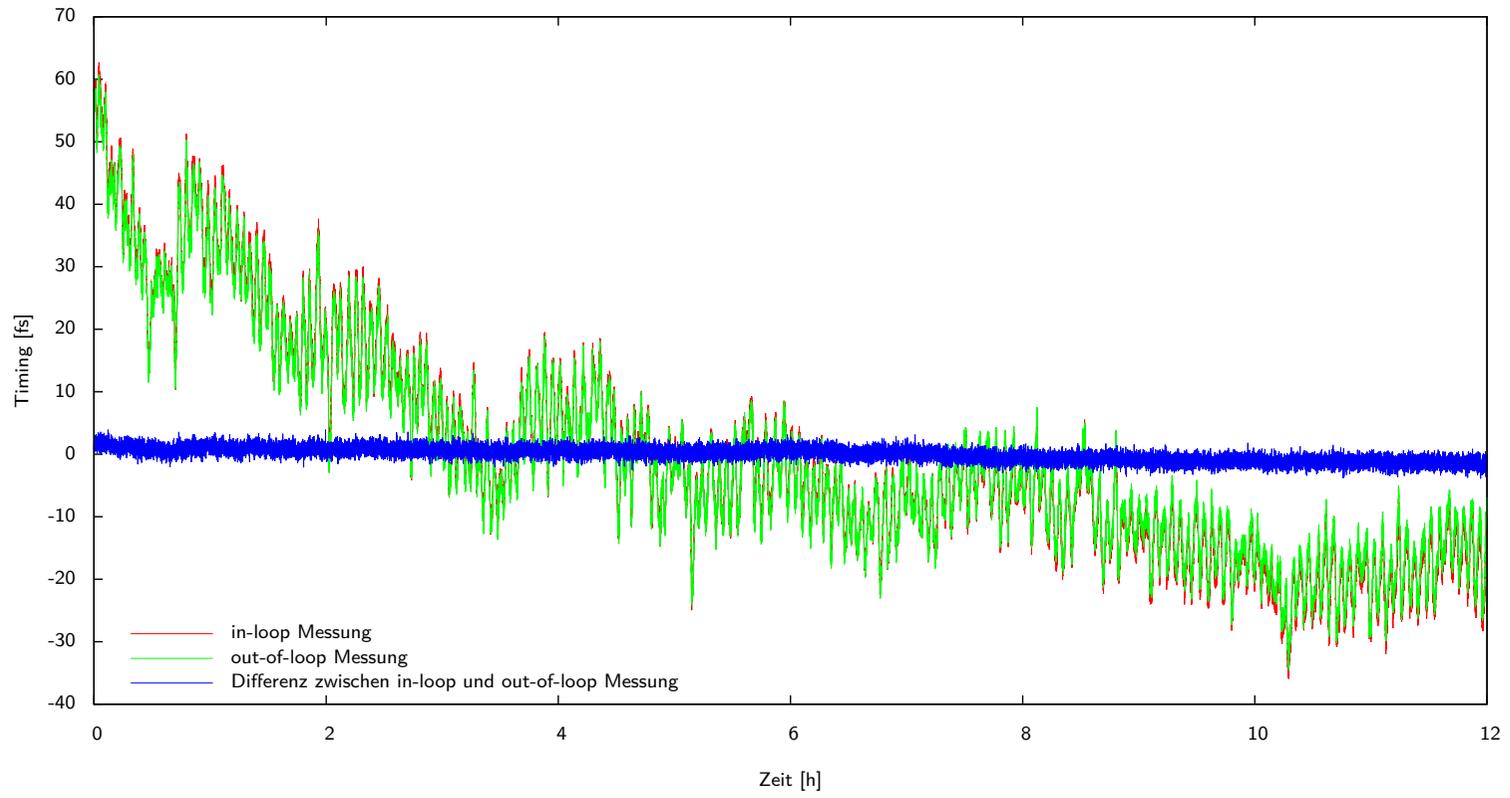


Abbildung 6.2.: Driftmessung mit niedriger Leistung

6.2. Die Low-Power Messung

Abbildung 6.1 zeigt eine Messung zur Untersuchung des Messaufbaus bei extrem niedrigen optischen Leistungen. In diesem konkreten Fall wurde eine optische Leistung auf den Photodioden von etwa $50 \mu\text{W}$ eingestellt. Aus dieser Leistung resultiert ein extrem großer Messbereich von über 13 ps. Die Kennlinie des Detektors verlässt, wenn man ihn bei diesem Messbereich aussteuert, den linearen Bereich des Detektors. Außerdem ist die Messauflösung sehr schlecht.

Als Linkfaser wird ein 1 m langes SMF-28 Patchcord verwendet. So sollen Effekte durch die Linkfaser minimiert werden und eventuell zu erwartende Probleme mit dem Detektor in den Vordergrund treten. Vor und nach der Messung wird wieder eine Kalibration des Detektors durchgeführt. Die gemittelten Kalibrationskonstanten sind mit $K_{\varphi,in} = 0,36 \text{ mV/fs}$ und $K_{\varphi,out} = 0,77 \text{ mV/fs}$ vergleichsweise klein.

Trotzdem zeigen die Messergebnisse, dass das Detektorprinzip auch bei diesen kleinen Leistungen tadellos funktioniert. Die blaue Kurve im Plot zeigt wieder die Differenz zwischen den beiden Detektoren. Das Messergebnis ist mit einer gemessenen peak-to-peak Abweichung zwischen den Detektoren von 7,82 fs und Standardabweichung von 1,11 fs über 12 h immer noch hervorragend.

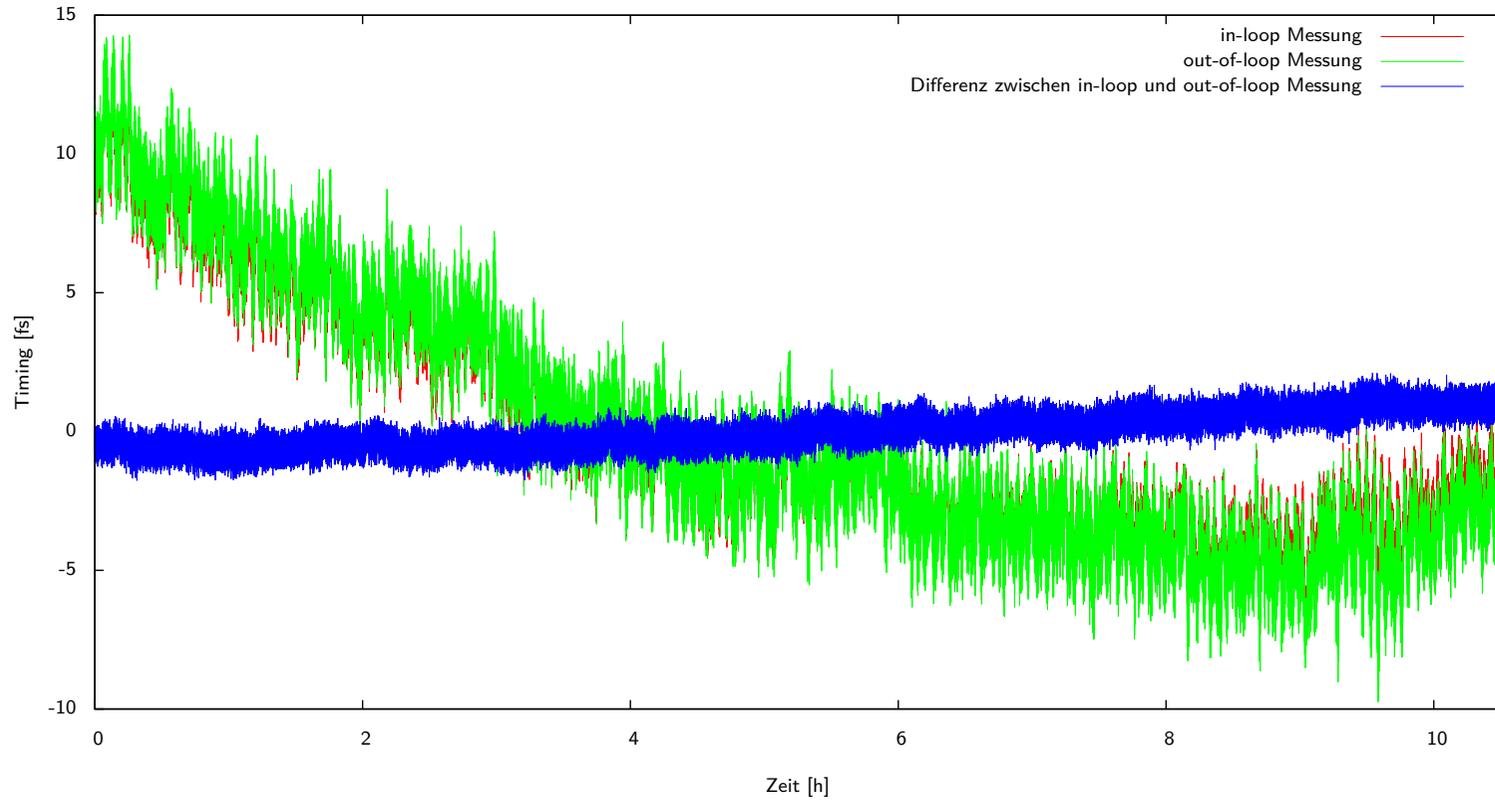


Abbildung 6.3.: Driftmessung mit hoher Leistung und nur einem Verstärker

6.3. Die High-Power Messung

Das Verhalten des Detektors bei hohen Leistungen wurde vor allem untersucht, um die Photodioden möglichst in Sättigung zu betreiben. Die Abhängigkeit von Leistungsschwankungen kann so minimiert werden.

Die maximale Leistung auf den Photodioden, die durch den Laser zur Verfügung gestellt wird, beträgt 2,4 mW. Hier ist allerdings zu beachten, dass auf jeden Pulszug nur die Hälfte dieser Leistung entfällt. Die Photodiode ist also noch nicht völlig gesättigt, Amplitudenschwankungen werden aber schon unterdrückt. Die Kalibrationskonstante bei solchen Leistungen ist sehr groß und der Messbereich somit zu klein für Driftmessungen. Aus diesem Grund wird der kleinere HF-Verstärker bei dieser und den folgenden Messung aus der Schaltung entfernt. Da die optische Leistung nicht mehr als Stellgröße für die Empfindlichkeit zur Verfügung steht, wird ab jetzt im HF-Bereich mit Abschwächern gearbeitet. So können Kalibrationskonstanten für die beiden Detektoren von $K_{\varphi,in} = 2,55 \text{ mV/fs}$ (in-loop) und $K_{\varphi,out} = 4,70 \text{ mV/fs}$ (out-of-loop) eingestellt werden. Der Messbereich ist somit ähnlich groß wie bei der ersten Messung.

Das 20 m lange Patchcord wird wieder für diesen Test verwendet und der Piezostretcher mit etwa 10 m Faser zur anschließenden Regelung des Links ist ebenfalls bereits eingebaut. Die eigentliche Linkfaser befindet sich in einer temperaturstabilisierten Klimakammer, von ihr sind also nur minimale Driften zu erwarten. Die Drift, die tatsächlich zu sehen ist, wird wahrscheinlich durch die Faser auf dem Stretcher verursacht. Sie zeigt wesentlich weniger und kleinere Ausschläge als in den vorhergehenden Messungen zu sehen waren, weil der Piezostretcher mit Schaumstoff gegen Temperaturänderungen isoliert ist.

Das Fehlersignal, also die Differenz aus den beiden Messungen, zeigt auch hier keine Überraschungen. Die peak-to-peak Abweichung zwischen den Detektoren beträgt im Messzeitraum 3,94 fs und die Standardabweichung 0,69 fs.

7. Entwurf einer digitalen Regelung

Der Entwurf und die Untersuchung des neuen Messprinzips stellen nur den ersten Schritt zur Fertigstellung eines stabilisierten Faserlinks dar. Die Stabilisierung beinhaltet außerdem noch die Auswahl eines geeigneten Aktuators, die Auslegung und Implementierung eines Reglers sowie die dazu nötigen Untersuchungen. Diese Schritte werden in den nächsten Abschnitten dargelegt.

Der Reglerentwurf orientiert sich eng an der bereits bei DESY vorhandenen und überwiegend eingesetzten Hardware. Gegenwärtig ist bei FLASH für viele Regelungsaufgaben ein standardisiertes System auf der Basis von Crates mit einer VMEbus-Backplane (VERSA Module Eurocard bus) im Einsatz. Neben verschiedenen Karten für Spezialaufgaben ist für allgemeine Regelungsaufgaben eine Kombination aus einer DSP-basierten Recheneinheit und per GigaLink angebotenen ADC- beziehungsweise DAC-Modulen üblich. Eine bereits in das Beschleunigerkontrollsystem integrierte Implementierung eines digitalen PID-Reglers auf Basis dieser Karten existiert bereits.

Der VMEbus ist ein, trotz seines Alters, in der Industrie weit verbreiteter, paralleler Rückwandbus, der im Jahr 1979 erstmals vorgestellt wurde. Der Bus setzt eine Multi-Master Architektur um. Die Busarbitrierung wird durch einen Arbitrer, der üblicherweise im ersten Slot der Backplane sitzt, vorgenommen. Datentransfers werden effizient per DMA (Direct Memory Access) abgewickelt. Da Neuentwicklungen im VME Bereich aufgrund des Alters immer seltener sind, wird eine Migration auf den fortschrittlicheren ATCA- (Advanced Telecommunications Computing Architecture) beziehungsweise μ TCA-Standard im Moment bei DESY aktiv vorangetrieben. Dieser Standard wird maßgeblich von der Telekommunikationsindustrie entwickelt und eingesetzt. Er basiert auf verbreiteten Technologien, wie Gigabit-Ethernet und PCI-Express und erfüllt die gestellten Anforderungen an Verfügbarkeit, Skalierbarkeit und Redundanz. Der Rückwandbus der Crates stellt hier für jeden Einschub eine flexible Anzahl an seriellen Hochgeschwindigkeits-Punkt-zu-Punkt Verbindungen auf der Basis von PCI-Express bereit und vermeidet so einige der Nachteile des VME Standards. Momentan stehen diese Hardwarekomponenten allerdings noch nicht zur Verfügung.

Aus Kompatibilitätsgründen wird deshalb zur aktiven Regelung des Faserlinks auf vorhandene und erprobte Hardwarekomponenten zurückgegriffen. Diese Komponenten und der dort implementierte Regler werden im Folgenden zunächst vorgestellt. Im Anschluss wird kurz auf die Auswahl des Aktuators eingegangen, um dann direkt zur Untersuchung der Regelstrecke und zur Auslegung des Digitalreglers überzugehen. Die

Stabilität des entworfenen Reglers wird sowohl am Modell als auch am realen System untersucht. Zuletzt werden Ergebnisse einer Langzeitmessung mit einer geregelten Übertragungstrecke präsentiert.

7.1. Hardware des digitalen Reglers

Der verwendete DSP ist ein mit 164 MHz getakteter DSP der Firma Texas Instruments vom Typ TMS320C6701. Er verfügt über 16 MB SDRAM sowie 1 MB Flash ROM auf dem VME Board. Der DSP beinhaltet unter anderem zwei 16-Bit Fließkomma-Multiplizierer und sechs Arithmetisch-Logische-Einheiten (ALU), die voneinander unabhängig Berechnungen durchführen können. Die Kommunikation über jeweils vier der insgesamt acht GigaLink Schnittstellen sowie den VME-Rückwandbus wird durch je einen Spartan-II FPGA abgewickelt. Die GigaLink Schnittstellen dienen zur schnellen Anbindung von E/A-Karten, in diesem Fall ADC- und DAC-Modulen. Bei GigaLink (oder auch G-Link) handelt es sich um von Agilent entwickelte Chips, zum Aufbau von seriellen Hochgeschwindigkeits-Direktverbindungen. Die Steuerung des DSP, also beispielsweise das Laden von neuem Programmcode, das Setzen von Parametern im Programm oder das Auslesen von Daten für das Kontrollsystem, erfolgt über den deutlich langsameren Rückwandbus.

Bei der eingesetzten Fast-ADC Karte mit acht Kanälen handelt es sich um ein individuell bei DESY entwickeltes Board. Es werden Analog-Digital-Converter vom Typ AD9240 der Firma Analog Devices verwendet. Diese ADC bieten eine nominelle Auflösung von 14 Bit und im Zusammenspiel mit dem übrigen Board eine Abtastrate (Samplingrate) von bis zu 10 MSamples pro Sekunde. Ihr Eingangsspannungsbereich sowie die Kopplung und der Eingangswiderstand sind variabel gestaltet und können über Jumper für jeden Kanal individuell eingestellt werden. Von der Spannungsreferenz auf dem Board kann ein Messbereich von $\pm 5\text{ V}$ oder $\pm 1\text{ V}$ abgeleitet werden und der Eingangswiderstand kann $50\ \Omega$ oder $1\text{ k}\Omega$ betragen. Die ADCs werden von einem Altera FPGA gesteuert und ausgelesen. Die abgetasteten Spannungswerte werden zunächst auf dem Board zwischengespeichert. Ein weiterer FPGA wickelt die Kommunikation über den VME-Rückwandbus und über eine GigaLink Schnittstelle, auf einer optionalen Tochterkarte, ab. Diese schnelle Schnittstelle wird üblicherweise zur Verbindung mit dem zuvor beschriebenen DSP-Board eingesetzt.

Das sogenannte DAC8 Board ist ebenfalls eine VME Karte. Die acht Ausgangskanäle werden jeweils durch 14 Bit DACs vom Typ THS 5671 des Herstellers Texas Instruments bereitgestellt. Die Ansteuerung erfolgt auch hier über zwei FPGAs, wobei jeweils ein FPGA vier Ausgangskanäle steuert und über den Rückwandbus mit anderen Karten im VME-Crate kommunizieren kann. Ein GigaLink Interface wird auch bei diesem Modul über eine Tochterkarte ermöglicht. Es ist aufgrund der Geschwindigkeit die bevorzugte

Anbindungsmöglichkeit an das DSP-Modul. Die Ausgangsverstärker stellen $\pm 6,25\text{ V}$ an $10\text{ k}\Omega$ bereit und die Karte kann extern mit maximal 30 MHz getaktet werden.

Die Verbindung des DSP-Boards zum Kontrollsystem übernimmt ein Server, der auf der CPU-Karte im VME-Crate läuft. Konkret kommt eine SPARC-CPU mit Solaris als Betriebssystem zum Einsatz. Der Server ermöglicht das Laden der Firmware (also des eigentlichen Programmcodes) in den DSP sowie rudimentäre Kontrollfunktionen wie Starten, Stoppen oder Resetten des Boards. Außerdem überträgt der Server die im Beschleunigerkontrollsystem eingestellten Reglerparameter auf das DSP-Board und er kann dazu genutzt werden, um aktuelle Werte aus dem Board auszulesen. Die Verbindung vom Server zum Kontrollsystem erfolgt per RPC (Remote Procedure Calls) über Ethernet.

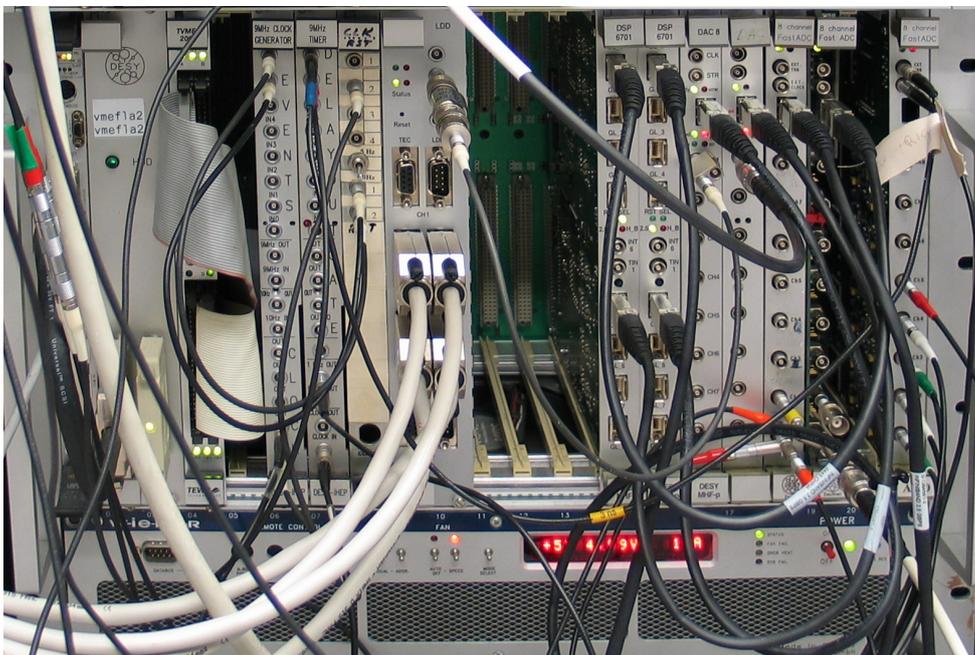


Abbildung 7.1.: VME-Crate mit Einschüben

In Abbildung 7.1 ist das zur Regelung verwendete VME-Crate zu sehen. Ganz links, von einigen Kabeln verdeckt, befindet sich die CPU-Karte, direkt daneben ein Einschub mit der Festplatte auf der das Betriebssystem und die Server gespeichert sind. Dann folgt ein Block mit einigen Modulen zur Timing- und Clock-Generierung sowie einem Treiber für Laserdioden. Die nächste Gruppe mit 6 Karten beinhaltet von links nach rechts jeweils zwei DSP-, zwei DAC- und zwei ADC-Karten, die per GigaLink verbunden sind. Im letzten Steckplatz befindet sich eine weitere ADC Karte, die zum Einlesen von Signalen unabhängig vom DSP-Regler verwendet werden kann.

7.2. Implementierung des digitalen Reglers

Die aktuelle Softwareimplementierung unterstützt bis zu zwei PID-Regler je DSP-Karte. Die Regler wurden mit einem großen Funktionsumfang implementiert, um für verschiedene Einsatzszenarien geeignet zu sein. Die Signalverarbeitung erfolgt, indem die vom ADC eingelesenen, unipolaren 14 Bit Werte zunächst über Addition eines Offsets in bipolare Werte umgewandelt werden. Die eingelesenen Werte können nun mit Hilfe eines Mittelwertfilters geglättet werden. Der Istwert kann aus zwei gewichteten ADC Kanälen kombiniert werden. Danach werden verschiedene, optionale Filter auf die eingelesenen Daten angewendet. Es stehen folgende Digitalfilter in der angegebenen Reihenfolge zur Verfügung:

Notch-Filter ein digitaler Kerbfilter zur Unterdrückung unerwünschter Resonanzen

IIR Filter ein digitaler Tiefpass erster Ordnung mit unendlicher Impulsantwort

FIR Filter ein Filter mit endlicher Impulsantwort und frei wählbarer Ordnung

Nach der Bestimmung der Regeldifferenz erfolgt die Berechnung der Stellgröße im eigentlichen Digitalregler. Sowohl die Stellgröße als auch der Integrator des Reglers können in ihrem Wertebereich eingeschränkt werden. Die berechnete Stellgröße wird über das DAC-Modul ausgegeben.

7.2.1. Der digitale PI-Regler

Zur Regelung des Faserlinks wird, wie in Unterabschnitt 7.4.3 erläutert, ein digitaler PI-Regler ausgewählt. Als Referenz wird daher im Folgenden kurz auf die konkrete Implementierung des PI-Reglers eingegangen. Die ideale Übertragungsfunktion $G_R(s)$ eines PI-Reglers im Bildbereich lautet [vgl. LW05, S. 478]

$$G_R(s) = K_p \cdot \left[1 + \frac{1}{s \cdot T_N} \right] \quad (7.1)$$

Sie wird in der vorliegenden Implementierung durch folgende Summenformeln angenähert.

$$I_k = I_{k-1} + K_i \cdot e_k \quad (7.2)$$

$$u_k = e_k \cdot K_p + I_k \quad (7.3)$$

In Gleichung 7.2 wird der aktuelle Wert des Integrators I_k berechnet. e_k entspricht der aktuellen Regelabweichung und u_k der momentanen Stellgröße. Die Übertragungsfunktion des Reglers im Z-Bereich lautet hier

$$G_R(z) = \frac{u(z)}{e(z)} = \frac{(K_i + K_p) \cdot z - K_p}{z - 1} \quad (7.4)$$

Der numerische Integrator entspricht einer Integration nach der Rückwärts-Rechteckregel. Die Möglichkeit der Zusammenfassung zu einer einzigen Summenformel wurde bei der Implementierung offensichtlich nicht berücksichtigt. Die Proportionalverstärkung K_p hat den gleichen Wert wie bei einem kontinuierlichen Regler, der Integrierbeiwert K_i ist durch

$$K_i = K_p \cdot \frac{T_s}{T_N} \quad (7.5)$$

definiert. Da der Algorithmus des Integrators bei jedem Abtastschritt durchlaufen wird, muss die Samplingfrequenz bei der Integration berücksichtigt werden. Der Parameter K_i ist sonst auf eine Sekunde normiert und nicht auf einen Abtastschritt. Die Berücksichtigung der Proportionalverstärkung K_p entspricht einer Reglerimplementierung in additiver Form, T_N bezeichnet die Nachstellzeit des Integrators.

Es muss außerdem beachtet werden, dass die am ADC mit 14 Bit bei ± 1 V eingelesenen Werte am ebenfalls 14 Bit breiten DAC auf $\pm 6,25$ V skaliert werden. Der Digitalregler hat also eine implizite Spannungsverstärkung von $v_U = 6,25$ V/V. Die Abtastzeit des gesamten vorliegenden DSP-Systems konnte zu $f_s = 100$ kHz bestimmt werden. Die Sampling- oder Abtastzeit beträgt somit $T_s = 1/f_s = 10$ μ s. Sie bleibt vermutlich aufgrund des auch an anderen Stellen kaum optimierten DSP Codes hinter den Möglichkeiten des Systems zurück.

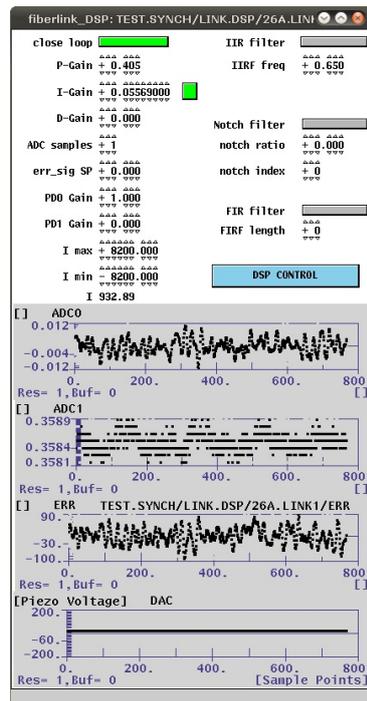


Abbildung 7.2.: Bedienpanel des PID Reglers im Kontrollsystem

7.2.2. Das digitale IIR-Filter

Der vorliegende digitale IIR Tiefpass erster Ordnung stellt eine einfache und kostengünstige Möglichkeit dar, die Regelungsbandbreite ohne weitere Veränderungen am System einzuschränken. Der Zusammenhang zwischen dem Filterkoeffizienten a_0 und der -3 dB Grenzfrequenz des Filters f_c ist nicht unmittelbar ersichtlich und wird deshalb kurz hergeleitet. Die Summenformel des implementierten Filters lautet

$$y_k = a_0 \cdot x_k + (1 - a_0) \cdot y_{k-1} \quad (7.6)$$

Diese Formel ist auch als exponentieller, gleitender Mittelwert bekannt. Die Zeitkonstante τ und somit auch die Grenzfrequenz des Filters lassen sich anhand der Sprungantwort berechnen. Diese folgt mit jedem Abtastschritt der Sprungantwort eines PT1 Elements oder Tiefpassfilters erster Ordnung. Die Zeitkonstante lässt sich annähern, indem man den ersten Abtastschritt mit der Sprungantwort eines kontinuierlichen Tiefpassfilters gleichsetzt. Aus dieser kann dann die äquivalente Grenzfrequenz f_c im Zeitbereich bestimmt werden. Die Antwort des Digitalfilters auf einen Einheitssprung nimmt nach einem Abtastschritt mit der Schrittweite T_s genau den Wert a_0 an. Setzt man diese Werte in die Gleichung der Sprungantwort eines idealen Tiefpassfilters, so erhält man

$$a_0 = 1 - e^{-\frac{T_s}{\tau}} \quad (7.7)$$

Umgeformt nach τ und eingesetzt in die allgemeine Formel für die Grenzfrequenz eines RC-Tiefpassfilters erster Ordnung ergeben sich die folgenden Näherungen für f_c beziehungsweise a_0

$$a_0 = 1 - e^{-2 \cdot \pi \cdot \frac{f_s}{f_c}} \quad (7.8)$$

$$f_c = f_s \cdot \frac{-\ln(1 - a_0)}{2 \cdot \pi} \quad (7.9)$$

Mit dieser Formel kann man die Grenzfrequenz des Digitalfilters zumindest abschätzen.

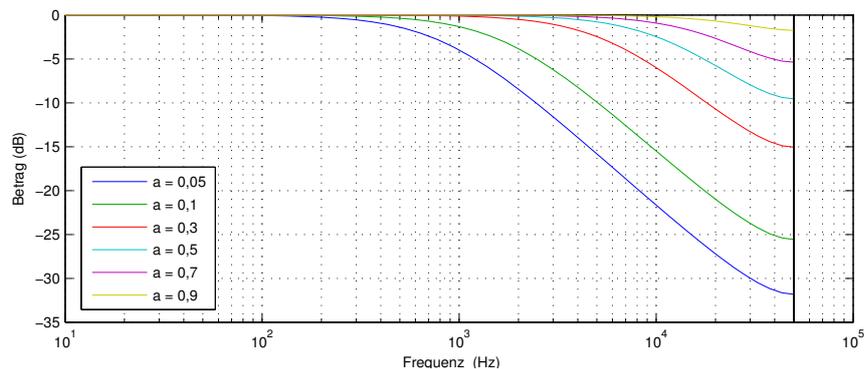


Abbildung 7.3.: Amplitudengang des digitalen IIR-Filters, G_{IIR}

In Abbildung 7.3 wird der Phasengang der Übertragungsfunktion G_{IIR} des Filters bei der gegebenen Samplingrate von $f_s = 100$ kHz und für verschiedene Werte des Filterparameters a_0 dargestellt. Die Näherung verliert an Genauigkeit, je näher die Grenzfrequenz des Filters an die Nyquistfrequenz des Digitalreglers kommt. Im Vergleich zu einem idealen Filter wird der Amplitudengang des Digitalfilters dort durch die Abtastung angehoben. Allerdings sind Grenzfrequenzen in der Nähe der Nyquistfrequenz wegen der geringen Steilheit des Filters kaum sinnvoll, da das Filter im nutzbaren Frequenzbereich der Digitalregelung kaum noch Dämpfung aufweist.

7.3. Der Aktuator

Für Aktuatoren in Faserlinks gibt es im Wesentlichen zwei Optionen, die sich ergänzen und deshalb meist zusammen eingesetzt werden. Für langsame Eingriffe mit einem großen Stellbereich eignen sich die bereits vorgestellten Verzögerungsstrecken auf der Basis von motorisierten Schlitten und Spiegeln, auch Delayline genannt. Für schnellere Regeleingriffe werden oft Faserstretcher auf Piezobasis verwendet. Solche Stretcher bieten eine gute Reproduzierbarkeit bei einer, im Vergleich zu Motoren, hohen Bandbreite, haben jedoch einen kleineren Stellbereich. Der limitierende Faktor bei solchen Stretchern ist der begrenzte Hub des Piezos in Verbindung mit der Gesamtlänge der auf dem Aktuator fixierten Faser.

Zur Einstellung des Arbeitspunktes und zum Ausgleich von Driften außerhalb des Stellbereiches des Piezostretchers kommt die bereits in Abschnitt 4.2 beschriebene und im Rahmen dieser Diplomarbeit entwickelte Delayline auf Basis eines Retroreflektors zum Einsatz. Ihr Stellbereich beträgt etwa 750 ps. Ein auf dem VME Crate laufender Server führt den Schlitten immer dann nach, wenn die Piezospaltung einen vorgegebenen Bereich verlässt. So wird sichergestellt, dass der Piezo nie die Spannungsbegrenzung von ± 200 V erreicht.

Faserstretcher mit ausreichend großem Stellbereich um den Motor zu entlasten und einer Dynamik bis in den kHz-Bereich werden beispielsweise von den Firmen Optiphase und Evanescent Optics Inc. angeboten. Die Bauformen der beiden Stretcher sind leicht verschieden und unterscheiden sich maßgeblich in der Länge der auf dem Aktuator aufbrachten Faser und ihrem Hub. Während der Stretcher Modell 915 des Herstellers Evanescent Optics Inc. mit einer Faserlänge von etwa 10 m arbeitet und so einen Stellbereich von ungefähr 1 ps bei einer Spannung von ± 200 V erreicht, wird beim Stretcher PZ2 des Herstellers Optiphase bei der gleichen Spannung und 40 Metern Faser ein Stellbereich von ungefähr 7,6 ps abgedeckt. Zusätzlich kann dieser Stretcher auch mit einer doppelt so großen Spannung noch arbeiten.

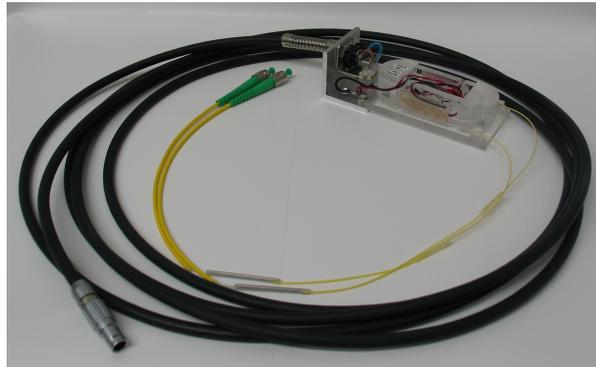


Abbildung 7.4.: Piezostretcher (Evanescent Optics Inc. - Modell 915, 5 Wicklungen)

Für den Versuchsaufbau wird auf das Modell 915 des Herstellers Evanescence Optics Inc. wegen der um Faktor 4 kürzeren Faser zurückgegriffen. Eine zusätzliche Temperaturstabilisierung des Piezostretchers mit der aufgewickelten Faser ist bei den folgenden Versuchen nicht möglich, sodass die am Piezostretcher auftretenden Faserdriften als zusätzliche, nicht steuerbare Störgröße am Streckenausgang eingehen. Vor allem mit Blick auf die Gesamtlänge des Testlinks von 20 m wird der Aktuator mit der kürzeren Faser ausgewählt. Der Hersteller spezifiziert den Hub des Stretchers pro Wicklung mit 12 nm/v . Der eingesetzte Stretcher besitzt 40 Wicklungen, was einem theoretischen Hub von 960 fs entspricht. Dieser Wert kann in einer eigenen Kalibration mit $11,5 \text{ nm/v}$ je Wicklung nahezu bestätigt werden. Dieser Stellbereich reicht aus, um kleine Temperatursprünge sowie Driften im Rahmen der Temperaturschwankungen im Labor ohne Verwendung der Delayline auszugleichen. Abbildung 7.4 zeigt einen bereits mit Anschlüssen und einer Halterung versehenen Stretcher aus der gleichen Baureihe, allerdings mit 5 statt 40 Faserwicklungen.

Um einen ersten Überblick über die zu erwartenden Resonanzen des Piezostretchers zu erhalten, wird die Übertragungsfunktion eines Stretchers Typ 915 bestimmt. Detaillierte Daten des Herstellers liegen hierüber nicht vor. Da der tatsächlich verwendete Stretcher mit 40 Wicklungen erst später geliefert wurde, wird diese Untersuchung an einem vorhandenen Stretcher mit nur 5 Wicklungen vorgenommen. Die gewonnenen Daten sind somit nur zur groben Orientierung geeignet. Frequenz- und Phasengang werden mit Hilfe eines Vector Signal Analyzers (VSA) des Herstellers HP (Modell 89410A) bestimmt. Der VSA beaufschlagt das Testobjekt mit weißem Rauschen und wertet das komplexe Übertragungsverhalten im Frequenzbereich aus. Die Messschaltung zur Bestimmung von G_{el} ist in Abbildung 7.5 skizziert.

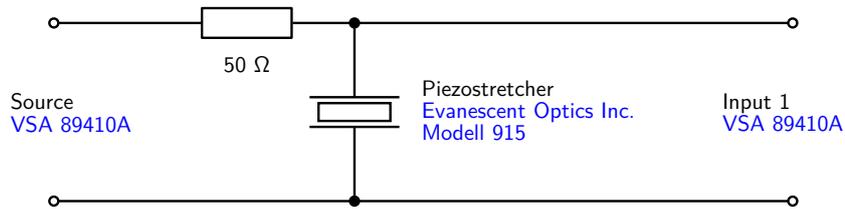


Abbildung 7.5.: Messschaltung für die elektronische Übertragungsfunktion des Piezostretchers

Das Bodediagramm der elektronischen Übertragungsfunktion G_{el} ist in Abbildung 7.6 zu sehen. Setzt man am Regler funktionierende Anti-Aliasing- und Rekonstruktionsfilter voraus, so ist nur die erste Resonanz bei etwa 37 kHz unterhalb der Nyquistfrequenz der Digitalregelung von 50 kHz relevant. Der zur Ansteuerung verwendete, bipolare

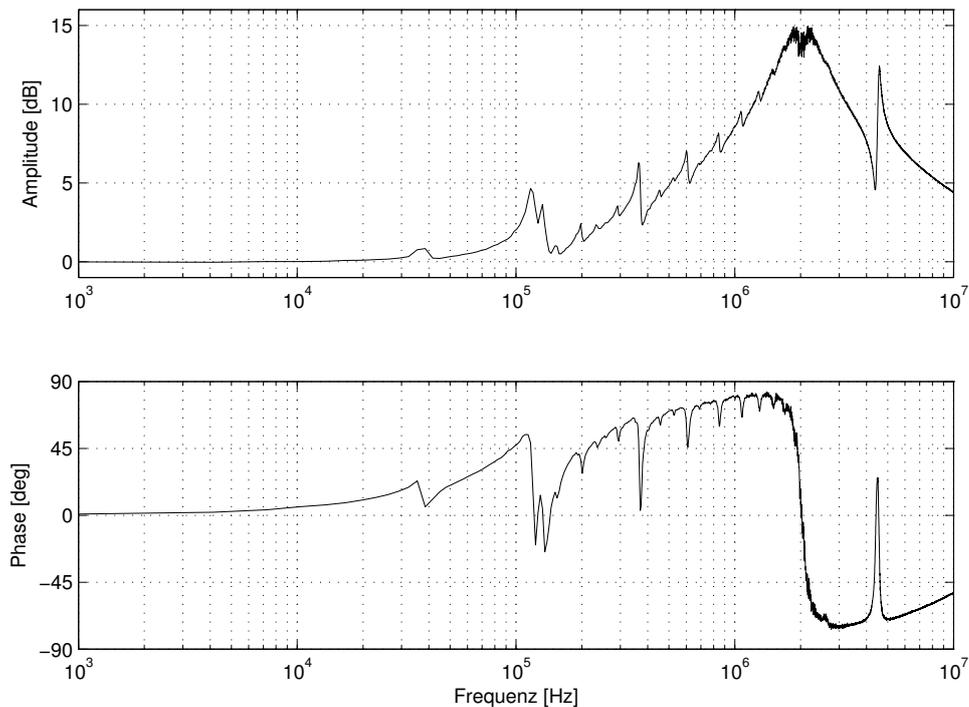


Abbildung 7.6.: Bodediagramm der Übertragungsfunktion G_{el} des Piezostretchers (Evanescence Optics Inc. - Modell 915, 5 Wicklungen)

7. Entwurf einer digitalen Regelung

Piezotreiber wurde bei DESY entwickelt. Die maximale Ausgangsspannung beträgt ± 200 V. Sie wird bei einer Eingangsspannung von $\pm 6,25$ V erreicht, was genau auf den Spannungshub des verwendeten DACs abgestimmt ist. Der Treiber besitzt einen Monitorausgang für die Hochspannung mit einer Spannungsübersetzung von 5 mV/V . Die Spannungsversorgung erfolgt über ein externes Netzteil mit ± 150 V Gleichspannung. Abbildung 7.7 zeigt ein Foto des Treibers einschließlich Netzteil.

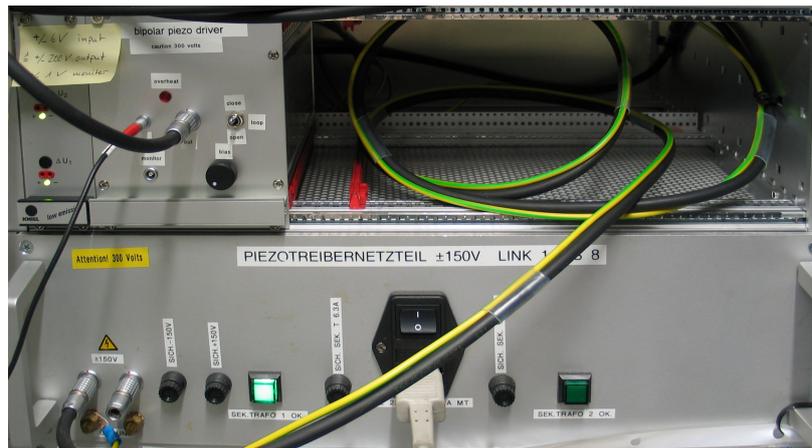


Abbildung 7.7.: Bipolarer Piezotreiber (oben links) mit Netzteil (unten)

7.4. Systemcharakterisierung und Reglerauslegung

Wie bereits in Kapitel 5 erläutert wurde, wird angestrebt, die Pulse des Master Laser Oszillators (MLO) möglichst driftfrei zum Ende eines Faserlinks zu transportieren. Ein Teil der Leistung jedes Pulses wird aus diesem Grund am Linkende zurückreflektiert. Am Linkanfang kann nun kontinuierlich die Phasenverschiebung jedes zurückreflektierten Pulses zu einem direkt vom MLO kommenden Referenzpuls verglichen werden. Temperaturdriften der Linkfaser verursachen Änderungen in der Länge und im Brechungsindex der Faser die zu Phasen- beziehungsweise Laufzeitveränderungen führen [vgl. Bou09, S. 97 ff.]. Mit Hilfe geeigneter Stellglieder können diese Laufzeitveränderungen kompensiert werden. Hält man nun die Laufzeit durch eine Regelung konstant, so kann angenommen werden, dass die Pulse auch am Linkende einen konstanten zeitlichen Bezug zum MLO besitzen.

Da die Messgröße nur indirekt über den Detektor zugänglich ist, wird dieser in die Charakterisierung der offenen Strecke einbezogen und nicht wie sonst üblich im Rückkopplungszweig berücksichtigt. Eine getrennte Untersuchung des Zeitverhaltens des Detektors und des Aktuators ist nicht möglich, da Regel- und Stellgröße nicht individuell zugänglich sind. Für den Reglerentwurf ist eine Umrechnung der gemessenen

Spannungen in echte Timingänderungen nicht zwingend nötig. Der Regler kann komplett als Spannungsregler entworfen werden. Ein Sollwert von 0 V beziehungsweise bei digitaler Vorgabe 0 digit entspricht bei korrekt kalibriertem Aufbau einer konstanten Phasenverschiebung zwischen den aus dem Link zurückkommenden und den Referenzpulsen von 180 Grad (zur Kalibrierung siehe auch Abschnitt 5.6). In der Simulation und zu Diagnosezwecken kann die gemessene Spannung am Ausgang der Regelstrecke mit Hilfe der zuvor bestimmten Kalibrationskonstante K_φ in konkrete Timingwerte umgerechnet werden.

Zur Simulation wird parallel zur Untersuchung ein Modell des geregelten Systems in Simulink entwickelt. Weitere Informationen zu diesem Modell sind in Anhang A zusammengestellt und das Modell selbst befindet sich im digitalen Anhang zu dieser Diplomarbeit. Wird im Folgenden von Simulationsergebnissen gesprochen, so sind Ergebnisse aus diesem Modell gemeint.

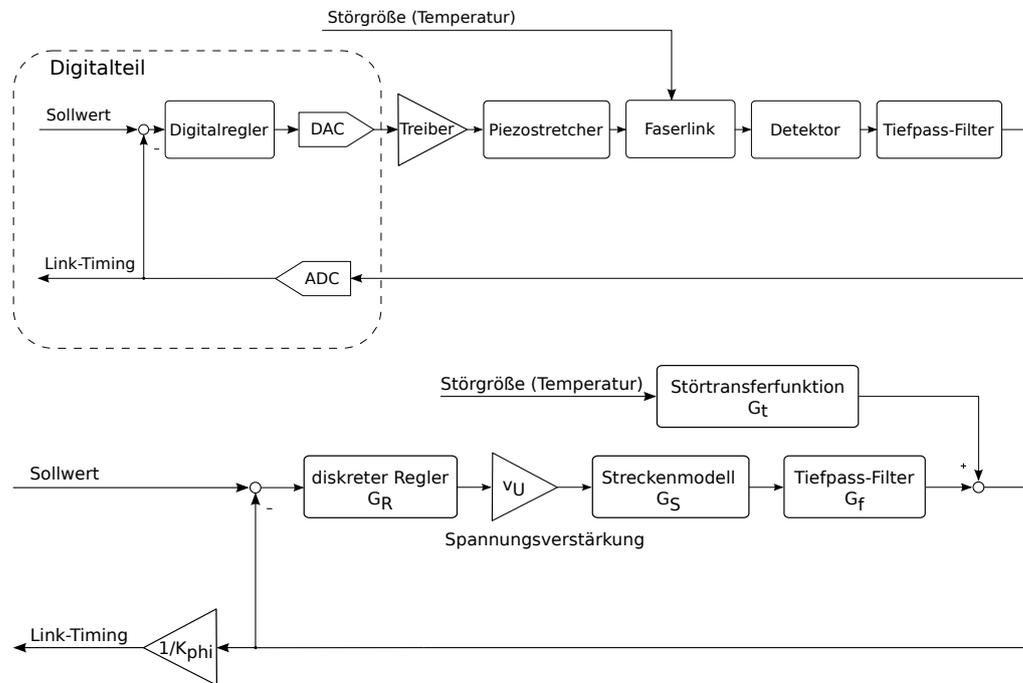


Abbildung 7.8.: Die aufgebaute Regelung (oben) und das angenäherte Modell zur Simulation (unten)

Der zu entwerfende Regler soll robust gegenüber dem Einfluss von Temperaturschwankungen an der Linkfaser sein, das Führungsverhalten ist nur von nachrangigem Interesse. Der Regler soll schnelle Änderungen bis zu einigen kHz unterdrücken, wobei langsame Driften durch die Delayline nachgeführt werden, bevor der Piezostretcher das Ende

seines Stellbereiches erreicht. Obwohl die Faserlinks im Produktivbetrieb bei FLASH die maximal zehn- bis zwanzigfache Länge des 20 m langen Testlinks haben, kann davon ausgegangen werden, dass eine sprunghafte Erwärmung nicht über die ganze Faserlänge sondern bestenfalls lokal auftritt. Eine Unterdrückung von Timingänderungen durch Temperatursprünge von 1 K an der 20 m langen Testfaser, was etwa 2 ps entspricht, mit einem Restfehler von unter kurzzeitig 50 fs wird somit als ausreichend angesehen. Abbildung 7.8 zeigt einen Überblick über den tatsächlichen Regelkreis und das dafür angesetzte Modell.

7.4.1. Untersuchung der offenen Strecke

Für den eigentlichen Reglerentwurf wird zunächst das Verhalten der offenen Strecke betrachtet. Dies beinhaltet das Übertragungsverhalten des Piezotreibers, des Piezostretchers, der eigentlichen Strecke und des Detektors (siehe Kapitel 5). Zur Identifizierung des Übertragungsverhaltens wird zunächst die Sprungantwort bestimmt. Dazu wird der Piezotreiber mit einem Rechteckgenerator angeregt und sowohl das Anregungs-, als auch das Ausgangssignal am Detektor mit einem digitalen Speicheroszilloskop aufgezeichnet. Die Ergebnisse des Sprungversuchs sind in Abbildung 7.9 zu sehen. Die Kalibrationskonstante beträgt, wenn nicht anders angegeben, bei allen Messungen in diesem Kapitel $K_\varphi = 1,8925 \text{ mV/fs}$. Das System zeigt deutliches Überschwingen, ist aber noch stabil und pendelt sich auf einen stationären Endwert ein.

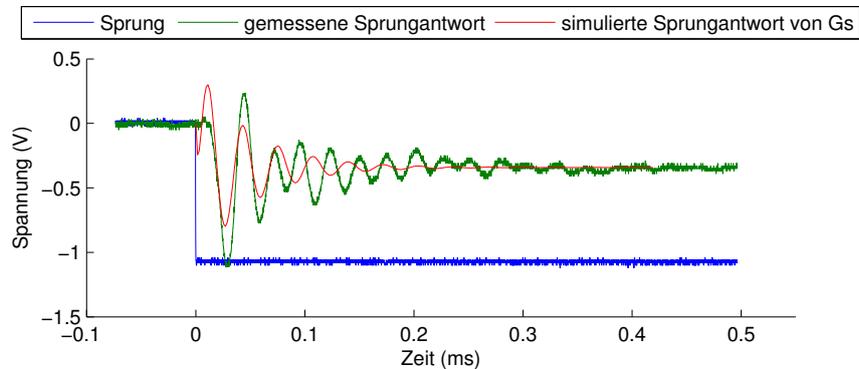


Abbildung 7.9.: Sprungantwort der offenen Strecke

Auf eine physikalische Modellbildung wird verzichtet, da zu viele Parameter des Systems und insbesondere des Aktuators unbekannt sind. Stattdessen wird mit Hilfe der System Identifikation Toolbox von Matlab ein Zustandsraummodell eines Systems dritter Ordnung an das reale System angenähert, das ein ähnliches Verhalten im Zeitbereich wie die experimentell bestimmte Sprungantwort zeigt. Das System wird als linear und

zeitinvariant, also als LTI-System, angenommen. Die Temperatur wird erst später als Ausgangsstörgröße in die Modellierung einbezogen.

Die allgemeinen Gleichungen für das Zustandsraummodell eines LTI-Systems mit den Eingängen $\vec{u}(t)$, den inneren Zuständen $\vec{x}(t)$ und den Ausgängen $\vec{y}(t)$ lauten [vgl. Lun08, S. 94]:

$$\begin{aligned}\dot{\vec{x}}(t) &= \mathbf{A} \cdot \vec{x}(t) + \mathbf{B} \cdot \vec{u}(t) \\ \vec{y}(t) &= \mathbf{C} \cdot \vec{x}(t) + \mathbf{D} \cdot \vec{u}(t)\end{aligned}\quad (7.10)$$

\mathbf{A} entspricht hier der Systemmatrix und \mathbf{B} der Steuermatrix. \mathbf{C} bezeichnet die Beobachtungsmatrix und \mathbf{D} die Durchgangsmatrix. Das angenäherte Modell besitzt die folgenden Parameter:

$$\mathbf{A} = \begin{bmatrix} -5272 & -1,896 \cdot 10^5 & 1,005 \cdot 10^5 \\ 1,826 \cdot 10^5 & -1,257 \cdot 10^4 & 3,885 \cdot 10^5 \\ 4652 & -4,231 \cdot 10^4 & -5,462 \cdot 10^5 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 1,588 \cdot 10^4 \\ 3,927 \cdot 10^4 \\ -5,777 \cdot 10^4 \end{bmatrix}$$

$$\mathbf{C} = [15,44 \quad -0,1703 \quad -0,7042] \quad \mathbf{D} = [0]$$

Die Sprungantwort des Modells ist zum Vergleich ebenfalls in Abbildung 7.9 eingezeichnet. Das Modell zeigt eine ausreichende Übereinstimmung mit den Messdaten, auf eine Verbesserung der Simulation durch eine höhere Ordnung des Modells wird aus Gründen der Komplexität verzichtet.

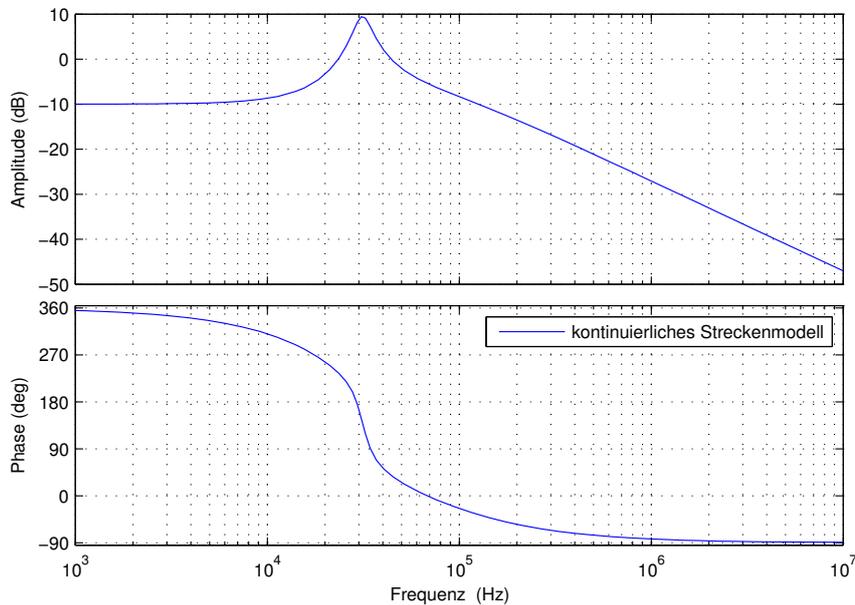


Abbildung 7.10.: Bodediagramm der angenäherten Streckenübertragungsfunktion G_S

Das Bodediagramm der Streckenübertragungsfunktion G_S in Abbildung 7.10 lässt eine schwach gedämpfte Resonanz bei 30,9 kHz erkennen, die auch schon in der zuvor gezeigten Sprungantwort als Schwingung zu sehen ist. Sie entspricht näherungsweise der in Abschnitt 7.3 bereits erwähnten kleinsten Resonanzfrequenz des Piezostretchers, wobei weiterhin zu beachten ist, dass der hier verwendete Stretcher nicht dem zuvor exemplarisch untersuchten entspricht.

Die bei der gegebenen Abtastrate mögliche Regelbandbreite von 50 kHz wird nicht benötigt, da keine derart hochfrequenten Temperaturschwankungen erwartet werden und die Faser insgesamt nur sehr träge auf Temperatursprünge anspricht (siehe Unterabschnitt 7.4.2). Auch Vibrationen sind mit diesem Frequenzbereich gut abgedeckt. Der digitale Tiefpass erster Ordnung zeigt aufgrund der geringen Ordnung und weil die Resonanz zu nah an der Nyquistfrequenz liegt keine zufriedenstellende Dämpfung in diesem Frequenzbereich. Deshalb wird nach einigen Versuchen mit dem digitalen IIR-Filter ein analoges RC-Tiefpassfilter zweiter Ordnung mit einer Grenzfrequenz von 10 kHz und der Übertragungsfunktion G_f eingesetzt. Der Einsatz eines analogen Tiefpassfilters hat Vor- und Nachteile. Er ist immer mit einer Dämpfung verbunden, da das Filter einen Spannungsteiler mit dem Eingangswiderstand des nachfolgenden Bauelements bildet. Außerdem wird das Gesamtsystem durch die Phasenverschiebung am Filter verlangsamt. Es wird angestrebt, dass das Filter im System mehrere Funktionen erfüllt, um diese Nachteile möglichst auszugleichen. Aus diesen Überlegungen folgt die Schlussfolgerung, das Filter hinter den Detektor und vor dem LNA (Low Noise Amplifier) einzubauen. Der Amplitudenverlust ist an dieser Stelle vergleichsweise gering, da der LNA einen großen Eingangswiderstand im Bereich von 100 M Ω besitzt. Die Phasenverschiebung wird begrenzt, indem nur ein Filter zweiter und nicht höherer Ordnung eingesetzt wird. Außerdem hat das Filter an dieser Position zwei weitere Funktionen. Es fungiert als Anti-Alias-Filter vor dem ADC und es verhindert, dass mehr Messrauschen als nötig über den Regler auf die Strecke zurückgeführt wird. Aus Sicht der Regelungstechnik müsste das Filter eigentlich direkt vor dem Aktuator eingebaut werden, da dort die Anregung der Resonanzfrequenz verhindert werden soll. Da die Regelung in diesem Fall nur für einen konstanten Sollwert ausgelegt werden muss und keine Sprünge oder Veränderungen der Führungsgröße vorgesehen sind, ist eine Verlegung an den Streckenausgang unproblematisch.

In Abbildung 7.11 ist ein Vergleich der Sprungantwort des Systems nach Einbau des Filters mit der Sprungantwort der Simulation mit Filter zu sehen. Der Endwert der Messung weicht leicht von der Simulation ab, da die Messwerte zu einem späteren Zeitpunkt aufgenommen wurden. Zu diesem Zeitpunkt war der Detektor neu kalibriert worden, was dazu führte, dass die Kalibrationskonstante K_φ sich auf 2,33 mV/fs verändert hat und die Verstärkung des Modells nun nicht mehr mit dem realen System übereinstimmt. Die Resonanzfrequenz wird in beiden Fällen ausreichend stark unterdrückt.

Auch im Bodediagramm (Abb. Abbildung 7.12) ist deutlich erkennbar, dass die unerwünschte Resonanz zufriedenstellend gefiltert wird.

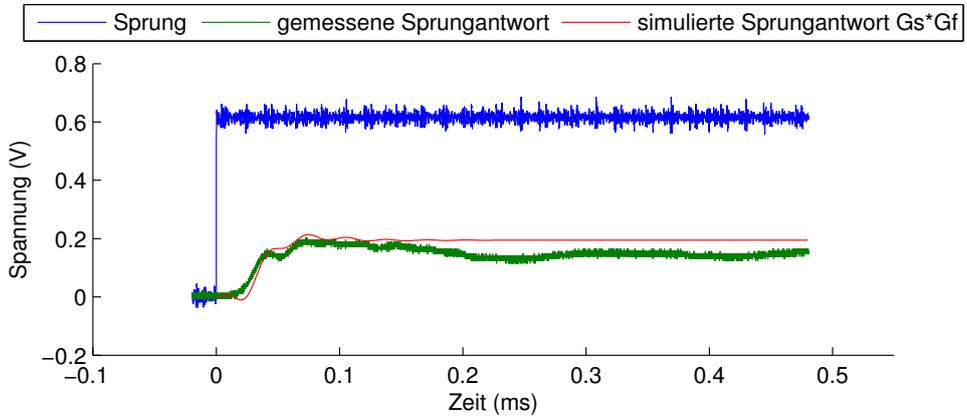


Abbildung 7.11.: Sprungantwort der offenen Strecke G_S mit Filter G_f

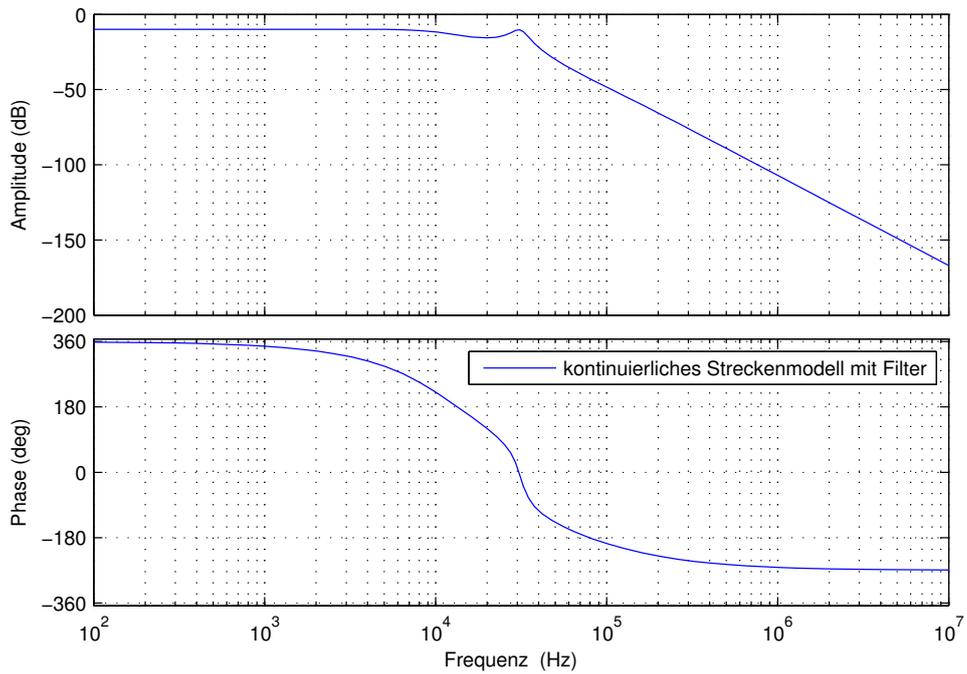


Abbildung 7.12.: Bodediagramm der offenen Strecke G_S mit Filter G_f

7.4.2. Temperaturverhalten der Faser

Um das Temperaturverhalten der Linkfaser in die Modellbildung einzubeziehen wird ihre Systemantwort auf einen Temperatursprung gemessen und daraus das Übertragungsverhalten abgeleitet. Die Linkfaser befindet sich zu diesem Zweck in einer kleinen Klimakammer, die mit einem Sollwertsprung beaufschlagt wird. Da sich außer der Faser nur Luft in der Kammer befindet und keine großen thermischen Massen geheizt werden müssen, kann die Klimakammer den Sollwert wesentlich schneller erreichen, als die Faser dem Temperatursprung folgen kann. Die Sprungantwort ist in Abbildung 7.13 abgebildet. Der Temperatursprung beträgt $\Delta T = 0,567 \text{ K}$. Die Faser folgt einem PT1-Verhalten. Die gemessene Temperaturempfindlichkeit von 112 fs/K je Meter Faser beträgt etwa das 1,5-fache des in [Bou09, S. 98] theoretisch hergeleiteten Wertes von 76 fs/K je Meter, ist jedoch in Anbetracht des unterschiedlichen Kabelaufbaus durchaus realistisch. Auch mechanische Einflüsse auf die Faser können Laufzeitveränderungen verursachen. Bei einer fest verlegten Faser kann allerdings davon ausgegangen werden, dass gegenüber Temperaturdriften nur kleine Auslenkungen, die beispielsweise durch Vibrationen verursacht werden, auftreten.

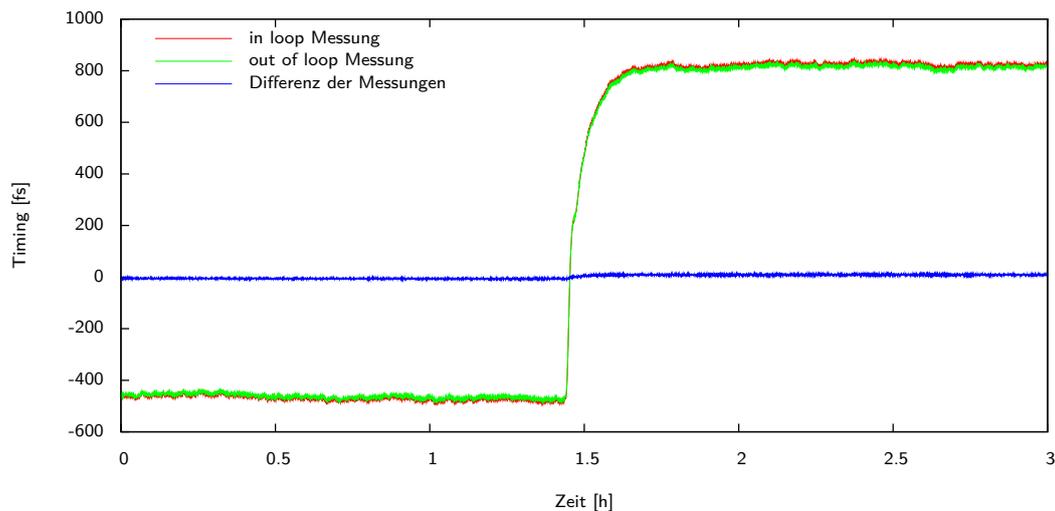


Abbildung 7.13.: Sprungantwort für Temperaturänderungen an der Linkfaser, Übertragungsfunktion G_t

Die Timingwerte im Plot sind wie in den bisher gezeigten Kapiteln auf das Linkende bezogen. Der Detektor am Linkanfang sieht also eine doppelt so große Zeitänderung. Die Timingänderung lässt sich aus den Messwerten zu $\Delta t = 1,28 \text{ ps}$ bestimmen. Die Zeitkonstante der Übertragungsfunktion wird durch grafisches Anlegen einer Tangente

an den Anfang der Sprungantwort bestimmt. Die Übertragungsfunktion lautet

$$G_t(s) = \frac{K_t}{1 + s \cdot T_t} \quad (7.11)$$

Die Parameter der PT1-Übertragungsfunktion bei 20 m Linkfaser lauten somit unter Berücksichtigung des in der Simulation verwendeten Wertes von K_φ

$$\begin{aligned} K_t &= 2 \cdot \Delta t / \Delta T \cdot K_\varphi \\ &= 4,5 \text{ ps/K} \cdot 1,8925 \text{ V/ps} = 8,52 \text{ V/K} \end{aligned} \quad (7.12)$$

$$T_t = 95,5 \text{ s} \quad (7.13)$$

Die nun bestimmte Übertragungsfunktion gibt die Übersetzung der Störgröße Temperatur in gemessene Spannung an. Diese Spannung ist proportional zu den tatsächlichen Timingänderungen und sie kann in der Simulation somit einfach am Streckenausgang auf das System aufsummiert werden. Die Berücksichtigung des Umrechnungsfaktors K_φ entspricht der in Abbildung 7.8 zu erkennenden Verlegung der Summationsstelle an den Streckenausgang. Es wird davon ausgegangen, dass das dynamische Verhalten des Detektors und des anschließenden Tiefpassfilters im Vergleich zur extrem langsamen Störübertragungsfunktion keine Rolle spielt.

7.4.3. Reglerparametrierung

Zur Reglerparametrierung muss zunächst entschieden werden, welcher Reglertyp zum Einsatz kommen soll. Im vorliegenden Fall fällt die Entscheidung auf einen PI-Regler. Der Proportionalregler verspricht eine gute Dynamik, der Integrator stationäre Genauigkeit. Auf die Verwendung des schnellen D-Anteils wird verzichtet, um den Regler unempfindlicher gegen hochfrequente Änderungen, wie beispielsweise Messrauschen, zu gestalten [vgl. LW05, S. 134].

Zur Parametrierung wird auf die Einstellregeln nach Takahashi zurückgegriffen. Sie entsprechen den bekannten Einstellregeln nach Ziegler und Nichols, dienen jedoch zur Auslegung diskreter und nicht zeitkontinuierlicher Regler. Der so entworfene Regler besitzt aufgrund der großzügigen Proportionalverstärkung nur ein mäßiges Führungsverhalten mit starkem Überschwingen, jedoch ein gutes Störverhalten, was den Anforderungen der vorliegenden Regelungsaufgabe entspricht [vgl. Sch04, S. 177]. Da die Einstellregeln nach Takahashi eigentlich nur für quasikontinuierliche Systeme gültig sind, wird die Stabilität anschließend im Modell und am echten System überprüft.

Es wird die Einstellung anhand der Stabilitätsgrenze ausgewählt. Der Regelkreis wird mit einem digitalen P-Regler geschlossen und die Proportionalverstärkung so lange erhöht bis das System schwingt. Die kritische Verstärkung beträgt $K_{krit} = 0,9$, wobei zur Einordnung dieser Größe immer die implizite Spannungsverstärkung des Reglers von $6,25 \text{ V/V}$ (siehe Unterabschnitt 7.2.1) bedacht werden muss. Die Schwingungsfrequenz

7. Entwurf einer digitalen Regelung

$f_{krit} = 11,4$ kHz kann am, zum Systemausgang parallel angeschlossenen Oszilloskop, abgelesen werden. Die Resonanz des Piezostretchers wird durch das analoge Tiefpassfilter gedämpft und ist hier nicht mehr messbar. Die tatsächlichen Reglerparameter können nun für einen digitalen PI-Regler nach [LW05, S. 487] und Gleichung 7.5 berechnet werden. Sie lauten

$$K_p = 0,45 \cdot K_{krit} = 0,405 \quad (7.14)$$

$$T_N = 0,83 \cdot T_{krit} = 72,72 \mu\text{s} \quad (7.15)$$

$$K_i = K_p \cdot T_s / T_N = 0,05569 \quad (7.16)$$

Abbildung 7.14 zeigt das Bodediagramm der offenen Strecke mit dem zuvor parametrisierten PI-Regler. Der Amplitudenrand der diskretisierten Übertragungsfunktion beträgt 3,6 dB bei einer Frequenz von 10,8 kHz, der Phasenrand 97,4 Grad bei einer Frequenz von 3,47 kHz.

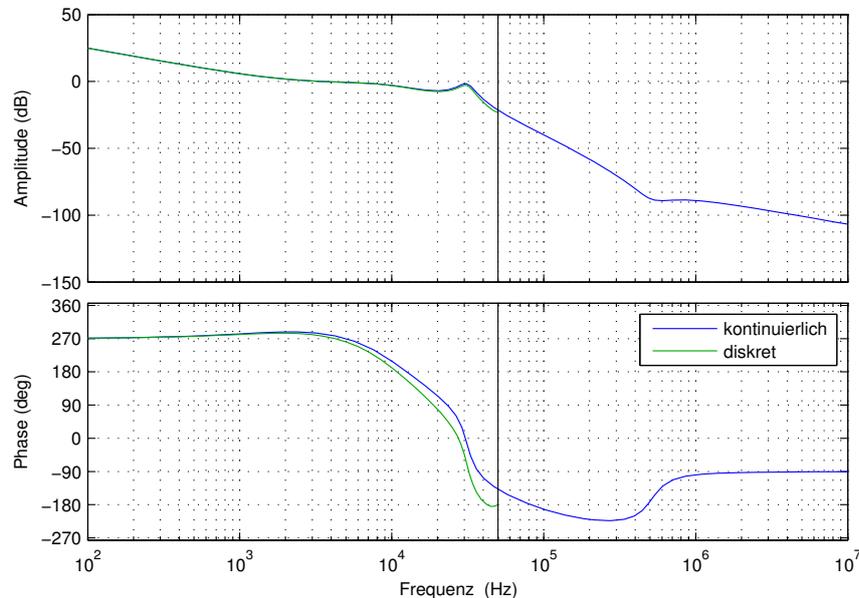


Abbildung 7.14.: Bodediagramm des Modells der offenen Strecke G_S mit Filter G_f und Regler G_R

In Abbildung 7.15 ist die Wurzelortskurve des geregelten Systems zu sehen. Der Pol nahe des Ursprungs auf der reellen Achse stammt vom I-Anteil des Reglers, alle Pole befinden sich jedoch in der linken Halbebene. Das geregelte System ist sowohl laut Bode-Diagramm als auch laut Wurzelortskurve stabil.

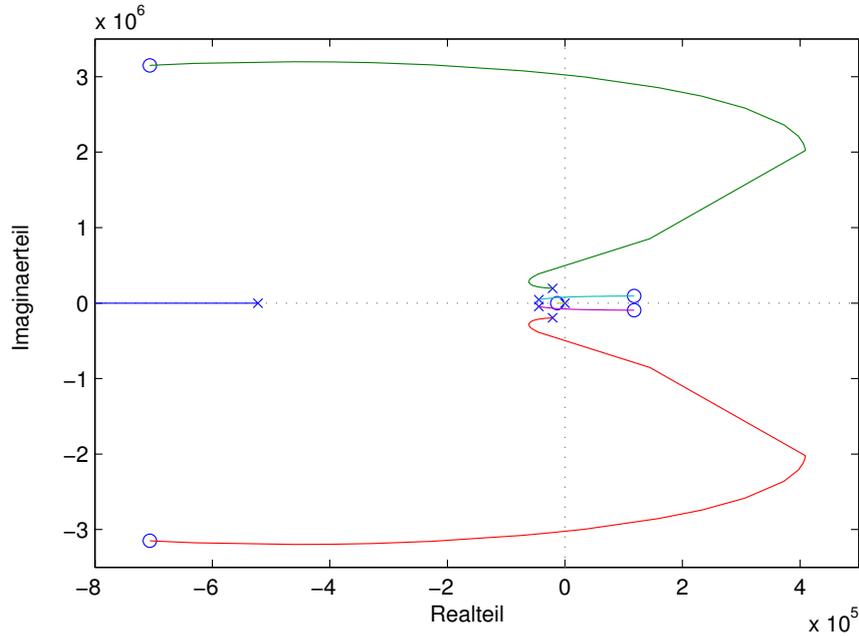


Abbildung 7.15.: Wurzelortskurve der geschlossenen Regelschleife mit den Streckenelementen G_S , G_f und G_R

7.4.4. Führungsverhalten

Das Führungsverhalten spielt für diese Regelung nur eine untergeordnete Rolle, da die Laufzeit der Laserpulse im Link konstant gehalten werden soll und der Sollwert im Betrieb somit nicht verändert werden muss. Praktisch wird aufgrund der besseren Fehlerunterdrückung mit dem optimalen Arbeitspunkt (siehe Abschnitt 5.5) als Sollwert gearbeitet. Trotzdem kann das Führungsverhalten zusätzliche Information über die Stabilität des Reglers liefern. In Abbildung 7.16 wird die Antwort des geregelten Systems auf einen Sprung der Führungsgröße von 7500 auf -3200 digit gezeigt. Dieser Sollwertsprung entspricht einem Spannungssprung von

$$U_e = 10700 \text{ digits} \cdot 0,00012207 \text{ V/digit} = 1,31 \text{ V} \quad (7.17)$$

Das System zeigt mit dieser Reglerauslegung wie bereits vermutet deutliches Überschwingen, konvergiert jedoch schnell auf den stationären Endwert. Die Simulation folgt diesem Verhalten nicht exakt, zeigt jedoch nach wie vor eine ausreichende Übereinstimmung mit dem realen System.

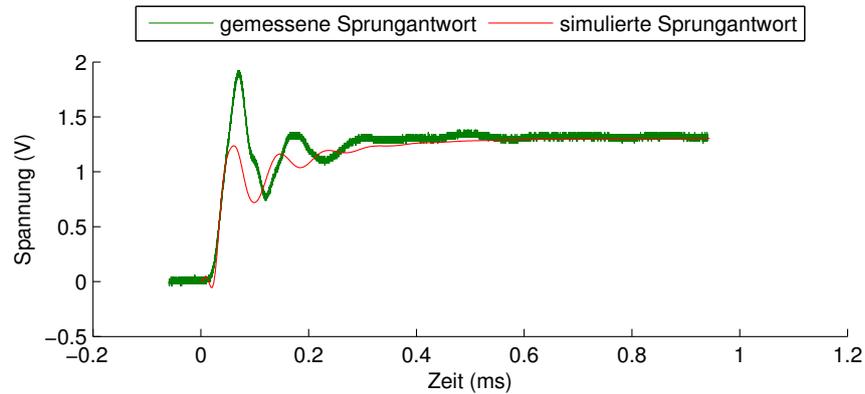


Abbildung 7.16.: Systemantwort auf eine sprunghafte Änderung der Führungsgröße, Modell aus Abbildung 7.8

7.4.5. Störverhalten

In Unterabschnitt 7.4.2 wurde bereits beschrieben, dass das System nur sehr langsam auf Temperaturänderungen reagiert. Die dort erarbeitete Übertragungsfunktion bestimmt die zeitliche Abhängigkeit des Link-Timings, gemessen in Volt, von der Temperatur. Verknüpft man diese Übertragungsfunktion mit dem vorhandenen Modell der Regelstrecke, so muss die Störgröße am Streckenausgang angreifen. Der Systemausgang wird weiterhin über einen zusätzlichen, zuvor kalibrierten Übertragungsblock in eine Zeit umgerechnet.

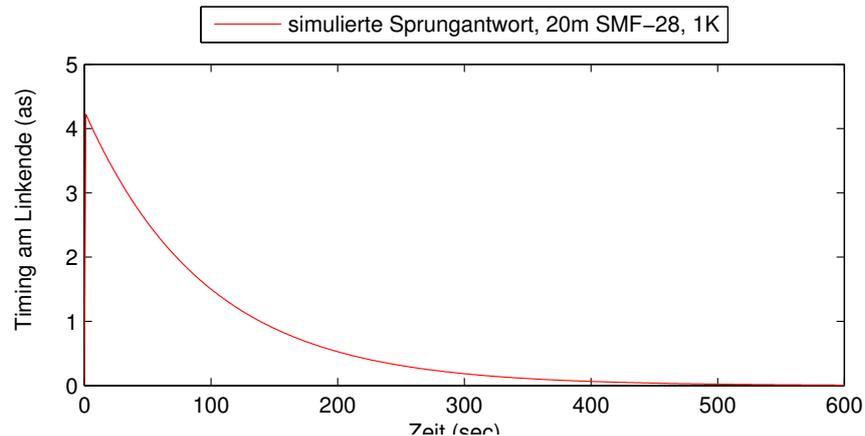


Abbildung 7.17.: Systemantwort auf eine sprunghafte Änderung der Störgröße, Modell aus Abbildung 7.8

Die in Abbildung 7.17 dargestellte Sprungantwort zeigt eine sehr lange Ausregelzeit von etwa 600 s. Der im Vergleich zur Regelgeschwindigkeit langsame Anstieg der Störgröße am Systemausgang kann in erster Näherung als Rampe interpretiert werden, die von einem einfachen Integrator nicht ausgeglet wird. Wenn die Störgröße ihrem Endwert immer näher kommt und der Anstieg der Rampe flacher wird, kann der Regler diesen Schleppfehler immer besser unterdrücken. Dieses Verhalten ist jedoch nicht von praktischer Bedeutung, da der simulierte Temperatursprung eine Höhe von 1 K hat, die maximale Regelabweichung jedoch nur 43 μ s beträgt. Im realen System liegt eine solche Abweichung außerhalb der Messauflösung. Zusätzlich kann davon ausgegangen werden, dass im Beschleunigungstunnel keine Temperatursprünge dieser Größe auf der ganzen Länge der Faser auftreten [vgl. Unterabschnitt 7.4.2].

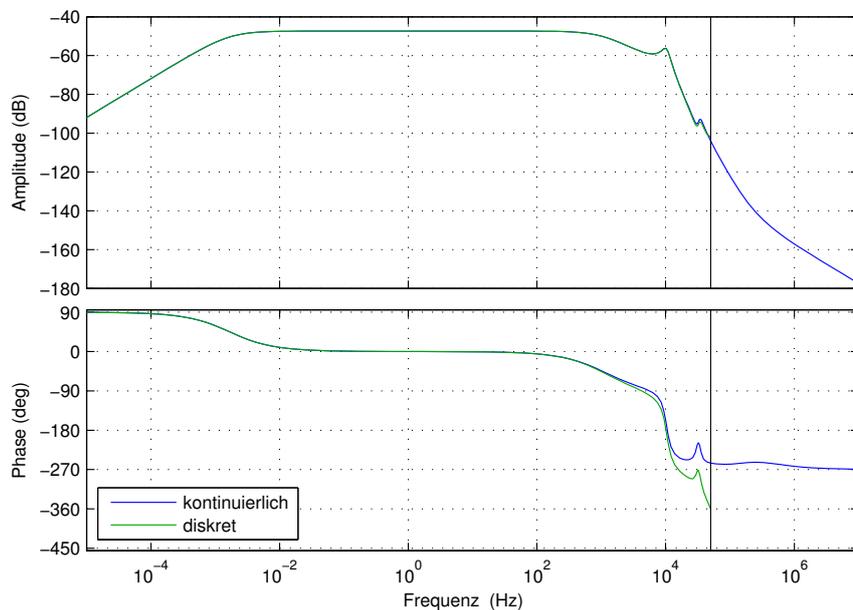


Abbildung 7.18.: Bodediagramm der Störübertragungsfunktion, Modell aus Abbildung 7.8

Im Bodediagramm der Störübertragungsfunktion in Abbildung 7.18 ist deutlich der stark gedämpfte Phasengang zu erkennen, der auf eine hervorragende Unterdrückung der Störgröße hindeutet.

7.5. Messungen mit aktiver Regelung

Um die geforderte Performance der Regelung zu zeigen, wird die 20 m lange Linkfaser in der Klimakammer mit einem Temperatursprung von 1 K beaufschlagt. Die optische

Leistung beträgt je Detektor etwa 2,4 mW. Sowohl der Digitalregler als auch das langsame Motorfeedback sind aktiv, um die Timingänderungen des Links auszuregulieren, denn der Stellbereich des Piezostretchers allein reicht dazu nicht aus. Abbildung 7.19 zeigt die in dieser Zeit aufgenommenen Messwerte. Zusätzlich zu den üblichen Daten ist mit einer zweiten y-Achse die Piezospannung aufgetragen, an der auch die Motorschritte abgelesen werden können. Das langsame Motorfeedback ist so konfiguriert, dass ab einer Piezospannung von 160 V der Motor um 3000 Schritte nachgestellt wird. Die Kalibrationskonstanten betragen $K_{\varphi,in} = 1,5 \text{ mV/fs}$ und $K_{\varphi,out} = 4,4 \text{ mV/fs}$.

Das rot dargestellt in-loop Messsignal zeigt keine Hinweise auf den erfolgten Temperatursprung, es ist völlig flach. Das geringere Rauschen ist auf den Einbau des zweiten Tiefpassfilters zur Schwingungsdämpfung zurückzuführen. Im out-of-loop Zweig ist nur das Filter zur Eliminierung der unerwünschten Mischerprodukte eingebaut und dementsprechend ein deutlich größerer Rauschbeitrag zu sehen. Das Motorfeedback hat erfolgreich verhindert, dass der Aktuator der Regelung an die Grenzen seines Stellbereichs stößt und somit auch erwartungsgemäß funktioniert.

Nicht erwartet ist jedoch die offensichtlich von der Aussteuerung des Piezoaktuators und der Position des Schlittens abhängige Beeinflussung der out-of-loop Messung, die als realer Timingfehler am Linkende zu interpretieren ist. Es wurde zuvor bereits angedeutet, dass die Motorposition über Verluste in der optischen Delayline einen Einfluss auf die Kalibration und Messung nimmt. Diese These wird hier bestätigt. Obwohl der Piezostretcher am Ende der Messung fast wieder so wie am Anfang ausgelenkt ist, zeigt sich eine bleibende Abweichung des Ausgangssignals. Die Regelung sieht diese Abweichung nicht, das in-loop Signal wird weiter auf Null geregelt. Außerdem zeigt sich eine überraschende Abhängigkeit der out-of-loop Messung von der Auslenkung des Piezostretchers. Es werden wahrscheinlich beide Messungen beeinflusst, der Messfehler wird jedoch durch die Regelung komplett auf die out-of-loop Messung verschoben. Eine konservative Abschätzung ergab einen Messfehler abhängig von der Auslenkung des Stretchers von ungefähr 20 fs/ps .

Die genaue Untersuchung der Ursache des Effekts am Piezostretcher steht noch aus. Eine Untersuchung mit einem anderen Piezostretcher eines anderen Herstellers wird ein erster Schritt in diese Richtung sein. Auch optische Verluste im Stretcher wurden nicht untersucht und können somit nicht ausgeschlossen werden, obwohl diese in erster Näherung durch das Messprinzip kompensiert werden. Darüber hinaus gibt es eine Reihe nichtlinearer Effekte in Glasfasern, wie beispielsweise Polarisationsmodendispersion (PMD), deren Einfluss auf die Messung noch unbekannt ist.

Trotz dieser Probleme liegt die hier gezeigte Abweichung am Linkende von 10 fs bei einem Temperatursprung von 1 K an einem 20 m Testlink innerhalb der geforderten Genauigkeit.

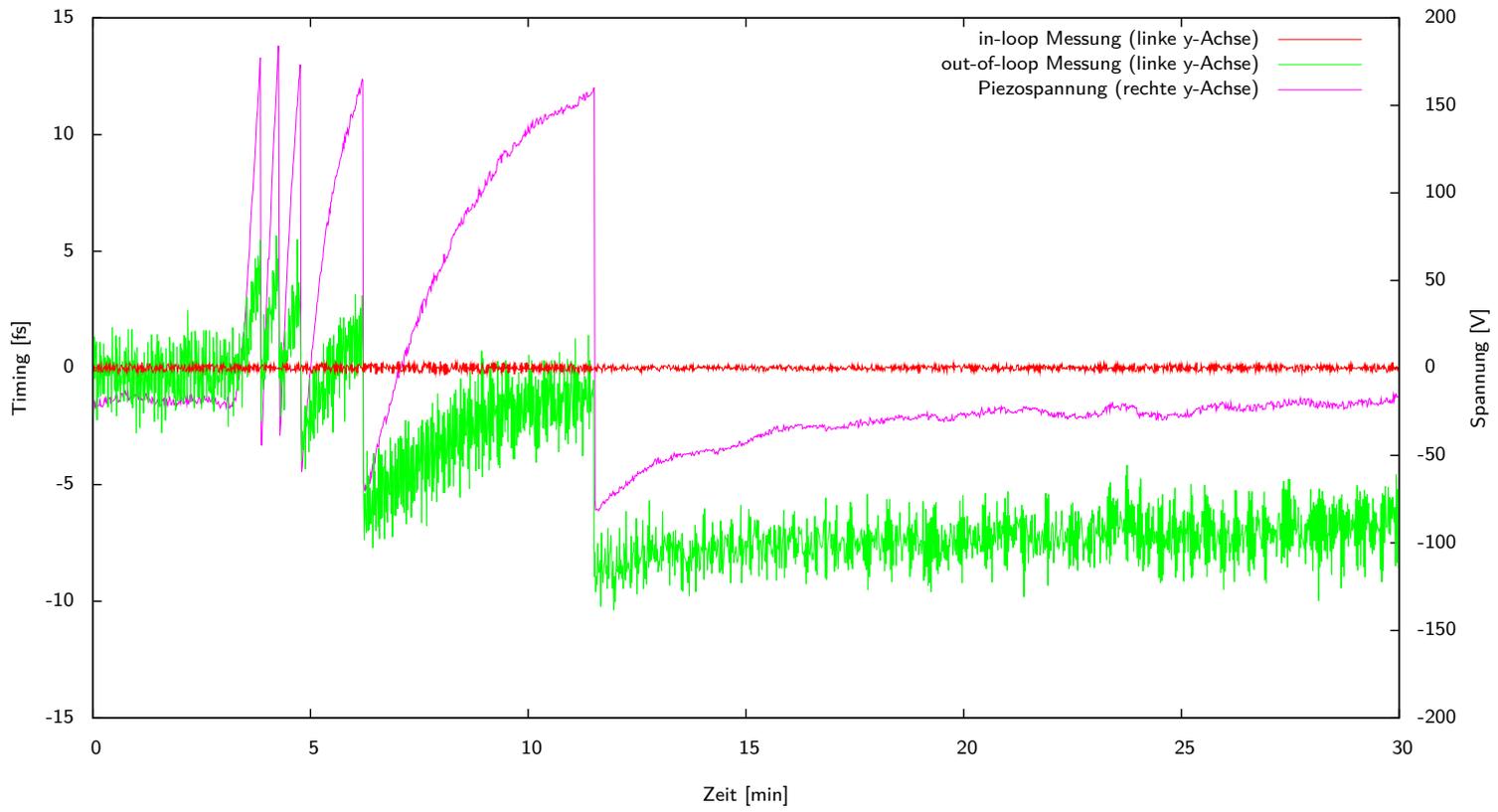


Abbildung 7.19.: Temperatursprung mit aktiver Regelung

7. Entwurf einer digitalen Regelung

Abbildung 7.20 zeigt eine ältere Messung über 24 Stunden mit aktiver Regelung. Als Link ist die 20 m lange Testfaser im Einsatz. Die optische Leistung beträgt auch hier wieder 2,4 mW. Abgebildet sind wie in den vorangegangenen Kapiteln die Ergebnisse der in-loop Messung und der out-of-loop Messung. Die Kalibrationskonstanten betragen $K_{\varphi,in} = 1,5 \text{ mV/fs}$ und $K_{\varphi,out} = 4,9 \text{ mV/fs}$. Die peak-to-peak Abweichung zwischen den Detektoren beträgt 34,3 fs, die Standardabweichung 4,9 fs. Bei dieser Messung war noch kein zweites Tiefpassfilter eingebaut. Dadurch wird bei der in-loop Messung deutlich mehr Detektorrauschen detektiert und die Reglerparameter müssen deutlich kleiner gewählt werden, um die Regelung stabil zu machen.

Um die Fehlertoleranz der Regelung zu untersuchen wurde die Linkfaser während der Messungen mit einigen Temperatursprüngen im Bereich um 0,2 K beaufschlagt. Die Sprünge sind so klein dimensioniert, um sie komplett mit dem Piezostretcher und ohne Einsatz der Delayline ausregeln zu können. Der längste und höchste Sprung mit 0,2 K über 30 min ist bei ungefähr 5 Stunden auch in den Messwerten sichtbar. Der Piezostretcher war in dieser Zeit auf fast 200 V ausgelenkt. Die Abweichung resultiert aus dem zuvor beschriebenen Messfehler, ausgelöst durch die Auslenkung des Piezostretchers. Der Regler hält zwar den Messwert innerhalb der Regelschleife konstant, das Timing am Linkende wird jedoch trotzdem beeinflusst, da der Regler auch den Messfehler ausgleicht.

Die Mittelwerte der Detektorsignale stimmen auch nach 24 Stunden noch sehr gut überein, es sind keine Driften erkennbar. Die korrekte Funktion der Regelung konnte über 24 Stunden trotz des schlecht eingestellten Reglers gezeigt werden. Wegen Problemen mit dem Laser konnte keine weitere Messung über einen Zeitraum dieser Länge mit den endgültigen Parametern aufgenommen werden. Es gibt allerdings keine Hinweise darauf, dass der entworfene und im Sprungversuch verwendete Regler nicht auch über lange Zeiträume funktioniert. Die Messdaten zeigen jedenfalls keine Probleme mit der Regelung, die offenen Fragen betreffen nur die Einflüsse der Aktuatoren auf die Messungen.

Es wurde sowohl die Stabilität als auch eine hinreichende Genauigkeit der Regelung unter Beweis gestellt.

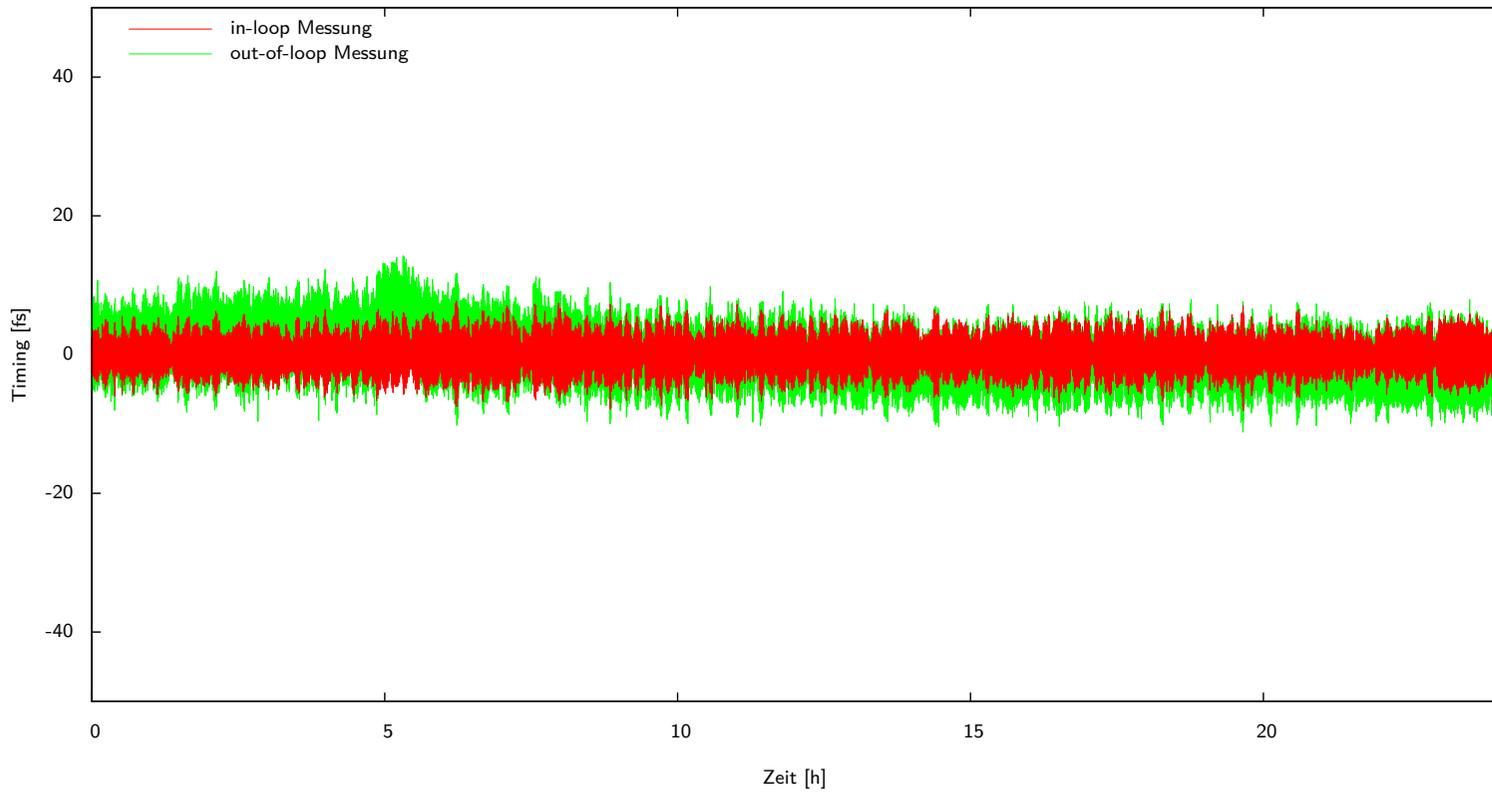


Abbildung 7.20.: Langzeitmessung mit aktiver Regelung

8. Zusammenfassung und Ausblick

In den ersten Kapiteln wurde zunächst ein kurzer Einblick in aktuelle Entwicklungen und Designs von hochpräzisen Synchronisationssystemen für Teilchenbeschleuniger gegeben. Die gebräuchlichsten Techniken zur Diagnose und Stabilisierung solcher Synchronisationssysteme wurden kurz vorgestellt und die Eigenschaften überlagerter Pulszüge aus Femtosekunden-Lasern untersucht.

Mit Hilfe einer Simulation konnte für den optischen Aufbau der Linkstabilisierung eine optimierte Version einer optischen Delayline entworfen und gebaut werden. Neben einem gegenüber aktuellen Delaylines verdoppelten Stellweg zeichnet sich diese Delayline durch die prinzipbedingte Vermeidung von Einkopplungsverlusten aus. Das Design auf Basis eines Retroreflektors zeigt keine winkelabhängigen Verluste mehr und Probleme durch Strahldivergenz können in einem bereits vorgeschlagenen weiteren Schritt ebenfalls behoben werden.

Parallel dazu wurde ein Detektor für die an Faserlinks hauptsächlich durch Temperaturschwankungen auftretenden Timingdriften entworfen. Der Detektor setzt ein neuartiges Konzept um, bei dem mit einer Photodiode die Überlagerung des driftenden Linkpulszuges mit einem Referenzpulszug detektiert und anschließend aus dem entstandenen Frequenzkamm eine Harmonische herausgefiltert wird. Nach Mischung an einem Hochfrequenzmischer mit einem weiteren Referenzsignal kann die in der Harmonischen kodierte Timinginformation direkt gemessen werden. Dieses Detektionsschema ist robust gegenüber fast allen bekannten Fehlerquellen und insbesondere gegenüber Temperaturdriften der Photodiode stabil. In Langzeitmessungen über bis zu 33 h an einem eigens aufgebauten Faserlink konnte durch eine out-of-loop Messung eine Langzeitstabilität von 4,8 fs (peak-to-peak) bei einer Standardabweichung von 0,86 fs gezeigt werden. Dadurch eröffnen sich völlig neue Anwendungsfelder für den hier entwickelten Detektor.

Diese Genauigkeit wurde weltweit bisher mit keinem photodiodenbasierten Detektionsschema erreicht.

Darauf aufbauend konnte ein Digitalregler entworfen und ein Faserlink aktiv stabilisiert werden. Die geforderte Genauigkeit von wenigen zehn Femtosekunden wurde erreicht und übertroffen.

Das entwickelte Detektionsschema wird nun für weitere Untersuchungen erste Schritte zur Industrialisierung durchlaufen. Eine miniaturisierte Version der Hochfrequenz-

8. Zusammenfassung und Ausblick

Elektronik auf einer kompakten Leiterplatte befindet sich derzeit in der Entwicklung. Eine Serienfertigung für den kommenden XFEL ist beabsichtigt.

Anhang

Anhang A.

Informationen zum digitalen Anhang

Alle beigelegten Modelle und Simulationen wurden mit Matlab R2009b entwickelt. Zum Teil kamen Toolboxen, wie zum Beispiel Simulink oder die System Control Toolbox zum Einsatz, die auch zur Ausführung der Modelle und Scripte benötigt werden. Insbesondere die Simulation des Retroreflektors benötigt unbedingt eine Matlabversion aus 2009 oder neuer, da in älteren Versionen von Matlab noch keine objektorientierte Programmierung unterstützt wird.

Auf der beiliegenden CD befinden sich 4 Verzeichnisse:

diplomarbeit Dieses Verzeichnis beinhaltet die vorliegende Diplomarbeit in elektronischer Form.

simulation_regler Hier sind die verwendeten Simulink-Modelle und das geschätzte Streckenmodell zu finden. Außerdem liegt ein Matlab Script bei, mit dem die Plots aus dem Kapitel der Regelungstechnik erzeugt werden können.

simulation_detektor Die GUI zur Detektorsimulation sowie das zugehörige Script befinden sich in diesem Verzeichnis.

simulation_retroreflektor Die Klassen zur Simulation von Optikkomponenten und das kurze Script zur Simulation der letzten Endes verwendeten Konfiguration sind hier zu finden.

Anhang B.

Quellcode zur Simulation der Delayline

B.1. Die Klasse *beam*

```
1 % -----
2 % matlab beam class
3 % geometric analysis of retroreflectors
4 % -----
5 % Thorsten Lamb
6 % DESY – MSK
7 % -----
8
9 classdef beam < handle
10     properties (SetAccess=private)
11         t = []; % base vector
12         v = []; % direction vector v1
13     end
14
15     methods
16         function this = beam(t, v)
17             % constructor takes one argument of type beam, two vectors or
18             % uses empty vectors in case of missing or unexpected input
19             try
20                 if (nargin == 2)
21                     this.set_t(t);
22                     this.set_v(v);
23                 elseif (nargin == 1 && isa(t, 'beam') )
24                     props=properties(t);
25                     for i=1:length(props)
26                         this.(props{i}) = t.(props{i});
27                     end
28                 else
29                     throw(MException('Beam:UsingDefaults', ...
30                                     'Beam: wrong parameters ... empty object created'));
31                 end
32             catch exception
33                 this.t = [];
34                 this.v = [];
35                 if (strcmp(exception.identifier, 'Beam:UsingDefaults') == false)
36                     exception = addCause(MException('Beam:WrongParameter', ...
```

```

37         'Beam: wrong parameters ... no object created'), exception ←
38         );
39         throw(exception);
40     else
41         warning('Beam:UsingDefaults', ...
42         'Beam: wrong parameters ... empty object created');
43     end
44 end
45
46 function display(this)
47     % function to display property values
48     disp(' ');
49     disp(['beam "', inputname(1), '" = '])
50     disp(['t': ', sprintf('\t%.3E', this.t)])
51     disp(['v': ', sprintf('\t%.3E', this.v)])
52     disp(' ');
53 end
54
55 function ret = is_valid(this)
56     % returns true, if all elements are properly set
57     ret = (isequal(size(this.t), [3 1]) && isnumeric(this.t) ...
58     && isequal(size(this.v), [3 1]) && isnumeric(this.v) ...
59     && (norm(this.v) ~= 0));
60 end
61
62 function ret = set_t(this, t)
63     % individually set beam.t
64     try
65         if (isequal(size(t), [3 1]) && isnumeric(t))
66             this.t = t;
67         else
68             throw(MException('Beam:UsingDefaults', ...
69             'Beam: wrong dimension for t, needs to be 3x1 ... t ←
70             cleared'));
71         end
72     catch exception
73         this.t = [];
74         if (strcmp(exception.identifier, 'Beam:UsingDefaults') == false)
75             exception = addCause(MException('Beam:UsingDefaults', ...
76             'Beam: incorrect base vector ... t cleared'), exception);
77         end
78         throw(exception);
79     end
80     ret = this;
81 end
82
83 function ret = set_base(this, base)
84     % set base vector of beam absolute
85     try
86         this.set_t(base);
87     catch exception

```

```

87         this.t = [];
88         exception = addCause(MException('Beam:UsingDefaults', ...
89             'Beam: incorrect base vector ... t cleared'), exception);
90         throw(exception);
91     end
92     ret = this;
93 end
94
95 function ret = set_base_rel(this, base_rel)
96     % set base vector of beam relative to old base vector
97     try
98         if (isequal(size(base_rel), size(this.t)) && isnumeric(base_rel))
99             this.set_t(this.t + base_rel);
100        else
101            warning('Beam:UsingDefaults', ...
102                'Beam: dimension mismatch ... nothing done');
103        end
104        catch exception
105            warning('Beam:UsingDefaults', ...
106                'Beam: incorrect base vector ... nothing done');
107            if (strcmp(exception.identifier, 'Beam:UsingDefaults') == true)
108                throw(exception);
109            end
110        end
111        ret = this;
112    end
113
114    function ret = set_v(this, v)
115        % individually set beam.v
116        try
117            if (isequal(size(v), [3 1]) && isnumeric(v) && (norm(v) ~= 0))
118                this.v = v;
119            else
120                throw(MException('Beam:UsingDefaults', ...
121                    'Beam: wrong dimension for dir, needs to be 3x1 ... v ←
122                    cleared'));
123            end
124            catch exception
125                this.v = [];
126                if (strcmp(exception.identifier, 'Beam:UsingDefaults') == false)
127                    exception = addCause(MException('Beam:UsingDefaults', ...
128                        'Beam: incorrect direction vector ... v ←
129                        exception);
130                end
131                throw(exception);
132            end
133            ret = this;
134        end
135
136    function ret = set_dir_kart(this, dir)
137        % set direction vector of beam absolute
138        try

```

Anhang B. Quellcode zur Simulation der Delayline

```
137         this.set_v(dir);
138     catch exception
139         this.v = [];
140         exception = addCause(MException('Beam:UsingDefaults', ...
141             'Beam: incorrect direction vector ... v cleared'), exception);
142         throw(exception);
143     end
144     ret = this;
145 end
146
147 function ret = set_dir_kart_rel(this, dir_rel)
148     % set direction vector of beam relative to old direction vector
149     try
150         if (isequal(size(dir_rel), size(this.v)) && isnumeric(dir_rel))
151             this.set_v(this.v+dir_rel);
152         else
153             warning('Beam:NoChange', ...
154                 'Beam: dimension mismatch ... nothing done');
155         end
156     catch exception
157         warning('Beam:UsingDefaults', ...
158             'Beam: incorrect direction vector ... nothing done');
159         if (strcmp(exception.identifier, 'Beam:UsingDefaults') == true)
160             throw(exception);
161         end
162     end
163     ret = this;
164 end
165
166 function ret = set_dir_spher(this, phi, theta)
167     % set direction vector absolute from beam base in spherical coordinates
168     % phi measured from x(positive) (0-360)
169     % theta measured from z(positive) (0-180) (in degree)
170     try
171         if (isscalar(phi) && isscalar(theta) ...
172             && isnumeric(phi) && isnumeric(theta))
173             % transform spherical coordinates into kartesian coordinates
174             if (phi/360 >= 1)
175                 warning('Beam:CutOffAngle', ...
176                     'Beam: phi >360 degrees ... cut off');
177                 phi = rem(phi, 360);
178             end
179             if (theta/180 >= 1)
180                 warning('Beam:CutOffAngle', ...
181                     'Beam: theta > 180 degrees ... cut off');
182                 theta = rem(theta, 180);
183             end
184             this.set_v([sind(theta)*cosd(phi);sind(theta)*sind(phi);cosd(
185                 theta)]);
186         else
187             throw(MException('Beam:UsingDefaults', ...
188                 'Beam: angles need to be scalar ... v cleared'));
189         end
190     end
191 end
```

```

188         end
189     catch exception
190         this.v = [];
191         if (strcmp(exception.identifier, 'Beam:UsingDefaults') == false)
192             exception = addCause(MException('Beam:UsingDefaults', ...
193                 'Beam: incorrect angles ... v cleared'), exception);
194         end
195         throw(exception);
196     end
197     ret = this;
198 end
199
200 function ret = set_dir_spher_rel(this, phi_rel, theta_rel)
201     % set direction vector of beam relative to old direction vector in
202     % sperical coordinates (in degree)
203     try
204         if (isscalar(phi_rel) && isscalar(theta_rel) ...
205             && isnumeric(phi_rel) && isnumeric(theta_rel))
206             if (norm(this.v) ~= 0) % direction vector already set -> move    ←
207                 % first convert old v to spherical coordinates, then add
208                 % phi_rel ans theta_rel
209                 phi_rel = rem(atan2(this.v(2),this.v(1))*180/pi+360, 360)    ←
210                     +phi_rel;
211                 theta_rel = acos(this.v(3)/norm(this.v))*180/pi +    ←
212                     theta_rel;
213             end % finaly convert back to kart. coord.
214             this.set_dir_spher(phi_rel, theta_rel);
215         else
216             warning('Beam:NoChange', ...
217                 'Beam: angles need to be scalar ... nothing done');
218         end
219     catch exception
220         warning('Beam:UsingDefaults', ...
221             'Beam: incorrect angles ... nothing done');
222     if (strcmp(exception.identifier, 'Beam:UsingDefaults') == true)
223         throw(exception);
224     end
225     ret = this;
226 end
227 end

```

B.2. Die Klasse *mirror*

```

1 % -----
2 % matlab mirror class
3 % geometric analysis of retroreflectors
4 % -----
5 % Thorsten Lamb

```

Anhang B. Quellcode zur Simulation der Delayline

```
6 % DESY - MSK
7 % -----
8
9 classdef mirror < handle
10     properties (SetAccess=private)
11         t = [];           % base vector
12         v1 = [];         % direction vector v1
13         v2 = [];         % direction vector v2
14         beam_in = [];    % incoming beam
15
16         normal = [];     % normal vector on mirror
17         intersection = []; % intersection of incoming beam with mirror plane
18         is_positive = false; % intersection is a positive recombination of v1 and v2
19         beam_out = [];   % reflected beam
20     end
21
22     methods (Access=private)
23         function ret = update_normal(this)
24             % calculate and return normal vector
25             normal_vec = cross(this.v1, this.v2);
26             % cross product is always orthogonal to both factors
27             if (norm(normal_vec) ~= 0) % vectors are independent -> return normalized    ←
28                 result
29                 this.normal = normal_vec / norm(normal_vec);
30             else
31                 this.normal = [];
32                 throw(MException('Mirror:NoNormal', ...
33                     'Mirror: direction vectors are not independent ... no normal    ←
34                     found'));
35             end
36             ret = this.normal;
37         end
38
39         function ret = update_intersection(this)
40             % calculate and return base vector of intersection with given beam
41             if isa(this.beam_in, 'beam')
42                 % left and right side matrix of linear equation system
43                 A = [this.v1, this.v2, -this.beam_in.v];
44                 b = this.beam_in.t - this.t;
45                 % use mldivide to solve linear system of equations
46                 % matlab performs intelligent selection of the appropriate
47                 % algorithm and warns if problems occur
48                 lastwarn('','');
49                 x = A \ b;
50                 [~, msgid] = lastwarn;
51                 if (strcmp(msgid, 'MATLAB:singularMatrix') == false)
52                     % calculate return vector by entering result into incoming
53                     % beam equation
54                     this.intersection = this.beam_in.t + x(3)*this.beam_in.v;
55                     this.is_positive = (x(1)>=0 && x(2)>=0) ...
56                         && ~(x(1)==0 && x(2)==0) ...
57                         && (dot(this.normal, this.beam_in.v)<0);
```

```

56         else
57             this.is_positive = false;
58             this.intersection = [];
59             warning('Mirror:NoIntersection', ...
60                 'Mirror: no intersection found ... beam_in is parallel to
                    mirror');
61         end
62     else
63         this.is_positive = false;
64         this.intersection = [];
65         throw(MException('Mirror:NoBeamIn', ...
66             'Mirror: beam_in not set ... no intersection calculated'));
67     end
68     ret = this.intersection;
69 end
70
71 function ret = reset_output(this)
72     % reset all output variables
73     this.normal = [];
74     this.intersection = [];
75     this.is_positive = false;
76     this.beam_out = [];
77     ret = this;
78 end
79 end
80
81 methods
82     function this = mirror(t, v1, v2)
83         % constructor takes one argument of type mirror, three vectors or
84         % uses empty vectors in case of missing or unexpected input
85         try
86             if (nargin == 3)
87                 this.set_t(t);
88                 this.set_v1(v1);
89                 this.set_v2(v2);
90             elseif (nargin == 1 && isa(t, 'mirror'))
91                 props=properties(t);
92                 for i=1:length(props)
93                     this.(props{i}) = t.(props{i});
94                 end
95             else
96                 throw(MException('Mirror:UsingDefaults', ...
97                     'Mirror: wrong parameters ... empty object created'));
98             end
99         catch exception
100             this.t = [];
101             this.v1 = [];
102             this.v2 = [];
103             if (strcmp(exception.identifier, 'Mirror:UsingDefaults') == false
104                 )
105                 exception = addCause(MException('Mirror:WrongParameter', ...
106                     'Mirror: wrong parameters ... no object created'),

```

```

106         exception);
107         throw(exception);
108     else
109         warning('Mirror:UsingDefaults', ...
110             'Mirror: wrong parameters ... empty object created');
111     end
112 end
113
114 function display(this)
115     % function to display property values
116     disp(' ');
117     disp(['mirror "', inputname(1), '" = ']);
118     disp(['t'      : ', sprintf('\t%.3E', this.t)])
119     disp(['v1'     : ', sprintf('\t%.3E', this.v1)])
120     disp(['v2'     : ', sprintf('\t%.3E', this.v2)])
121     disp(['normal' : ', sprintf('\t%.3E', this.normal)])
122     disp(['intersection': ', sprintf('\t%.3E', this.intersection)])
123     if this.is_positive == true
124         disp('is_positive : true')
125     else
126         disp('is_positive : false')
127     end
128     disp(' ');
129 end
130
131 function ret = is_valid(this)
132     % returns true, if all elements except beam_in are properly set
133     ret = (isequal(size(this.t), [3 1]) && isnumeric(this.t) ...
134         && isequal(size(this.v1), [3 1]) && isnumeric(this.v1) ...
135         && (norm(this.v1) ~= 0) && isequal(size(this.v2), [3 1]) ...
136         && isnumeric(this.v2) && (norm(this.v2) ~= 0));
137 end
138
139 function ret = set_t(this, t)
140     % individually set vector t
141     try
142         if (isequal(size(t), [3 1]) && isnumeric(t))
143             this.t = t;
144             this.reset_output;
145         else
146             throw(MException('Mirror:UsingDefaults', ...
147                 'Mirror: wrong dimension for t, needs to be 3x1 ... t
148                 cleared'));
149         end
150     catch exception
151         this.t = [];
152         this.reset_output;
153         if (strcmp(exception.identifier, 'Mirror:UsingDefaults') == false
154             )
155             exception = addCause(MException('Mirror:UsingDefaults', ...
156                 'Mirror: incorrect base vector ... t cleared'), exception)
157         end
158     end

```

```

155         end
156         throw(exception);
157     end
158     ret = this;
159 end
160
161 function ret = set_base(this, base)
162     % set base vector of mirror absolute
163     try
164         this.set_t(base);
165     catch exception
166         exception = addCause(MException('Mirror:UsingDefaults', ...
167             'Mirror: incorrect base vector ... t cleared'), exception);
168         throw(exception);
169     end
170     ret = this;
171 end
172
173 function ret = set_base_rel(this, base_rel)
174     % set base vector of mirror relative to old base vector
175     try
176         if (isequal(size(base_rel), size(this.t)) && isnumeric(base_rel))
177             this.set_t(this.t + base_rel);
178         else
179             warning('Mirror:UsingDefaults', ...
180                 'Mirror: dimension mismatch ... nothing done');
181         end
182     catch exception
183         warning('Mirror:UsingDefaults', ...
184             'Mirror: incorrect base vector ... nothing done');
185         if (strcmp(exception.identifier, 'Mirror:UsingDefaults') == true)
186             throw(exception);
187         end
188     end
189     ret = this;
190 end
191
192 function ret = set_v1(this, v1)
193     % individually set vector v1
194     try
195         if (isequal(size(v1), [3 1]) && isnumeric(v1) && (norm(v1) ~= 0))
196             this.v1 = v1;
197             this.reset_output;
198         else
199             throw(MException('Mirror:UsingDefaults', ...
200                 'Mirror: wrong dimension for dir, needs to be 3x1 ... v1 ←
201                 cleared'));
202         end
203     catch exception
204         this.v1 = [];
205         this.reset_output;

```

Anhang B. Quellcode zur Simulation der Delayline

```
205         if (strcmp(exception.identifier, 'Mirror:UsingDefaults') == false ←
206             )
207             exception = addCause(MException('Mirror:UsingDefaults', ... ←
208                 'Mirror: incorrect direction vector ... v1 cleared'), ←
209                 exception);
210         end
211         throw(exception);
212     end
213     ret = this;
214 end
215
216 function ret = set_v2(this, v2)
217     % individually set vector v2
218     try
219         if (isequal(size(v2), [3 1]) && isnumeric(v2) && (norm(v2) ~= 0))
220             this.v2 = v2;
221             this.reset_output;
222         else
223             throw(MException('Mirror:UsingDefaults', ... ←
224                 'Mirror: wrong dimension for dir, needs to be 3x1 ... v2 ←
225                 cleared'));
226         end
227     catch exception
228         this.v2 = [];
229         this.reset_output;
230         if (strcmp(exception.identifier, 'Mirror:UsingDefaults') == false ←
231             )
232             exception = addCause(MException('Mirror:UsingDefaults', ... ←
233                 'Mirror: incorrect direction vector ... v2 cleared'), ←
234                 exception);
235         end
236         throw(exception);
237     end
238     ret = this;
239 end
240
241 function ret = set_beam_in(this, beam_in)
242     % set incoming beam
243     try
244         if (isa(beam_in, 'beam') && ~isempty(beam_in.v) && ~isempty( ←
245             beam_in.t)) ←
246             this.beam_in = beam_in;
247             this.reset_output;
248         else
249             throw(MException('Mirror:UsingDefaults', ... ←
250                 'Mirror: beam empty or non existent ... beam_in cleared')) ←
251             ;
252         end
253     catch exception
254         this.beam_in = [];
255         this.reset_output;
256         if (strcmp(exception.identifier, 'Mirror:UsingDefaults') == false ←
```

```

250         )
251         exception = addCause(MException('Mirror:UsingDefaults', ...
252             'Mirror: beam empty or non existent ... beam_in cleared'), ←
253             exception);
254     end
255     throw(exception);
256 end
257
258 function ret = update_all(this)
259     % calculate norm, intersection and beam_out
260     try
261         if (this.is_valid() == true)
262             this.reset_output;
263             this.update_normal;
264             this.update_intersection;
265             [~, msgid] = lastwarn;
266             if (strcmp(msgid, 'Mirror:NoIntersection') == true)
267                 this.beam_out = this.beam_in;
268                 ret = this;
269                 return;
270             end
271         else
272             throw(MException('Mirror:NoCalculation', ...
273                 'Mirror: not all properties are set'));
274         end
275     catch exception
276         this.reset_output;
277         exception = addCause(MException('Mirror:NoBeamOut', ...
278             'Mirror: previous error prevents calculation of beam_out'), ←
279             exception);
280         throw(exception);
281     end
282     % calculate beam reflected by plane
283     % normalise incoming beam and change sign because of different directions
284     beam_in_norm = this.beam_in.v/norm(this.beam_in.v)*-1;
285     % calculate projection of incoming beam on mirror normal
286     projection = dot(this.normal, beam_in_norm)* this.normal;
287     % compute opposite leg of orthogonal triangle
288     distance = projection - beam_in_norm;
289     % add opposite leg twice to incoming beam to gain
290     % reflected beam (intrinsically normalized)
291     this.beam_out = beam(this.intersection, beam_in_norm + 2*distance);
292     ret = this;
293 end
294 end

```

B.3. Die Klasse *retroreflector*

Anhang B. Quellcode zur Simulation der Delayline

```

1 % -----
2 % matlab retroreflector class
3 % geometric analysis of retroreflectors
4 % -----
5 % Thorsten Lamb
6 % DESY – MSK
7 % 2009–June–04
8 % -----
9
10 classdef retroreflector < handle
11     properties (SetAccess=private)
12         t = [];           % base vector
13         r = 0;           % radius (size) of reflector
14         psi = 0;         % rotation around optical axis (0–360) (in degree)
15         theta = 0;       % spherical cordinates, theta measured from z (positive)   ←
16                             (0–180) (in degree)
17         phi = 0;         % spherical cordinates, phi measured from x (positive)   ←
18                             (0–360),
19         beam_in = [];    % incoming beam
20
21         intersections = []; % intersections with mirror planes
22         beam_out = [];   % outgoing beam
23         normal_vectors = false; % plot normal vectors (bool)
24     end
25
26     properties (SetAccess=private, GetAccess=private)
27         dir_rot = [];    % direction vectors (rotated and scaled)
28         faces = [];     % vector of 3 mirrors
29         normals = [];   % normals at intersection
30         res = 256;      % resolution of reflector plot
31         reflector_h = []; % handle for plotting
32         normals_h = []; % handle for plotting
33     end
34
35     methods (Access=private)
36         function update_mirrors_v(this)
37             % update mirrors by scaled direction vectors → rotate internal
38             % direction vectors by rotation angles and set correct mirror size
39             % uses: psi, theta, phi, r, faces()
40             % inputs should always be valid, error handling is otherwise
41             % performed by mirror class (e.g. r=0)
42             E = eye(3);
43             % set angles to tilt reflectors axis on positive x axis
44             beta=atand(1/sqrt(2));
45             gamma=-45;
46             % define rotation matrices
47             Ry = [cosd(beta) 0 sind(beta);0 1 0;-sind(beta) 0 cosd(beta)];
48             Rz = [cosd(gamma) -sind(gamma) 0;sind(gamma) cosd(gamma) 0;0 0 1];
49             R_psi = [1 0 0;0 cosd(this.psi) -sind(this.psi); 0 sind(this.psi)   ←
50                     cosd(this.psi)];
51             R_theta = [cosd(this.theta) 0 sind(this.theta);0 1 0;-sind(this.theta) ←
52                       ) 0 cosd(this.theta)];

```

```

49     R_phi = [cosd(this.phi) -sind(this.phi) 0; sind(this.phi) cosd(this.  ←
              phi) 0; 0 0 1];
50     % rotate direction vectors
51     this.dir_rot = R_phi*R_theta*R_psi*Ry*Rz*E*this.r*sqrt(2);
52     % update mirrors
53     this.faces(1).set_v1(this.dir_rot(:,1));
54     this.faces(1).set_v2(this.dir_rot(:,2));
55     this.faces(2).set_v1(this.dir_rot(:,2));
56     this.faces(2).set_v2(this.dir_rot(:,3));
57     this.faces(3).set_v1(this.dir_rot(:,3));
58     this.faces(3).set_v2(this.dir_rot(:,1));
59     end
60
61     function update_mirrors_t(this)
62     % update mirrors by base vector t
63     % error handling is performed by mirror class
64         this.faces(1).set_t(this.t);
65         this.faces(2).set_t(this.t);
66         this.faces(3).set_t(this.t);
67     end
68
69     function ret = update_beam_out(this)
70     % Calculate all reflections , set outgoing beam and
71     % intersections. Warn if reflector is missed.
72     % uses: beam_in, faces()
73     try
74         if (isa(this.beam_in, 'beam') ...
75             && this.beam_in.is_valid() && this.faces(1).is_valid ...
76             && this.faces(2).is_valid && this.faces(3).is_valid)
77             count = 1;
78             this.intersections = [];
79             this.normals = [];
80             this.faces(1).set_beam_in(this.beam_in);
81             this.faces(1).update_all;
82             while (~ (this.faces(1).is_positive ...
83                 && this.radial_dist(this.faces(1).intersection)<=this.  ←
84                 r) && count<=3)
85                 if (count == 3);
86                     warning('Retroreflector:MissingReflection', ...
87                         'Retroreflector: no intersection with  ←
88                         retroreflector');
89                     this.beam_out = this.beam_in;
90                     ret = this.beam_out;
91                     return;
92                 end
93                 this.faces = circshift(this.faces, [0 1]);
94                 this.faces(1).set_beam_in(this.beam_in);
95                 this.faces(1).update_all;
96                 count = count + 1;
97             end
98             this.intersections = [this.intersections, this.faces(1).  ←
99                 intersection];

```

Anhang B. Quellcode zur Simulation der Delayline

```
97         this.normals = [this.normals, this.faces(1).normal];
98         this.faces(2).set_beam_in(this.faces(1).beam_out);
99         this.faces(2).update_all;
100        if ~(this.faces(2).is_positive ...
101            && this.radial_dist(this.faces(2).intersection)<=this.    ←
102                r))
103            this.faces = [this.faces(1) this.faces(3) this.faces(2)];
104        end
105        this.faces(2).set_beam_in(this.faces(1).beam_out);
106        this.faces(2).update_all;
107        if ~(this.faces(2).is_positive ...
108            && this.radial_dist(this.faces(2).intersection)<=this.    ←
109                r))
110            warning('Retroreflector:MissingReflection', ...
111                'Retroreflector: only one intersection with    ←
112                    retroreflector');
113            this.beam_out = this.faces(1).beam_out;
114            ret = this.beam_out;
115            return;
116        end
117        this.intersections = [this.intersections, this.faces(2).    ←
118            intersection];
119        this.normals = [this.normals, this.faces(2).normal];
120        this.faces(3).set_beam_in(this.faces(2).beam_out);
121        this.faces(3).update_all;
122        if ~(this.faces(3).is_positive ...
123            && this.radial_dist(this.faces(3).intersection)<=this.    ←
124                r))
125            warning('Retroreflector:MissingReflection', ...
126                'Retroreflector: only two intersections with    ←
127                    retroreflector');
128            this.beam_out = this.faces(2).beam_out;
129            ret = this.beam_out;
130            return;
131        end
132        this.intersections = [this.intersections, this.faces(3).    ←
133            intersection];
134        this.normals = [this.normals, this.faces(3).normal];
135        this.beam_out = this.faces(3).beam_out;
136    else
137        throw(MException('Retroreflector:InvalidInput', ...
138            'Retroreflector: beam_in or reflector not properly set'));
139    end
140    catch exception
141        this.reset_output;
142        if (strcmp(exception.identifier, 'Retroreflector:InvalidInput')    ←
143            == false)
144            exception = addCause(MException('Retroreflector:    ←
145                NoIntersectionCalculated', ...
146                'Retroreflector: no intersections calculated'), exception)    ←
147                ;
148        end
149    end
```

```

139         throw(exception);
140     end
141     ret = this.beam_out;
142 end
143
144 function ret = radial_dist(this, point)
145     % calculate distance from given point to the optical axis of the
146     % reflector
147     % uses: t, point, dir_rot
148     try
149         if (isequal(size(this.t), [3 1]) && isnumeric(this.t) ...
150             && isequal(size(this.dir_rot), [3 3]) && (norm(this.  ←
151                 dir_rot) ~=0) ...
152             && isequal(size(point), [3 1]) && isnumeric(point))
153             axis_v = sum(this.dir_rot, 2);
154             % construct plane orthogonal to optical axis (axis_v)
155             % shift to ensure linear equation system is not singular
156             shift = 0;
157             while ((abs(axis_v(rem(shift, 3)+1)) > abs(axis_v(rem(shift+2,  ←
158                 3)+1))) ...
159                 || (abs(axis_v(rem(shift+1, 3)+1)) > abs(axis_v(rem(  ←
160                     shift+2, 3)+1))))
161                 shift = shift+1;
162             end
163             % trick to create independent orthogonal vectors: set one
164             % element zero, interchange the other elements and invert one
165             % finally shift back
166             v1 = circshift([0; axis_v(rem(shift+2, 3)+1); -axis_v(rem(  ←
167                 shift+1, 3)+1)], shift);
168             v2 = circshift([-axis_v(rem(shift+2, 3)+1); 0; axis_v(rem(  ←
169                 shift, 3)+1)], shift);
170             % use mirror class to calculate intersection
171             help_m = mirror(point, v1, v2);
172             help_b = beam(this.t, axis_v);
173             help_m.set_beam_in(help_b);
174             help_m.update_all;
175             % return distance from point to intersection
176             ret = norm(point-help_m.intersection);
177         else
178             throw(MException('Retroreflector:InvalidInput', ...
179                 'Retroreflector: invalid input while calculating radial  ←
180                 distance'));
181         end
182     catch exception
183         if (strcmp(exception.identifier, 'Retroreflector:InvalidInput')  ←
184             == false)
185             exception = addCause(MException('Retroreflector:  ←
186                 NoDistanceCalculated', ...
187                 'Retroreflector: no radial distance calculated'),  ←
188                 exception);
189         end
190     end
191     throw(exception);

```

```

182         end
183     end
184
185     function ret = reset_output(this)
186         this.intersections = [];
187         this.normals = [];
188         this.beam_out = [];
189         this.dir_rot = [];
190         warning off Mirror:UsingDefaults
191         this.faces = [mirror() mirror() mirror()];
192         warning on Mirror:UsingDefaults
193         ret = this;
194     end
195 end
196
197 methods
198     function this = retroreflector(t, r, psi, theta, phi)
199         % constructor, takes one argument of type retroreflector ,
200         % one vector, radius and direction or uses empty vectors
201         % in case of missing or unexpected input
202         try
203             warning off Mirror:UsingDefaults
204             this.faces = [mirror() mirror() mirror()];
205             warning on Mirror:UsingDefaults
206             if (nargin == 5)
207                 this.rotate_absolute(psi, theta, phi);
208                 this.set_t(t);
209                 this.set_r(r);
210             elseif (nargin == 1 && isa(t, 'retroreflector'))
211                 props=properties(t);
212                 for i=1:length(props)
213                     this.(props{i}) = t.(props{i});
214                 end
215             else
216                 throw(MException('Retroreflector:UsingDefaults', ...
217                     'Retroreflector: wrong parameters ... empty object created' ←
218                     '));
219
220             end
221         catch exception
222             this.t = [];
223             this.r = 0;
224             this.psi = 0;
225             this.theta = 0;
226             this.phi = 0;
227             if (strcmp(exception.identifier, 'Retroreflector:UsingDefaults') ←
228                 == false)
229                 exception = addCause(MException('Retroreflector:WrongParameter ←
230                     ', ...
231                     'Retroreflector: wrong parameters ... no object created'), ←
232                     exception);
233             throw(exception);
234         else
235

```

```

230         warning('Retroreflector:UsingDefaults', ...
231             'Retroreflector: wrong parameters ... empty object created ←
                ');
232     end
233 end
234 end
235
236 function display(this)
237     % function to display property values
238     disp(' ');
239     disp(['retroreflector "', inputname(1),'" = '])
240     disp(['t'          : ', sprintf('\t%.3E', this.t)])
241     disp(['r'          : ', sprintf('\t%.3E', this.r)])
242     disp(['psi, theta, phi : ', sprintf('\t%.3E\t%.3E\t%.3E', this.psi, ←
                this.theta, this.phi)])
243     if this.normal_vectors == true
244         disp('normal_vectors : true')
245     else
246         disp('normal_vectors : false')
247     end
248     disp(' ');
249     disp(['intersections'' : ', sprintf('\t%.3E\t%.3E\t%.3E\n\t\t', this. ←
                intersections)])
250     disp(['normals''      : ', sprintf('\t%.3E\t%.3E\t%.3E\n\t\t', this. ←
                normals)])
251 end
252
253 function ret = set_t(this, t)
254     % individually set vector t
255     try
256         if (isequal(size(t), [3 1]) && isnumeric(t))
257             this.t = t;
258             this.reset_output;
259         else
260             throw(MException('Retroreflector:UsingDefaults', ...
261                 'Retroreflector: wrong dimension for t, needs to be 3x1 ←
                ... t cleared'));
262         end
263     catch exception
264         this.t = [];
265         this.reset_output;
266         if (strcmp(exception.identifier, 'Retroreflector:UsingDefaults') ←
                == false)
267             exception = addCause(MException('Retroreflector:UsingDefaults' ←
                , ...
268                 'Retroreflector: invalid base vector ... t cleared'), ←
                exception);
269         end
270         throw(exception);
271     end
272     ret = this;
273 end

```

Anhang B. Quellcode zur Simulation der Delayline

```
274
275 function ret = set_base(this, base)
276     % set base vector of retroreflector absolute
277     try
278         this.set_t(base);
279     catch exception
280         exception = addCause(MException('Retroreflector:UsingDefaults', ←
281             ...
282             'Retroreflector: invalid base vector ... t cleared'), ←
283             exception);
284         throw(exception);
285     end
286     ret = this;
287 end
288
289 function ret = set_base_rel(this, base_rel)
290     % set base vector of retroreflector relative to old base vector
291     try
292         if (isequal(size(base_rel), size(this.t)) && isnumeric(base_rel))
293             this.set_t(this.t + base_rel);
294         else
295             warning('Retroreflector:UsingDefaults', ...
296                 'Retroreflector: dimension mismatch ... nothing done');
297         end
298     catch exception
299         warning('Retroreflector:UsingDefaults', ...
300             'Retroreflector: invalid base vector ... nothing done');
301         if (strcmp(exception.identifier, 'Retroreflector:UsingDefaults') ←
302             == true)
303             throw(exception);
304         end
305     end
306     ret = this;
307 end
308
309 function ret = set_r(this, r)
310     % individually set radius
311     try
312         if (isscalar(r) && isnumeric(r))
313             this.r = r;
314             this.reset_output;
315         else
316             throw(MException('Retroreflector:UsingDefaults', ...
317                 'Retroreflector: wrong dimension for r, needs to be scalar ←
318                 ... r cleared'));
319         end
320     catch exception
321         this.r = 0;
322         this.reset_output;
323         if (strcmp(exception.identifier, 'Retroreflector:UsingDefaults') ←
324             == false)
325             exception = addCause(MException('Retroreflector:UsingDefaults' ←
```

```

321         , ...
          'Retroreflector: invalid radius ... r cleared'), exception ←
          );
322     end
323     throw(exception);
324 end
325 ret = this;
326 end
327
328 function ret = rotate_absolute(this, psi, theta, phi)
329     % individually rotate by angles psi, theta and phi
330     try
331         if (isscalar(psi) && isnumeric(psi) ...
332             && isscalar(theta) && isnumeric(theta) ...
333             && isscalar(phi) && isnumeric(phi))
334             if (psi/360 >= 1)
335                 warning('Retroreflector:CutOffAngle', ...
336                     'Retroreflector: psi >360 degrees ... cut off');
337                 psi = rem(psi, 360);
338             end
339             if (phi/360 >= 1)
340                 warning('Retroreflector:CutOffAngle', ...
341                     'Retroreflector: phi >360 degrees ... cut off');
342                 phi = rem(phi, 360);
343             end
344             if (theta/180 >= 1)
345                 warning('Retroreflector:CutOffAngle', ...
346                     'Retroreflector: theta > 180 degrees ... cut off');
347                 theta = rem(theta, 180);
348             end
349             this.psi = psi;
350             this.theta = theta;
351             this.phi = phi;
352             this.reset_output;
353         else
354             throw(MException('Retroreflector:UsingDefaults', ...
355                 'Retroreflector: angles have to be scalar'));
356         end
357     catch exception
358         this.psi = 0;
359         this.theta = 0;
360         this.phi = 0;
361         this.reset_output;
362         if (strcmp(exception.identifier, 'Retroreflector:UsingDefaults') ←
363             == false)
364             exception = addCause(MException('Retroreflector:UsingDefaults' ←
365                 , ...
366                 'Retroreflector: invalid angles ... psi, theta and phi ←
367                 cleared'), exception);
368         end
369     end
370     throw(exception);
371 end

```

Anhang B. Quellcode zur Simulation der Delayline

```
368         ret = this;
369     end
370
371     function ret = rotate_relative(this, psi_rel, theta_rel, phi_rel)
372         % rotate in addition to old rotation angles
373         try
374             if (isscalar(psi_rel) && isnumeric(psi_rel) ...
375                 && isscalar(theta_rel) && isnumeric(theta_rel) ...
376                 && isscalar(phi_rel) && isnumeric(phi_rel))
377                 this.rotate_absolute( ...
378                     this.psi + psi_rel, this.theta + theta_rel, this.phi + ←
379                         phi_rel);
380             else
381                 warning('Retroreflector:UsingDefaults', ...
382                     'Retroreflector: wrong format ... nothing done');
383             end
384         catch exception
385             warning('Retroreflector:UsingDefaults', ...
386                 'Retroreflector: incorrect angles ... nothing done');
387             if (strcmp(exception.identifier, 'Retroreflector:UsingDefaults') ←
388                 == true)
389                 throw(exception);
390             end
391         end
392         ret = this;
393     end
394
395     function ret = set_beam_in(this, beam_in)
396         % set incoming beam
397         try
398             if (isa(beam_in, 'beam') && ~isempty(beam_in.v) && ~isempty( ←
399                 beam_in.t))
400                 this.beam_in = beam_in;
401                 this.reset_output;
402             else
403                 throw(MException('Retroreflector:UsingDefaults', ...
404                     'Retroreflector: beam empty or non existent ... beam_in ←
405                         cleared'));
406             end
407         catch exception
408             this.beam_in = [];
409             this.reset_output;
410             if (strcmp(exception.identifier, 'Retroreflector:UsingDefaults') ←
411                 == false)
412                 exception = addCause(MException('Retroreflector:UsingDefaults' ←
413                     , ...
414                     'Retroreflector: beam empty or non existent ... beam_in ←
415                         cleared'), exception);
416             end
417         end
418         throw(exception);
419     end
420     ret = this.beam_in;
```

```

413     end
414
415     function ret = update_all(this)
416         % calculate all reflections , update all values
417         try
418             this.reset_output;
419             this.update_mirrors_v;
420             this.update_mirrors_t;
421             this.update_beam_out;
422         catch exception
423             exception = addCause(MException('Retroreflector:UpdateImpossible' ←
424                 , ...
425                 'Retroreflector: update impossible'), exception);
426             throw(exception);
427         end
428         ret = this;
429     end
430
431     function ret = draw(this)
432         % draw reflector including normals if desired
433         % uses: t, r, dir_rot, normal_vectors, intersections, normals,
434         % psi, theta, phi
435         try
436             this.update_mirrors_v;
437             this.update_mirrors_t;
438             hold on;
439             % remove old objects from plot
440             if (ishandle(this.reflector_h)); delete(this.reflector_h); end
441             if (ishandle(this.normals_h)); delete(this.normals_h); end
442             % use rotated face to get coordinates
443             Y = linspace(0, this.r*sqrt(2), this.res);
444             Z = linspace(0, this.r*sqrt(2), this.res);
445             [Y Z] = meshgrid(Y, Z);
446             X = zeros(this.res, this.res);
447             face = surf(X,Y,Z,'facecolor', 'r','edgecolor','none','facealpha' ←
448                 ,.5);
449             rotate(face, [0 0 1], -45, [0;0;0]);
450             rotate(face, [0 1 0], atand(1/sqrt(2)), [0;0;0]);
451             X=get(face, 'XData');
452             Y=get(face, 'YData');
453             Z=get(face, 'ZData');
454             delete(face);
455             % distance from axis
456             distance=sqrt(Z.^2+Y.^2);
457             distance(distance > this.r) = NaN;
458             % detect edge
459             [~,I] = max(fliplr(distance), [], 2); %von rechts
460             [~,J] = max(flipud(distance), [], 1); %von oben
461             % convert to indices and delete NaNs
462             A = [(1:size(I,1))', this.res+1-I];
463             B = [this.res+1-J', (1:size(J,2))'];
464             A(isnan(distance(sub2ind([this.res this.res], A(:,1), A(:,2))))) ←

```

```

        ,:)=[];
463     B(isnan(distance(sub2ind([this.res this.res], B(:,1), B(:,2)))) ←
        ,:)=[];
464     % stick together indices of both edges
465     A = sortrows(A, 1);
466     B = sortrows(B, -2);
467     coord = [A(1:find(abs(A(:,1)-A(:,2))<=1),:);
468             B(find(abs(B(:,1)-B(:,2))<=1):end,:)]];
469     idx = [1 sub2ind([this.res this.res], coord(:,1)', coord(:,2)')]];
470     % get actual coordinates
471     corners = [X(idx)'+this.t(1) Y(idx)'+this.t(2) Z(idx)'+this.t(3) ←
        ];
472     num_corners = size(corners, 1);
473     connect = 1:num_corners;
474     % draw faces
475     this.reflector_h(1) = patch('vertices', corners, 'faces', connect ←
        , ...
476         'facecolor', 'g','edgecolor','black','facealpha',.5);
477     this.reflector_h(2) = patch('vertices', corners, 'faces', connect ←
        , ...
478         'facecolor', 'g','edgecolor','black','facealpha',.5);
479     rotate(this.reflector_h(2),[1 0 0], 120, this.t);
480     this.reflector_h(3) = patch('vertices', corners, 'faces', connect ←
        , ...
481         'facecolor', 'g','edgecolor','black','facealpha',.5);
482     rotate(this.reflector_h(3),[1 0 0], 240, this.t);
483     % do final rotations
484     rotate(this.reflector_h,[1 0 0], this.psi, this.t);
485     rotate(this.reflector_h,[0 1 0], this.theta, this.t);
486     rotate(this.reflector_h,[0 0 1], this.phi, this.t);
487     % draw normals
488     if (this.normal_vectors == true)
489         corners = [this.t, this.intersections, (this.t+(this.dir_rot ←
            (:,1)+this.dir_rot(:,2)+this.dir_rot(:,3))*0.5), this. ←
            intersections+this.normals*this.r*0.5]';
490         num_corners = size(corners, 1);
491         connect = [1:num_corners/2; num_corners/2+1:num_corners]';
492         this.normals_h = patch('vertices', corners, 'faces', connect, ←
            'facecolor', 'g', 'edgecolor', 'black', 'facealpha', 1);
493     end
494     catch exception
495         exception = addCause(MException('Retroreflector:DrawingImpossible ←
            ', ...
496             'Retroreflector: drawing impossible'), exception);
497         throw(exception);
498     end
499     ret = this;
500 end
501
502 function ret = draw_normal_vectors(this, varargin)
503     % decide whether normals should be drawn or not
504     % uses: normal_vectors

```

```

505         if (nargin>1 && islogical(varargin{1}))
506             this.normal_vectors = logical(varargin{1});
507         end
508         ret = this.normal_vectors;
509     end
510 end
511 end

```

B.4. Das Simulationsscript

```

1  % -----
2  % geometric analysis of retroreflectors
3  % Thorsten Lamb
4  % DESY – MSK
5  % -----
6
7  clf
8  clear all
9  whitebg('black');
10
11 % parameters for koos
12 s_koos = 2;           % plotting size of koos
13
14 % parameters for beam
15 t_beam = [0;0;70.5]; % base
16 v_beam = [1;0;0];   % initial direction
17 theta_beam = 0;     % up/down
18 phi_beam = 0;       % left/right
19
20 % parameters for retroreflector
21 t_retro = [35;0;65]; % base 35 – 95
22 r_retro = 12.7;     % radius of reflector
23 psi_retro = 90;     % rotate
24 theta_retro = 0;    % up/down
25 phi_retro = 180;    % left/right
26
27 % parameters for prism
28 s_roof = 10;        % base length (only for plotting, size is infinite)
29 t_roof = [15; 3; 59.5]; % base
30 psi_roof = -92;     % rotate
31 theta_roof = 0;     % left/right
32 phi_roof = 0;       % up/down
33
34 % parameters for colimator
35 % face is parallel to y/z-face
36 s_col = 1;          % size (only for plotting, size is infinite)
37 t_col = [0;-6;70.5]; % base
38
39 % parameters for spheres at intersections
40 sphere_size = 1.5;  % radius
41

```

Anhang B. Quellcode zur Simulation der Delayline

```
42 % -----
43
44
45 % incoming beam
46 beam_in = beam(t_beam, v_beam);
47 beam_in.set_dir_spher_rel(phi_beam, theta_beam);
48
49 % reflection at retroreflector_1
50 reflector_1 = retroreflector(t_retro, r_retro, psi_retro, theta_retro, phi_retro  ↔
    );
51 reflector_1.set_beam_in(bean_in);
52 reflector_1.update_all;
53
54 beam_intermediate_1 = reflector_1.beam_out;
55
56 % calculations for roof mirror (workaround with beam and mirror class)
57 v = [beam([0;0;0],[0;1;0]) beam([0;0;0],[1;0;0]) beam([0;0;0],[1;0;0])];
58 v(1).set_dir_spher_rel(phi_roof, 0);
59 v(2).set_dir_spher_rel(phi_roof, -45+theta_roof);
60 v(3).set_dir_spher_rel(phi_roof, 45+theta_roof);
61 R_psi = [1 0 0;
62          0 cosd(psi_roof) -sind(psi_roof);
63          0 sind(psi_roof) cosd(psi_roof)];
64 v(1).set_v(R_psi*v(1).v);
65 v(2).set_v(R_psi*v(2).v);
66 v(3).set_v(R_psi*v(3).v);
67 roof_top = mirror(t_roof,v(1).v,v(2).v);
68 roof_bottom = mirror(t_roof,v(1).v,v(3).v);
69
70 % reflection at roof mirror
71 roof_bottom.set_beam_in(bean_intermediate_1);
72 roof_bottom.update_all;
73
74 beam_roof = roof_bottom.beam_out;
75
76 roof_top.set_beam_in(bean_roof);
77 roof_top.update_all;
78
79 beam_intermediate_2 = roof_top.beam_out;
80
81 % store first intersections because reflector_1 is used twice
82 intersections = [beam_in.t, reflector_1.intersections, ...
83                 roof_bottom.intersection, roof_top.intersection]';
84
85 % 2nd Reflection at reflector_1
86 reflector_1.set_beam_in(bean_intermediate_2);
87 reflector_1.update_all;
88
89 beam_out = reflector_1.beam_out;
90
91 % use mirror as workaround for collimator
92 collimator = mirror(t_col,[0;0;1]*s_col,[0;-1;0]*s_col);
```

```

93 collimator.set_beam_in(beam_out);
94 collimator.update_all;
95
96 % draw reflectors
97 %reflector_1.draw_normal_vectors(true);
98 reflector_1.draw;
99
100 % draw beam
101 if (dot(collimator.normal, collimator.beam_in.v) < 0)
102     intersections = [intersections', reflector_1.intersections, ...
103         collimator.intersection]';
104 else
105     intersections = [intersections', reflector_1.intersections]';
106     intersections = [intersections; intersections(end,:)+reflector_1.beam_out.v ←
        '*80];
107 end
108 num_intersections = size(intersections, 1);
109 connect=[1:num_intersections-1; 2:num_intersections]';
110 patch_beam = patch('vertices', intersections, 'faces', connect, ...
111     'edgecolor','red', 'LineWidth', 1.5);
112
113 % terminal output and diagnostics
114 beam_out.display
115 deviation_yz = collimator.intersection - collimator.t;
116 deviation_mm = norm(deviation_yz);
117 angle_d = 180-acosd(dot(beam_out.v, collimator.normal) ...
118     /norm(beam_out.v)/norm(collimator.normal));
119 distance = 0;
120 for i=2:num_intersections
121     distance = distance + abs(norm(intersections(i-1,:)-intersections(i,:)));
122 end
123 fprintf(['deviation y/z (mm) : \t\t%.3E \t%.3E \n\ndeviation absolute (mm) :', ←
        ...
124     '\t%.3E \n\ndistance (mm) : \t\t%.3E \n\nangle (degree) : \t\t%.3E \n\n'], ←
        ...
125     deviation_yz(2), deviation_yz(3), deviation_mm, distance, angle_d);
126
127 % draw roof mirror
128 corners = [-roof_top.v1'/2; roof_top.v1'/2; roof_top.v1'/2+roof_top.v2';
129     roof_top.v2'-roof_top.v1'/2; roof_top.v1'/2+roof_bottom.v2';
130     roof_bottom.v2'-roof_top.v1'/2];
131 corners = corners*s_roof + repmat(roof_top.t,1,6)';
132 faces = [1 2 3 4; 1 2 5 6];
133 patch_roof = patch('vertices',corners,'faces',faces, ...
134     'facecolor', 'g','edgecolor','black','facealpha',.5);
135
136 % draw coordinate system in red to improve orientation
137 %corners = [0 0 0; 0 0 s_koos; 0 s_koos s_koos; 0 s_koos 0;
138     s_koos 0 s_koos; s_koos 0 0; s_koos s_koos 0 ];
139 %faces = [1 2 3 4; 1 2 5 6; 1 4 7 6];
140 %patch_koos = patch('vertices',corners,'faces',faces, ...
141 %     'facecolor', 'r','edgecolor','white','facealpha',.3);

```

Anhang B. Quellcode zur Simulation der Delayline

```
142
143 % draw collimator face in blue
144 corners = [0 0 0; 0 0 s_col; 0 s_col s_col;
145            0 s_col 0; 0 -s_col 0; 0 -s_col s_col;
146            0 0 -s_col; 0 s_col -s_col; 0 -s_col -s_col ];
147 corners = corners + repmat(collimator.t,1,9)';
148 faces = [1 2 3 4; 1 2 6 5; 1 7 8 4; 1 5 9 7];
149 patch_collimator = patch('vertices',corners,'faces',faces, ...
150    'facecolor','b','edgecolor','white','facealpha',.3);
151
152 % draw spheres at intersections
153 % for num=2:num_intersections-1
154 % [sphere_x,sphere_y,sphere_z] = sphere;
155 % surf(sphere_x*sphere_size+intersections(num,1), ...
156 %      sphere_y*sphere_size+intersections(num,2), ...
157 %      sphere_z*sphere_size+intersections(num,3), ...
158 %      'facecolor','r','edgecolor','none','facealpha',.15);
159 % end
160
161 % plot properties
162 view(3)
163 axis equal
164 grid off
165 axis off
166 rotate3d on
```

Anhang C.

Quellcode zur Simulation des Messprinzips

```
1 function varargout = shortlink_simulation(varargin)
2 % SHORTLINK_SIMULATION M-file for shortlink_simulation.fig
3 %   SHORTLINK_SIMULATION, by itself, creates a new SHORTLINK_SIMULATION   ↔
   or raises the existing
4 %   singleton *.
5 %
6 %   H = SHORTLINK_SIMULATION returns the handle to a new   ↔
   SHORTLINK_SIMULATION or the handle to
7 %   the existing singleton *.
8 %
9 %   SHORTLINK_SIMULATION('CALLBACK',hObject,eventData,handles,...) calls the   ↔
   local
10 %   function named CALLBACK in SHORTLINK_SIMULATION.M with the given   ↔
   input arguments.
11 %
12 %   SHORTLINK_SIMULATION('Property','Value',...) creates a new   ↔
   SHORTLINK_SIMULATION or raises the
13 %   existing singleton *. Starting from the left, property value pairs are
14 %   applied to the GUI before shortlink_simulation_OpeningFcn gets called. An
15 %   unrecognized property name or invalid value makes property application
16 %   stop. All inputs are passed to shortlink_simulation_OpeningFcn via varargin.
17 %
18 %   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
19 %   instance to run (singleton)".
20 %
21 % See also: GUIDE, GUIDATA, GUIHANDLES
22
23 % Edit the above text to modify the response to help shortlink_simulation
24
25 % Last Modified by GUIDE v2.5 07-Nov-2010 13:26:06
26
27 % Begin initialization code - DO NOT EDIT
28 gui_Singleton = 1;
29 gui_State = struct('gui_Name',    mfilename, ...
30                   'gui_Singleton', gui_Singleton, ...
31                   'gui_OpeningFcn', @shortlink_simulation_OpeningFcn, ...
32                   'gui_OutputFcn', @shortlink_simulation_OutputFcn, ...
33                   'gui_LayoutFcn', [] , ...
34                   'gui_Callback', []);
```

Anhang C. Quellcode zur Simulation des Messprinzips

```
35 if nargin && ischar(varargin{1})
36     gui_State.gui_Callback = str2func(varargin{1});
37 end
38
39 if narginout
40     [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
41 else
42     gui_mainfcn(gui_State, varargin{:});
43 end
44 % End initialization code – DO NOT EDIT
45
46 function shortlink_simulation_OpeningFcn(hObject, ~, handles, varargin)
47 handles.output = hObject;
48 global harmonics;
49
50 listener_slider_timing = addlistener(handles.slider_timing,'Action',@(src,evnt) ←
    cb_slider_timing(handles,src,evnt)); %#ok<NASGU>
51 listener_slider_A_ref = addlistener(handles.slider_A_ref,'Action',@(src,evnt) ←
    cb_slider_A_ref(handles,src,evnt)); %#ok<NASGU>
52 listener_slider_sigma_ref = addlistener(handles.slider_sigma_ref,'Action',@(src, ←
    evnt)cb_slider_sigma_ref(handles,src,evnt)); %#ok<NASGU>
53 listener_slider_A_link = addlistener(handles.slider_A_link,'Action',@(src,evnt) ←
    cb_slider_A_link(handles,src,evnt)); %#ok<NASGU>
54 listener_slider_sigma_link = addlistener(handles.slider_sigma_link,'Action',@( ←
    src,evnt)cb_slider_sigma_link(handles,src,evnt)); %#ok<NASGU>
55 listener_slider_A_LO = addlistener(handles.slider_A_LO,'Action',@(src,evnt) ←
    cb_slider_A_LO(handles,src,evnt)); %#ok<NASGU>
56 listener_slider_Phi_LO = addlistener(handles.slider_Phi_LO,'Action',@(src,evnt) ←
    cb_slider_Phi_LO(handles,src,evnt)); %#ok<NASGU>
57 listener_slider_mix_off = addlistener(handles.slider_mix_off,'Action',@(src,evnt) ←
    )cb_slider_mix_off(handles,src,evnt)); %#ok<NASGU>
58
59 set(handles.timing_min,'String',get(handles.slider_timing,'Min'));
60 set(handles.timing_max,'String',get(handles.slider_timing,'Max'));
61 set(handles.slider_timing,'Value',1/2/216.666e-6);
62 set(handles.edit_timing,'String',num2str(get(handles.slider_timing,'Value'), ' ←
    %6.1f'));
63
64 set(handles.A_ref_min,'String',get(handles.slider_A_ref,'Min'));
65 set(handles.A_ref_max,'String',get(handles.slider_A_ref,'Max'));
66 set(handles.slider_A_ref,'Value', 1);
67 set(handles.edit_A_ref,'String',num2str(get(handles.slider_A_ref,'Value'), '%4.2 ←
    f'));
68
69 set(handles.sigma_ref_min,'String',get(handles.slider_sigma_ref,'Min'));
70 set(handles.sigma_ref_max,'String',get(handles.slider_sigma_ref,'Max'));
71 set(handles.slider_sigma_ref,'Value', 10);
72 set(handles.edit_sigma_ref,'String',num2str(get(handles.slider_sigma_ref,'Value' ←
    ), '%5.0f'));
73
74 set(handles.A_link_min,'String',get(handles.slider_A_link,'Min'));
75 set(handles.A_link_max,'String',get(handles.slider_A_link,'Max'));
```

```

76 set(handles.slider_A_link,'Value', 1);
77 set(handles.edit_A_link,'String',num2str(get(handles.slider_A_link,'Value'), ' ←
    %4.2f'));
78
79 set(handles.sigma_link_min,'String',get(handles.slider_sigma_link,'Min'));
80 set(handles.sigma_link_max,'String',get(handles.slider_sigma_link,'Max'));
81 set(handles.slider_sigma_link,'Value', 10);
82 set(handles.edit_sigma_link,'String',num2str(get(handles.slider_sigma_link,' ←
    Value'), '%5.0f'));
83
84 set(handles.A_LO_min,'String',get(handles.slider_A_LO,'Min'));
85 set(handles.A_LO_max,'String',get(handles.slider_A_LO,'Max'));
86 set(handles.slider_A_LO,'Value', 1);
87 set(handles.edit_A_LO,'String',num2str(get(handles.slider_A_LO,'Value'), '%4.2f' ←
    ));
88
89 set(handles.Phi_LO_min,'String',get(handles.slider_Phi_LO,'Min'));
90 set(handles.Phi_LO_max,'String',get(handles.slider_Phi_LO,'Max'));
91 set(handles.slider_Phi_LO,'Value', 0.5);
92 set(handles.edit_Phi_LO,'String',num2str(get(handles.slider_Phi_LO,'Value'), ' ←
    %4.2f'));
93
94 set(handles.mix_off_min,'String',get(handles.slider_mix_off,'Min'));
95 set(handles.mix_off_max,'String',get(handles.slider_mix_off,'Max'));
96 set(handles.slider_mix_off,'Value', 0);
97 set(handles.edit_mix_off,'String',num2str(get(handles.slider_mix_off,'Value'), ' ←
    %4.2f'));
98
99 set(handles.group_simulation,'SelectedObject', handles.radio_dirac);
100 set(handles.slider_sigma_ref, 'Enable','off');
101 set(handles.edit_sigma_ref, 'Enable','off');
102 set(handles.slider_sigma_link, 'Enable','off');
103 set(handles.edit_sigma_link, 'Enable','off');
104
105 modulation(handles);
106
107 xlabel(handles.modulation_plot, 'frequency f/f_0');
108 ylabel(handles.modulation_plot, 'intensity (a.u.)');
109 xlim(handles.modulation_plot, [0, harmonics+1]);
110
111 xlabel(handles.mixer_plot, 'timing [t/T(9.75GHz)]');
112 ylabel(handles.mixer_plot, 'amplitude (a.u.) / angle (rad/pi)');
113 grid(handles.mixer_plot, 'on');
114 xlim(handles.mixer_plot, [-1, 1]);
115 ylim(handles.mixer_plot, [-2, 3]);
116 legend(handles.mixer_plot, 'amplitude (modulation)', 'cosine of mixer phase', ' ←
    mixer out', 'phase (modulation+phase shifter)', 'Location', 'SouthEastOutside' ←
    );
117 set(legend(handles.mixer_plot), 'TextColor',[0 0 0], 'EdgeColor',[0 0 0]);
118
119 guidata(hObject, handles);
120

```

Anhang C. Quellcode zur Simulation des Messprinzips

```

121 function varargout = shortlink_simulation_OutputFcn(~, ~, handles)
122 varargout{1} = handles.output;
123
124
125 function modulation(handles)
126 global harmonics;
127 %----- get values from GUI
128 dt = get(handles.slider_timing, 'Value')*1e-12; % timing
129 A_ref = get(handles.slider_A_ref, 'Value'); % reference amplitude
130 A_link = get(handles.slider_A_link, 'Value'); % link amplitude
131 sigma_ref = get(handles.slider_sigma_ref, 'Value')*10E-15; % reference pulse width ( ↔
FWHM)
132 sigma_link = get(handles.slider_sigma_link, 'Value')*10E-15; % link pulse width ( ↔
FWHM)
133 A_LO = get(handles.slider_A_LO, 'Value'); % LO amplitude
134 phi_shift = get(handles.slider_Phi_LO, 'Value')*pi; % additional phase at phase ↔
shifter before mixer
135 mix_off = get(handles.slider_mix_off, 'Value'); % DC offset @ mixer
136 %----- fixed parameters
137 harmonics = 45; % harmonics to show at modulation plot
138 res = 25; % resolution (points per harmonic)
139 f_0 = 216.6666e6; % base frequency
140 harm = 45; % harmonic to use for mixing
141 f = harm*f_0; % frequency of harmonic
142 mixer_axis = linspace(-1,1,25*res+1); % axis for mixer plot [t/T] range: \pm T
143 phi_off = 2*pi*(mixer_axis*dt*f); % offset range of superposed pulse trains (with ↔
respect to working point) for mixer calculation
144 f_axis=linspace(0, (harmonics+1)*f_0, (harmonics+1)*res+1); % frequency axis for ↔
modulation plot
145 %----- calculate modulation
146 switch get(get(handles.group_simulation, 'SelectedObject'), 'Tag') % get tag of ↔
selected radio button
147 case 'radio_dirac'
148 env = sqrt((A_ref^2+A_link^2)/4 + A_ref*A_link/2*cos(2*pi*f_axis*dt));
149 A_ref_mod = A_ref;
150 A_link_mod = A_link;
151 case 'radio_gauss'
152 sigma_ref = sigma_ref/(2*sqrt(2*log(2))); sigma_link = sigma_link/(2* ↔
sqrt(2*log(2))); % convert from FWHM
153 env = sqrt(A_ref^2/4./exp(1/2*(2*pi*f_axis).^2*sigma_ref^2)+A_link^2/4./ ↔
exp(1/2*(2*pi*f_axis).^2*sigma_link^2)+ ...
154 1/2*A_ref*A_link./sqrt(exp(1/2*(2*pi*f_axis).^2*(sigma_ref^2+ ↔
sigma_link^2))).*cos(-2*pi*f_axis*dt));
155 A_ref_mod = exp(-2*(pi*f*sigma_ref)^2)*A_ref;
156 A_link_mod = exp(-2*(pi*f*sigma_link)^2)*A_link;
157 case 'radio_sech2'
158 sigma_ref = sigma_ref/(2*acosh(sqrt(2))); sigma_link = sigma_link/(2* ↔
acosh(sqrt(2))); % convert from FWHM
159 env = sqrt(1/16*pi^2*(2*pi*f_axis).^2.*(A_link^2*sigma_link^2*csch(pi^2* ↔
f_axis*sigma_link).^2 + A_ref^2*sigma_ref^2*csch(pi^2*f_axis*
sigma_ref).^2 + ...
160 2*A_ref*sigma_ref*A_link*sigma_link*csch(pi^2*f_axis*sigma_ref).*csch ↔

```

```

                (pi^2*f_axis*sigma_link).*cos(-dt*2*pi*f_axis));
161     A_ref_mod = pi/2*sigma_ref*2*pi*f*csch(pi^2*sigma_ref*f)*A_ref;
162     A_link_mod = pi/2*sigma_link*2*pi*f*csch(pi^2*sigma_link*f)*A_link;
163 end
164 %----- calculate dirac comb
165 dirac_comb = ones(harmonics*3,2)*(-1);
166 for i=1:harmonics
167     dirac_comb(3*i-2:3*i,1)=i;
168     dirac_comb(3*i-1,2) = env(i*res+1);
169 end
170 %----- calculate mixing
171 Amp = sqrt(A_ref_mod^2+A_link_mod^2+2*A_ref_mod*A_link_mod*cos(phi_off)); % ←
    amplitude of modulated pulse train
172 Ph = atan2(sin(phi_off),A_ref_mod/A_link_mod+cos(phi_off)) + phi_shift; % actual ←
    phase offset of modulated pulse train
173 cos_Ph = cos(Ph);
174 mixer = A_L0*Amp/2.*cos_Ph + mix_off; % mixing
175 %----- plot modulation
176 ylim(handles.modulation_plot, [0, max(env)+0.1]);
177 plot(handles.modulation_plot, f_axis/f_0, env, dirac_comb(:,1), dirac_comb(:,2)) ←
    ;
178 %----- plot mixing
179 plot(handles.mixer_plot, ...
180     mixer_axis, Amp/2, ...
181     mixer_axis, cos_Ph, ...
182     mixer_axis, mixer, ...
183     mixer_axis, Ph/pi, ...
184     0,mixer(floor(25*res/2)+1),'.', ...
185     [0 ; 0], [0 ; mixer(floor(25*res/2)+1)],'-', ...
186     'MarkerSize', 15);
187 set(handles.result,'String',num2str(mixer(25*res+1),'%6.3f'));
188
189
190 function modulation_plot_CreateFcn(hObject, ~, ~) %#ok<DEFNU>
191 colorOrder = [ 1 0.7 0.7 ; 0 0 1 ];
192 set(hObject, 'ColorOrder', colorOrder);
193
194 function mixer_plot_CreateFcn(hObject, ~, ~) %#ok<DEFNU>
195 colorOrder = [ 0.5 1 1 ; 0.5 1 0.5 ; 0 0 1 ; 1 0 0 ; 1 0 1 ; 1 0 1 ];
196 set(hObject, 'ColorOrder', colorOrder);
197
198
199 function slider_timing_CreateFcn(hObject, ~, ~) %#ok<DEFNU>
200 if isequal(get(hObject,'BackgroundColor'), get(0,' ←
    defaultUicontrolBackgroundColor'))
201     set(hObject,'BackgroundColor',[.9 .9 .9]);
202 end
203
204 function edit_timing_Callback(~, ~, handles) %#ok<DEFNU>
205 editValue = str2double(get(handles.edit_timing,'String'));
206 if (isempty(editValue) || isnan (editValue) || editValue < get(handles. ←
    slider_timing,'Min'))

```

Anhang C. Quellcode zur Simulation des Messprinzips

```
207     editValue = get(handles.slider_timing,'Min');
208 elseif (editValue > get(handles.slider_timing,'Max'))
209     editValue = get(handles.slider_timing,'Max');
210 end
211 set(handles.edit_timing,'String', num2str(editValue, '%6.1f'));
212 set(handles.slider_timing,'Value', editValue);
213 modulation(handles);
214
215 function edit_timing_CreateFcn(hObject, ~, ~) ##ok<DEFNU>
216 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,' ←
    defaultUicontrolBackgroundColor'))
217     set(hObject,'BackgroundColor','white');
218 end
219
220 function cb_slider_timing(handles, ~, ~)
221 set(handles.edit_timing,'String',num2str(get(handles.slider_timing,'Value'), ' ←
    %6.1f'));
222 modulation(handles);
223
224
225 function slider_A_ref_CreateFcn(hObject, ~, ~) ##ok<DEFNU>
226 if isequal(get(hObject,'BackgroundColor'), get(0,' ←
    defaultUicontrolBackgroundColor'))
227     set(hObject,'BackgroundColor',[.9 .9 .9]);
228 end
229
230 function edit_A_ref_Callback(~, ~, handles) ##ok<DEFNU>
231 editValue = str2double(get(handles.edit_A_ref,'String'));
232 min = get(handles.slider_A_ref,'Min');
233 max = get(handles.slider_A_ref,'Max');
234 if (isempty(editValue) || isnan (editValue) || editValue < min)
235     editValue = min;
236 elseif (editValue > max)
237     editValue = max;
238 end
239 set(handles.edit_A_ref,'String', num2str(editValue, '%4.2f'));
240 set(handles.slider_A_ref,'Value', editValue);
241 modulation(handles);
242
243 function edit_A_ref_CreateFcn(hObject, ~, ~) ##ok<DEFNU>
244 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,' ←
    defaultUicontrolBackgroundColor'))
245     set(hObject,'BackgroundColor','white');
246 end
247
248 function cb_slider_A_ref(handles, ~, ~)
249 set(handles.edit_A_ref,'String',num2str(get(handles.slider_A_ref,'Value'), '%4.2 ←
    f'));
250 modulation(handles);
251
252
253 function slider_sigma_ref_CreateFcn(hObject, ~, ~) ##ok<DEFNU>
```

```

254 if isequal(get(hObject,'BackgroundColor'), get(0,' ←
      defaultUicontrolBackgroundColor'))
255     set(hObject,'BackgroundColor',[.9 .9 .9]);
256 end
257
258 function edit_sigma_ref_Callback(~, ~, handles) %#ok<DEFNU>
259 editValue = str2double(get(handles.edit_sigma_ref,'String'));
260 min = get(handles.slider_sigma_ref,'Min');
261 max = get(handles.slider_sigma_ref,'Max');
262 if (isempty(editValue) || isnan (editValue) || editValue < min)
263     editValue = min;
264 elseif (editValue > max)
265     editValue = max;
266 end
267 set(handles.edit_sigma_ref,'String', num2str(editValue, '%5.0f'));
268 set(handles.slider_sigma_ref,'Value', editValue);
269 modulation(handles);
270
271 function edit_sigma_ref_CreateFcn(hObject, ~, ~) %#ok<DEFNU>
272 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,' ←
      defaultUicontrolBackgroundColor'))
273     set(hObject,'BackgroundColor','white');
274 end
275
276 function cb_slider_sigma_ref(handles, ~, ~)
277 set(handles.edit_sigma_ref,'String',num2str(get(handles.slider_sigma_ref,'Value' ←
      ), '%5.0f'));
278 modulation(handles);
279
280
281 function slider_A_link_CreateFcn(hObject, ~, ~) %#ok<DEFNU>
282 if isequal(get(hObject,'BackgroundColor'), get(0,' ←
      defaultUicontrolBackgroundColor'))
283     set(hObject,'BackgroundColor',[.9 .9 .9]);
284 end
285
286 function edit_A_link_Callback(~, ~, handles) %#ok<DEFNU>
287 editValue = str2double(get(handles.edit_A_link,'String'));
288 min = get(handles.slider_A_link,'Min');
289 max = get(handles.slider_A_link,'Max');
290 if (isempty(editValue) || isnan (editValue) || editValue < min)
291     editValue = min;
292 elseif (editValue > max)
293     editValue = max;
294 end
295 set(handles.edit_A_link,'String', num2str(editValue, '%4.2f'));
296 set(handles.slider_A_link,'Value', editValue);
297 modulation(handles);
298
299 function edit_A_link_CreateFcn(hObject, ~, ~) %#ok<DEFNU>
300 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,' ←
      defaultUicontrolBackgroundColor'))

```

Anhang C. Quellcode zur Simulation des Messprinzips

```
301     set(hObject,'BackgroundColor','white');
302 end
303
304 function cb_slider_A_link(handles,~,~)
305 set(handles.edit_A_link,'String',num2str(get(handles.slider_A_link,'Value'),'↔
    %4.2f'));
306 modulation(handles);
307
308
309 function slider_sigma_link_CreateFcn(hObject,~,~) %#ok<DEFNU>
310 if isequal(get(hObject,'BackgroundColor'),get(0,'↔
    defaultUicontrolBackgroundColor'))
311     set(hObject,'BackgroundColor',[.9 .9 .9]);
312 end
313
314 function edit_sigma_link_Callback(~,~,handles) %#ok<DEFNU>
315 editValue = str2double(get(handles.edit_sigma_link,'String'));
316 min = get(handles.slider_sigma_link,'Min');
317 max = get(handles.slider_sigma_link,'Max');
318 if (isempty(editValue) || isnan(editValue) || editValue < min)
319     editValue = min;
320 elseif (editValue > max)
321     editValue = max;
322 end
323 set(handles.edit_sigma_link,'String',num2str(editValue,'%5.0f'));
324 set(handles.slider_sigma_link,'Value',editValue);
325 modulation(handles);
326
327 function edit_sigma_link_CreateFcn(hObject,~,~) %#ok<DEFNU>
328 if ispc && isequal(get(hObject,'BackgroundColor'),get(0,'↔
    defaultUicontrolBackgroundColor'))
329     set(hObject,'BackgroundColor','white');
330 end
331
332 function cb_slider_sigma_link(handles,~,~)
333 set(handles.edit_sigma_link,'String',num2str(get(handles.slider_sigma_link,'↔
    Value'),' %5.0f'));
334 modulation(handles);
335
336
337 function slider_A_LO_CreateFcn(hObject,~,~) %#ok<DEFNU>
338 if isequal(get(hObject,'BackgroundColor'),get(0,'↔
    defaultUicontrolBackgroundColor'))
339     set(hObject,'BackgroundColor',[.9 .9 .9]);
340 end
341
342 function edit_A_LO_Callback(~,~,handles) %#ok<DEFNU>
343 editValue = str2double(get(handles.edit_A_LO,'String'));
344 min = get(handles.slider_A_LO,'Min');
345 max = get(handles.slider_A_LO,'Max');
346 if (isempty(editValue) || isnan(editValue) || editValue < min)
347     editValue = min;
```

```

348 elseif (editValue > max)
349     editValue = max;
350 end
351 set(handles.edit_A_LO,'String', num2str(editValue, '%4.2f'));
352 set(handles.slider_A_LO,'Value', editValue);
353 modulation(handles);
354
355 function edit_A_LO_CreateFcn(hObject, ~, ~) %#ok<DEFNU>
356 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,' ←
    defaultUicontrolBackgroundColor'))
357     set(hObject,'BackgroundColor','white');
358 end
359
360 function cb_slider_A_LO(handles, ~, ~)
361 set(handles.edit_A_LO,'String',num2str(get(handles.slider_A_LO,'Value'), '%4.2f' ←
    ));
362 modulation(handles);
363
364
365 function slider_Phi_LO_CreateFcn(hObject, ~, ~) %#ok<DEFNU>
366 if isequal(get(hObject,'BackgroundColor'), get(0,' ←
    defaultUicontrolBackgroundColor'))
367     set(hObject,'BackgroundColor',[.9 .9 .9]);
368 end
369
370 function edit_Phi_LO_Callback(~, ~, handles) %#ok<DEFNU>
371 editValue = str2double(get(handles.edit_Phi_LO,'String'));
372 min = get(handles.slider_Phi_LO,'Min');
373 max = get(handles.slider_Phi_LO,'Max');
374 if (isempty(editValue) || isnan (editValue) || editValue < min)
375     editValue = min;
376 elseif (editValue > max)
377     editValue = max;
378 end
379 set(handles.edit_Phi_LO,'String', num2str(editValue, '%4.2f'));
380 set(handles.slider_Phi_LO,'Value', editValue);
381 modulation(handles);
382
383 function edit_Phi_LO_CreateFcn(hObject, ~, ~) %#ok<DEFNU>
384 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,' ←
    defaultUicontrolBackgroundColor'))
385     set(hObject,'BackgroundColor','white');
386 end
387
388 function cb_slider_Phi_LO(handles, ~, ~)
389 set(handles.edit_Phi_LO,'String',num2str(get(handles.slider_Phi_LO,'Value'), ' ←
    %4.2f'));
390 modulation(handles);
391
392
393 function slider_mix_off_CreateFcn(hObject, ~, ~) %#ok<DEFNU>
394 if isequal(get(hObject,'BackgroundColor'), get(0,' ←

```

Anhang C. Quellcode zur Simulation des Messprinzips

```
        defaultUicontrolBackgroundColor'))
395     set(hObject,'BackgroundColor',[.9 .9 .9]);
396 end
397
398 function edit_mix_off_Callback(~, ~, handles) %#ok<DEFNU>
399 editValue = str2double(get(handles.edit_mix_off,'String'));
400 min = get(handles.slider_mix_off,'Min');
401 max = get(handles.slider_mix_off,'Max');
402 if (isempty(editValue) || isnan (editValue) || editValue < min)
403     editValue = min;
404 elseif (editValue > max)
405     editValue = max;
406 end
407 set(handles.edit_mix_off,'String', num2str(editValue, '%4.2f'));
408 set(handles.slider_mix_off,'Value', editValue);
409 modulation(handles);
410
411 function edit_mix_off_CreateFcn(hObject, ~, ~) %#ok<DEFNU>
412 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,' ←
        defaultUicontrolBackgroundColor'))
413     set(hObject,'BackgroundColor','white');
414 end
415
416 function cb_slider_mix_off(handles, ~, ~)
417 set(handles.edit_mix_off,'String',num2str(get(handles.slider_mix_off,'Value'), ' ←
        %4.2f'));
418 modulation(handles);
419
420
421 function group_simulation_SelectionChangeFcn(~, eventdata, handles) %#ok< ←
        DEFNU>
422 switch get(eventdata.NewValue,'Tag') % Get Tag of selected object.
423     case 'radio_dirac'
424         set(handles.slider_sigma_ref, 'Enable','off');
425         set(handles.edit_sigma_ref, 'Enable','off');
426         set(handles.slider_sigma_link, 'Enable','off');
427         set(handles.edit_sigma_link, 'Enable','off');
428     case 'radio_gauss'
429         set(handles.slider_sigma_ref, 'Enable','on');
430         set(handles.edit_sigma_ref, 'Enable','on');
431         set(handles.slider_sigma_link, 'Enable','on');
432         set(handles.edit_sigma_link, 'Enable','on');
433     case 'radio_sech2'
434         set(handles.slider_sigma_ref, 'Enable','on');
435         set(handles.edit_sigma_ref, 'Enable','on');
436         set(handles.slider_sigma_link, 'Enable','on');
437         set(handles.edit_sigma_link, 'Enable','on');
438 end
439 modulation(handles)
440
441
442 function button_reset_Callback(hObject, eventdata, handles) %#ok<DEFNU>
```

```
443 shortlink_simulation_OpeningFcn(hObject, eventdata, handles);
```

Anhang D.

Literaturverzeichnis

- [Agr01] AGRAWAL, Govind P.: *Nonlinear Fiber Optics*. 3. Edition. San Diego : Academic Press, 2001.
- [BFG+10] BOCK, M. K. ; FELBER, M. ; GESSLER, P. ; HACKER, K. E. ; LUDWIG, F. ; SCHLARF, H. ; SCHMIDT, B. ; SCHULZ, S. ; WISSMANN, L.-G. ; ZEMELLA, J.: *Recent Developments of the Bunch Arrival Time Monitor with Femtosecond Resolution at FLASH*. In: *Proceedings of IPAC10, Kyoto, Japan*. 2010. URL: <http://accelconf.web.cern.ch/AccelConf/IPAC10/papers/weocmh02.pdf>.
- [Bou09] BOUSONVILLE, Michael: *Optische Übertragung phasensynchroner Taktsignale unter Verwendung des Wellenlängen-Multiplex-Verfahrens*. Dissertation. Technische Universität Darmstadt, 2009.
- [DR06] DIELS, Jean-Claude ; RUDOLPH, Wolfgang: *Ultrashort Laser Pulse Phenomena*. 2. Edition. Burlington : Academic Press, 2006.
- [Dam04] DAMASK, Jay N.: *Polarization Optics in Telecommunications*. 1. Edition. Springer Science+Business Media Inc., 2004.
- [Dem09] DEMTRÖDER, Wolfgang: *Experimentalphysik 2*. 5. Auflage. Berlin / Heidelberg : Springer-Verlag, 2009.
- [HSZ03] HACKBUSCH, Wolfgang ; SCHWARZ, Hans R. ; ZEIDLER, Eberhard: *Teubner-Taschenbuch der Mathematik*. 2. Auflage. Wiesbaden : B.G. Teubner Verlag / GWV Fachverlage GmbH, 2003.
- [LLS+07] LORBEER, Bastian ; LUDWIG, F. ; SCHLARF, H. ; WINTER, A.: *Noise and Drift Characterization of Direct Laser to RF Conversion Scheme for the Laser Based Synchronization System for Flash at DESY*. In: *Proceedings of PAC07, Albuquerque, USA*. 2007. URL: <http://accelconf.web.cern.ch/AccelConf/p07/PAPERS/MOPAN017.PDF>.

- [LML+07] LORBEER, Bastian ; MÜLLER, Jost ; LUDWIG, Frank ; LOEHL, Florian ; SCHLARB, Holger ; WINTER, Axel: *Noise and Drift Characterization of Critical Components for the Laser Based Synchronization System at Flash*. In: *Proceedings of DIPAC07, Venice, Italy*. 2007. URL: <http://accelconf.web.cern.ch/AccelConf/d07/papers/wepb08.pdf>.
- [LSC+07] LÖHL, Florian ; SCHLARB, H. ; CHEN, J. ; KÄRTNER, F. X. ; KIM, Jung-Won: *First Prototype of an Optical Cross-Correlation Based Fiber-Link Stabilization for the FLASH Synchronization System*. In: *Proceedings of DIPAC07, Venice, Italy*. 2007. URL: <http://accelconf.web.cern.ch/AccelConf/d07/papers/wepb16.pdf>.
- [LW05] LUTZ, Holger ; WENDT, Wolfgang: *Taschenbuch der Regelungstechnik*. 6., erweiterte Auflage. Frankfurt am Main : Wissenschaftlicher Verlag Harri Deutsch, 2005.
- [Lun08] LUNZE, Jan: *Automatisierungstechnik*. 2., überarbeitete Auflage. München : Oldenbourg Wissenschaftsverlag GmbH, 2008.
- [Maa93] MAAS, Stephen A.: *Microwave Mixers*. 2. Edition. Norwood : Artech House Inc., 1993.
- [Oz05] OZ OPTICS (Hrsg.): *collimators and focusers – pigtail style*. Datenblatt. 19. Feb. 2005. URL: http://www.ozoptics.com/ALLNEW_PDF/DTS0060.pdf (besucht am 03.09.2010).
- [PI08] PHYSIK INSTRUMENTE GMBH (Hrsg.): *M-112 Compact Micro-Translation Stage*. Datenblatt. 2008. URL: http://www.physikinstrumente.com/en/pdf/M110_Datasheet.pdf (besucht am 12.08.2010).
- [ST91] SALEH, Bahaa E. A. ; TEICH, Malvin C.: *Fundamentals of Photonics*. 1. Edition. New York : Wiley, 1991.
- [Sch04] SCHULZ, Gerd: *Regelungstechnik 1*. 2. Auflage. München : Oldenbourg Wissenschaftsverlag GmbH, 2004.
- [Sch10] SCHULZ, Sebastian: *Installation Progress of the Laser-based Synchronization System at FLASH*. Folien eines Vortrags, präsentiert auf dem 48th ICFA Advanced Beam Dynamics Workshop on Future Light Sources, SLAC National Accelerator Laboratory, Menlo Park, USA. 1.–5. März 2010. URL: https://slacportal.slac.stanford.edu/sites/ad_public/events/FLS2010/Lists/WorkingGroup6/Attachments/10/Progress_FLASH-LbSynSys_FLS2010_2010-03-04Schulz.pdf.

- [ZAB+09] ZEMELLA, Johann ; ARSOV, V. R. ; BOCK, M. K. ; FELBER, M. ; GESSLER, P. ; HACKER, K. E. ; LOEHL, F. ; LUDWIG, F. ; SCHLARB, H. ; SCHMIDT, B. ; WINTER, A. ; ZEMELLA, J. ; SCHULZ, S. ; WISSMANN, L.-G.: *RF-Based Detector for Measuring Fiber Length Changes with Sub-5 Femtosecond Long-Term Stability over 50h*. In: *Proceedings of FEL2009, Liverpool, UK*. 2009. URL: <http://accelconf.web.cern.ch/AccelConf/FEL2009/papers/froa05.pdf>.
- [Zem08] ZEMELLA, Johann: *Driftfreier Detektor zur Messung des Zeitversatzes zweier verschiedener Laserpulszüge*. Diplomarbeit. Universität Hamburg, 2008.