TAMPEREEN TEKNILLINEN YLIOPISTO

Janne Hyyti

THIRD-HARMONIC INTERFEROMETRIC FREQUENCY-RESOLVED OPTICAL GATING FOR INVESTIGATING ULTRAFAST OPTICAL PHENOMENA ON THE FEW-CYCLE SCALE

MASTER'S THESIS

Examiners:
Prof. Günter Steinmeyer,
Dr. Lasse Orsila
Examiners and topic approved by the Faculty Council of Faculty of Science and Environmental Engineering on 5 December 2012

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Teknis-luonnontieteellinen koulutusohjelma
JANNE HYYTI: Ultranopeiden optisten ilmiöiden tutkiminen taajudenkolmentamista hyödyntävällä, interferometrisella, taajuuserotteisella optisella näytteistämismenetelmällä muutaman optisen syklin mittaisten laserpulssien luokassa
Diplomityö, 89 sivua, 32 liitesivua
Marraskuu 2013
Pääaine: Teknillinen fysiikka
Tarkastajat: Prof. Günter Steinmeyer, TkT Lasse Orsila
Avainsanat: Ultranopea optiikka, laserpulssin mittaus, Kerr-epälineaarisuus

Työssä tutkitaan uutta, ultralyhyiden laserpulssien mittaamiseen soveltuvaa menetelmää, jossa pulssin muoto ja vaihe voidaan rekonstruoida mittausjäljestä ohjelmallisesti. Vaikka ensimmäiset taajudenkolmentamista hyödyntävällä, interferometrisella, taajuuserotteisella optisella näytteistämismenetelmällä (*engl.* third-harmonic interferometric frequency-resolved optical gating) tehdyt mittaukset on julkaistu, niiden täysipainoinen hyödyntäminen ei ole ollut mahdollista, sillä menetelmälle soveltuvaa, pulssin rekonstruoivaa ohjelmistoa ei ole ollut saatavilla. Ensimmäinen tällainen ohjelmisto esitellään tässä työssä. Ohjelma kykenee määrittämään mitatun laserpulssin täydellisesti, sillä sekä sähkökentän verhokäyrä että vaihe rekonstruoidaan iteratiivisesti. Pulssinrekonstruointiohjelmiston esittelyn lisäksi työssä tutkitaan mittausmenetelmän fysiikkaa yleisellä tasolla, ja johdetaan mittausjäljen rakennetta havainnollistava yhtälö.

Ohjelmiston avulla analysoidaan kaksi saksalaisessa Max Born instituutissa suoritettua mittausta, joissa femtosekuntiluokan laserpulssi kulkee joko puhtaasta titaanidioksidista tai piidioksidista koostuvan ohutkalvon lävitse. Mittausjälkien erilaisia modulaatiokomponentteja hyödyntämällä suoritetaan kuusi simulaatiota, joiden avulla rekonstruoidaan kolme pulssimuotoa kustakin näytteestä. Näiden välituloksien avulla kootaan edelleen kutakin mittausta vastaavat varsinaiset, täydellisesti määritetyt pulssit. Titaanidioksidikalvon läpäisseen laserpulssin pituudeksi määritetään 15,7 fs, kun piidioksidinäyttettä vastaavaksi pulssipituudeksi saadaan vain 10,1 fs. Mittausjärjestelmän komponenttien dispersiivisistä ominaisuuksista saadaan tietoa tutkimalla rakennettujen pulssien kokemaa ryhmäviivedispersiota. Tämä on ensimmäinen kerta, kun menetelmällä tuotetusta mittausjäljestä on rekonstruoitu pulssi.

Rekonstruoitujen laserpulssien kestoissa havaittu ero johtuu ohutkalvonäytteiden

erilaisista epälineaarisista optisista vasteista. Siinä missä laserpulssin indusoima epälineaarinen polarisaatio syntyy ja katoaa piidioksidinäytteessä miltei välittömästi pulssin sähkökentän voimakkuutta mukaillen, on titaanidioksidikalvon polarisaatiolla äärellinen elinaika – titaanidioksidikalvo pysyy hetken aikaa polarisoituna laserpulssin jo poistuttua materiaalista. Tämän vuorovaikutuksen tarkkaa mekanismia ei tunneta, mutta sen elinaika voidaan määrittää rekonstruoitujen pulssien avulla. Konvoloimalla piidioksidinäytteelle rekonstruoitu pulssi ja aikavakiolla 6,5 fs sovitettu yksipuoleinen eksponentiaalifunktio, saadaan uusi pulssimuoto, joka vastaa lähes täydellisesti titaanidioksidikalvolle rekonstruoitua pulssia. Määritetty aikavakio kuvaa titaanidioksidinäytteen eksponentiaalisesti vaimenevan epälineaarisen polarisaation elinaikaa.

Pohjimmaiset syyt sille, miksi vain titaanidioksidikalvossa havaitaan äärellinen polarisaation elinaika, löytyvät näytteiden ominaisuuksia tarkastelemalla. Ensimmäinen osatekijä on se, että epälineaarisen vuorovaikutuksen voimakkuutta kuvaava suure, kolmannen asteen suskeptibiliteetti, on titaanidioksidikalvolla monikymmenkertainen piidioksidikalvoon nähden. Toinen vaikuttava tekijä on näytemateriaalien energia-aukkojen leveyksien suuri ero. Laserpulssin sisältämät lähi-infrapuna-alueen sekä näkyvän valon fotonit ovat riittävän energisiä vuorovaikuttamaan titaanidioksidinäytteen kanssa monifotoniabsorption turvin, kun taas piidioksidinäytteen absorptio on materiaalin leveän energia-aukon vuoksi olematonta. Näiden ominaisuuksien vuoksi titaanidioksidikalvon vuorovaikutus laserpulssin kanssa on voimakkaampaa, mikä aikaansaa voimakkaammaan polarisaation. On mahdollista, että epälineaarisen prosessin elinaika on kytköksissä väliaineen suskeptibiliteetin voimakkuuteen.

Epälineaarisen polarisaation ultralyhyt elinaika on nopeimpia muotolukitetulla titaanisafiirilaserilla mitattuja luonnonilmiöitä. Esitelty pulssinrekonstruointiohjelmisto mahdollistaa tämän ilmiön perusteellisemman tutkimuksen, ja voi lisätä epälineaarisen optiikan tietämystä. Lisätutkimus voi johtaa titaanisafiirilaserin ultralyhyiden pulssien tuottamisessakin hyödynnettävän optisen Kerr-ilmiön elinajan selvittämiseen.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Science and Engineering
JANNE HYYTI: Third-Harmonic Interferometric Frequency-Resolved Optical Gating for Investigating Ultrafast Optical Phenomena on the Few-Cycle Scale
Master of Science Thesis, 89 pages, 32 Appendix pages
November 2013
Major: Advanced Engineering Physics
Examiners: Prof. Günter Steinmeyer, Dr. Lasse Orsila
Keywords: frequency-resolved optical gating, pulse characterization, Kerr nonlinearity

The recently introduced characterization technique of *third-harmonic interferometric frequency-resolved optical gating* for ultrashort laser pulses is investigated. The first pulse retrieval software for this complete characterization technique is presented and used to conduct pulse retrievals for the first time. In addition, the physics of this measurement technique is studied, and a simple equation describing the trace is derived.

The subjects for the retrieval procedure are the measured traces for femtosecond laser pulses that have been guided through a thin film sample of either pure titanium dioxide or pure silicon dioxide. These experiments were conducted in the Max Born Institute of Berlin, Germany in 2012. Different combinations of modulational components of the interferometric trace are used in the simulations to produce three pulses for each of the two samples. The retrieved pulses are combined to produce two representative pulses for the thin films, completely describing the electric field envelope and the phase of the measured laser pulses. Full width at half maximum pulse widths of 10.1 fs and 15.7 fs are measured for the pulses of silicon- and titanium dioxide samples, respectively. The retrieved pulses are further examined by analyzing their spectral phases and the experienced group delay dispersion.

The significant difference observed in the pulse durations for the two samples is attributed to multiphoton absorption processes in the titanium dioxide thin film, although the exact mechanism of the noninstantaneous third-order polarization remains unclear. The intensity envelopes of the reconstructed pulses are harnessed to study the lifetime of this process using a deconvolution strategy. By convolving the pulse for silicon dioxide with a one-sided exponential decay function with a time constant of 6.5 fs, a third pulse is produced, perfectly replicating the retrieved pulse shape for the titanium dioxide sample.

This is one of the fastest phenomena ever measured with a Ti:sapphire laser. The

measurement software presented in this work facilitates additional research on the subject, understanding of which could increase our knowledge of nonlinear optics. More light can be shed on the lifetime of the Kerr nonlinearity, a mechanism of elementary nature in the production of ultrashort pulses with the Ti:sapphire laser.

PREFACE

This thesis was done in Tampere University of Technology for the Optoelectronics Research Centre serving under the department of Science and Environmental Engineering during 2012–2013. Part of the programming for the pulse retrieval software presented in this work was done in Max Born Institute of Berlin, Germany, during spring 2012. The author would like to thank Prof. Günter Steinmeyer for the chance to visit his home institute, and for making this thesis possible. The guidance and support given by Prof. Steinmeyer is greatly appreciated, and was of enormous help in both the writing process and in the underlying work. Both Prof. Steinmeyer and Dr. Lasse Orsila are praised for their detailed and abundant remarks and suggestions, which steered this project to the finish line. Many thanks go to Prof. Susanta Kumar Das and the people at the Max Born Institute for conducting the measurements, and for introducing the experimental setup to the author.

The author is grateful for the support friends and family showed in this laborious time. Mother Tuula, father Eero, and sister Laura were always there when needed. An honourable mention is given to the servants of the Black Hand and the overlords of #MYSR for being awesome friends. With their help the occasional build-up of steam could be vented, and without them the completion of this thesis would not have been possible.

Janne Hyyti

Tampere, November the 19$^{\text{th}}$, 2013

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTS OF ABBREVIATIONS, SYMBOLS, AND PROGRAM PARAMETERS

The language of this thesis is American English. Names of software **functions** are written in bold letters, *parameters* in italic letters.

Abbreviations

3PA	Three-photon absorption
AC	Background-free intensity autocorrelation
BBO	Beta barium borate (β-BaB_2O_4) crystal
BS	Beam splitter
CCD	Charge-coupled device
DC	Direct current
DFT	Discrete Fourier transform
EMCCD	Electron multiplying charge-coupled device
FFT	Fast Fourier transform
FM-FROG	Fundamental modulation frequency-resolved optical gating
FM	Fundamental modulation
FROG	Frequency-resolved optical gating
FT	Fourier transform
FWHM	Full width at half maximum
FWM	Four-wave mixing
GDD	Group delay dispersion
GPA	Generalized projections algorithm
GSA	Gerchberg–Saxton algorithm
GUI	Graphical user interface
GVD	Group velocity dispersion
IAC	Interferometric autocorrelation
IFROG	Interferometric frequency-resolved optical gating
KLM	Kerr-lens mode-locking
NLS	Nonlinear Schrödinger equation
OMA	Optical multichannel analyzer
PICASO	Phase and intensity from correlation and spectrum only
sech^2	Hyperbolic secant squared
SESAM	Semiconductor saturable absorber mirror
SHFROG	Second-harmonic generation frequency-resolved optical gating
SHG	Second-harmonic generation
SHIFROG	Second-harmonic interferometric frequency-resolved optical gating

Abbreviations (continued)

SPIDER	Spectral phase interferometry for direct electric-field reconstruction
SPM	Self-phase modulation
SPP	Surface plasmon polariton
SSA	Stibenz–Steinmeyer algorithm
STHG	Surface third-harmonic generation
SVEA	Slowly-varying envelope approximation
TBP	Time-bandwidth product
THG	Third-harmonic generation
THIFROG	Third-harmonic interferometric frequency-resolved optical gating
TIVI	Temporal information via intensity
UV	Ultraviolet wavelength range
XUV	Extreme ultraviolet wavelength range

Symbols

All symbols are used as listed here unless indicated otherwise.

X	Scalar quantity
$\mathcal{X}$	Complex quantity
$\mathbf{X}$	Vector quantity
$\tilde{X}$	Rapidly oscillating quantity, function of time
$\mathscr{F}$	Forward Fourier transform
$\mathscr{F}^{-1}$	Inverse Fourier transform
$\Re[\mathcal{X}]$	Real part of a complex quantity $\mathcal{X}$
$\Im[\mathcal{X}]$	Imaginary part of a complex quantity $\mathcal{X}$
$\mathbb{C}$	The set of complex numbers
$\mathbb{R}$	The set of real numbers
$\mathbf{min}[f(x)]$	Minimum of function $f(x)$
$\mathbf{max}[f(x)]$	Maximum of function $f(x)$
$\underset{x}{\mathbf{arg\,min}}\ f(x)$	Returns the argument x that minimizes $f(x)$
$\tilde{\mathbf{D}}$	Electric displacement
$\tilde{\mathbf{D}}_{\mathrm{L}}$	Linear electric displacement
$\tilde{\mathbf{E}}$	Electric field strength
$\tilde{\mathbf{B}}$	Magnetic flux density
$\tilde{\mathbf{H}}$	Magnetic field strength
$\tilde{\mathbf{J}}$	Total current density
ρ_f	Free charge density
$\tilde{\mathbf{M}}$	Magnetization

Symbols (continued)

$\tilde{\mathbf{P}}$	Polarization
$\tilde{\mathbf{P}}_{\mathbf{L}}$	Linear polarization
$\tilde{\mathbf{P}}_{\mathbf{NL}}$	Nonlinear polarization
ϵ	Permittivity
ϵ_0	Vacuum permittivity
μ_0	Vacuum permeability
c_0	Vacuum speed of light
ω	(Angular) frequency
ω_0	(Angular) carrier frequency
$\omega_{\mathbf{pl}}$	(Angular) plasmon frequency
$\Delta\omega$	Frequency bandwidth, (angular) frequency offset from carrier frequency
$\delta\omega$	Frequency chirp
$\mathcal{A}$	Complex electric field envelope
A	Electric field envelope
k	Wave number
k_0	Vacuum wave number
$\chi^{(1)}$	Linear electric susceptibility
$\chi^{(3)}$	Third-order electric susceptibility
$\chi^{(3)}_{ijkl}$	Third-order electric susceptibility tensor
$\chi^{(n)}$	n^{th}-order electric susceptibility
D	Degeneracy factor, power division factor
I	Optical field intensity
n	Refractive index
n_0	Linear refractive index
n_2	Nonlinear refractive index
$\mathcal{E}$	Complex spectral amplitude of the electric field
$\tilde{\mathcal{E}}$	Complex temporal amplitude of the electric field
$\mathcal{E}^+$	One-sided spectrum of the complex electric field
E	Complex electric field strength
ϕ	Phase
$\phi_{\mathbf{mod}}$	Phase of second harmonic modulation in I_{SHIFROG}
v_g	Group velocity
T	Retarded time coordinate, time constant of exponential decay
λ	Wavelength
λ_0	Carrier wavelength
τ	Delay

Symbols (continued)

$\Delta\tau$	Delay step
Ω	Delay-frequency
$g(t-\tau)$	Gate function
S	Spectrogram, signal function
$E_{\mathbf{sig}}$	Signal field
$\hat{E}_{\mathbf{sig}}$	Fourier transform of signal field
$I_{\mathbf{FROG}}$	FROG intensity
$I_{\mathbf{FROG}}^{\mathbf{meas}}$	Measured FROG intensity
$I_{\mathbf{FROG}}^{(k)}$	FROG intensity of the k^{th} iteration
$I_{\mathbf{SHFROG}}$	Second-harmonic generation FROG intensity
$I_{\mathbf{SHIFROG}}$	Second-harmonic generation interferometric FROG intensity
$I_{\mathbf{FM\text{-}FROG}}$	Fundamental modulation FROG intensity
$I_{\mathbf{THIFROG}}$	Third-harmonic generation interferometric FROG intensity
$E^{(k)}$	Electric field of the k^{th} iteration
$E_{\mathbf{SH}}$	Second-harmonic generation electric field
$E_{\mathbf{TH}}$	Third-harmonic generation electric field
$E_{\mathbf{SHFROG}}$	Second-harmonic generation FROG electric field
$E_{\mathbf{THFROG}}$	Third-harmonic generation FROG electric field
Z	Functional distance
μ	Minimizing factor
α	Minimizing factor for FROG error
G	FROG error
f	Frequency
f_N	Nyquist frequency
$f_{\mathbf{max}}$	Maximum frequency
f_0	Carrier frequency
T_0	Carrier period
R	Response function
γ	Line width
Γ	Spectral line width
$\hbar$	Planck constant divided by 2π
L_c	Coherence length
E_g	Band gap
t	Time
t_r	Rise time
t_f	Fall time
h_k	k^{th} sample for discrete Fourier transform

Symbols (continued)

H_n	n^{th} sample in the Fourier domain
ν	Frequency
ν_c	Central frequency
$\varphi(\omega_L)''$	Group delay dispersion
ϕ_2	Total group delay dispersion
β_2	Group velocity dispersion
$(f * g)(t)$	Convolution of functions $f(t)$ and $g(t)$
$I_{\mathbf{SiO_2}}$	THIFROG intensity for silica thin film
$I_{\mathbf{TiO_2}}$	THIFROG intensity for titania thin film

Program Parameters

The parameters for running the main function of the pulse retrieval software are listed below.

pathD	Location of the D file containing the preprocessed trace
pathP	Location of the P file containing the parameters of preprocessed trace
savePath	Path for the output files
scenario	Name of the sample included in output filenames
tag	Additional identifier included in output filenames
delayRange	Length of the delay axis
frequencyResolution	Number of frequency samples in the main trace
optimizationPoints	Number of points in optimization
bands	Sub-traces used in optimization
weights	Weights for the different sub-traces' FROG errors
optimize	Optimization on/off

1. INTRODUCTION

Ever since the first demonstration of mode-locking in the 1960's [1, 2], there has been a continuous race towards ever shorter pulses, reaching ever higher intensities, broader spectra and finer time resolutions. The logic is simple: the shorter the pulse, the shorter is the event it can be used to resolve. With the early progresses in generation of short pulses with techniques such as Q-switching [3] and mode-locking with their many variants [4,5], it became quickly more and more difficult to characterize the laser pulses with optoelectronic methods. Early optical characterization methods included the autocorrelator [6], only capable of roughly estimating the width of a pulse. When pulse durations were already reaching tens of femtoseconds [7], characterization methods finally took a giant leap when frequency-resolved optical gating (FROG) was developed in the early 1990's [8]. Not only was the technique able to temporally resolve the electric field amplitude of the pulse with unprecedented precision, but the phase could also be reconstructed, giving vital information about the chirp of the pulse, and the experienced group delay dispersion. As both amplitude and phase can be retrieved, FROG is a complete characterization method capable of defining the electric field profile and spectrum of a pulse. In-built self-consistency checks offer reliability and the possibility to detect flaws in the experimental setup just by inspecting the measured data. There are also limitations of FROG, of course. Depending on the particular variant, the temporal direction of the pulse might not be unambiguously defined, and physical constraints of the geometry might induce limitations to the accuracy of the measurement. FROG is also computationally intensive, making real-time acquisition of a pulse difficult.

Since the nineties, many FROG variants based on different nonlinearities have been introduced [9–11]. All these FROG variants rely on non-collinear interaction and avoid interference effects. Therefore, they are not applicable in a tight-focusing geometry. In 2006, a method was introduced that overcomes this problem and actually makes good use of the interferometric term, i.e., interferometric FROG or IFROG [12].

While FROG techniques are targeted at retrieving pulse shapes of laser pulses, it was recognized quickly that IFROG also holds an enormous potential as a spectroscopic

tool [13], enabling to temporally resolve the temporal response of nonlinear optical processes on a few-femtosecond time scale.

Recently, a novel variant of IFROG based on third-harmonic generation was introduced [14], denominated as the third-harmonic interferometric frequency-resolved optical gating (THIFROG). Indications are that this technique can resolve extremely fast processes down to the single-cycle response time. Resolving such processes is considered the holy grail in this 50 year race towards resolving ever faster processes in nature. Comparison THIFROG measurements using silica and titania revealed a marked difference that is indicative of a few-femtosecond response time of titania.

However, there is a major problem with THIFROG. Other than for any of all the above-cited FROG variants, there simply exists no description of a suitable retrieval software for this novel variant. In fact, THIFROG can be considered completely uncharted territory, despite of more than 20 years of publication on various FROG methods. The goal of this thesis is therefore to develop a suitable method that allows interpretation of measured THIFROG traces. This requires a thorough analysis of the structure of the THIFROG trace and a suitable approach for the retrieval software. Once the software was developed and tested, it was applied to measured THIFROG traces from the Max Born Institute in Berlin, Germany. The ultimate task was the interpretation of the apparent differences between the silica and the titania traces, ultimately yielding a 6.5 fs response time of the titania material. This response is among the fastest processes that was ever measured with spectroscopic methods in the visible/near-infrared spectral region.

In relation to this work, a conference paper was submitted for the XVIII$^{\text{th}}$ International Conference on Ultrafast Phenomena in Lausanne, Switzerland [15], and a poster presentation was given at the Physics Days 2012 in Joensuu, Finland [16].

2. FUNDAMENTALS OF OPTICS

In this chapter the theoretical framework for analysing and understanding the relevant phenomena behind creation, behaviour, and most importantly measurement and characterization of laser pulses are introduced. The first and second sections cover the basic theory for light and its propagation. Nonlinear light-matter interaction is discussed in the third section while the fourth and final section illuminates the properties and creation of ultrashort pulses.

2.1 Maxwell's Equations

The propagation of optical fields is governed by the Maxwell equations for electromagnetic phenomena, originally published in 1861 [17], here in their time-dependent, differential form and in SI units [18],

$$\nabla \cdot \tilde{\mathbf{D}} = \rho_f \tag{2.1a}$$

$$\nabla \cdot \tilde{\mathbf{B}} = 0 \tag{2.1b}$$

$$\nabla \times \tilde{\mathbf{E}} = -\frac{\partial \tilde{\mathbf{B}}}{\partial t} \tag{2.1c}$$

$$\nabla \times \tilde{\mathbf{H}} = \tilde{\mathbf{J}} + \frac{\partial \tilde{\mathbf{D}}}{\partial t}, \tag{2.1d}$$

where $\tilde{\mathbf{E}}$ and $\tilde{\mathbf{H}}$ are the electric and the magnetic field vectors, respectively. The sources for the electromagnetic field are represented by the total current density $\tilde{\mathbf{J}}$ and the free charge density ρ_f. Here the tilde symbol ($\tilde{}$) is used to stress the fact that the quantity at hand oscillates rapidly in time. The electric and the magnetic flux densities $\tilde{\mathbf{D}}$ and $\tilde{\mathbf{B}}$, respectively, arise as the corresponding $\tilde{\mathbf{E}}$ and $\tilde{\mathbf{H}}$ fields propagate inside a medium inducing a polarization $\tilde{\mathbf{P}}$ and a magnetization $\tilde{\mathbf{M}}$ of the material. This interaction is described by the *constitutive relations*,

$$\tilde{\mathbf{D}} = \epsilon_0 \tilde{\mathbf{E}} + \tilde{\mathbf{P}} \tag{2.2a}$$

$$\tilde{\mathbf{H}} = \frac{\tilde{\mathbf{B}}}{\mu_0} - \tilde{\mathbf{M}}\,, \tag{2.2b}$$

where ϵ_0 and μ_0 are the electric permittivity and the magnetic permeability, respectively, in vacuum. In the power series picture, polarization $\tilde{\mathbf{P}}$ has the general form [19]

$$\tilde{\mathbf{P}} = \epsilon_0 \sum_{n=1}^{\infty} \chi^{(n)} * \tilde{\mathbf{E}}^n, \tag{2.3}$$

where $\chi^{(n)}$ is the n^{th} order susceptibility, a tensor of the order $(n+1)$, and the $*$ symbol is used to denote convolution. Terms of the order $n \geq 2$ are referred to as nonlinear polarization, as their dependence on the electric field is nonlinear, while the term $n = 1$ is the linear polarization. The fundamental origin of nonlinear polarization lies in the anharmonic motion of bound electrons in response to intense optical fields.

The first of Maxwell's four equations is Gauss's law, Equation 2.1a, describing the relationship between electric field flux and their source, the electric charge [20]. Second is Gauss's law for magnetism, which states that the magnetic flux through any closed surface is exactly zero. This statement is equal to saying that unlike for electric fields, there are no sources or sinks of magnetic field, i.e., no magnetic charge (a magnetic monopole) exists. The third equation is Faraday's induction law, which says that a changing magnetic field will induce an electric field. The fourth and final equation is the Ampere's law, restated by Maxwell to include the electric displacement $\partial\tilde{\mathbf{D}}/\partial t$. This law states that a magnetic field can be created by current *and* by electric field changing in time. Likewise, an electric field is created by a time-dependent magnetic field. Maxwell's addition to the law was of special importance because it allows the existence of freely propagating electromagnetic waves. As will be shown in the following section, wave equations with the propagation speed $v = \sqrt{1/(\epsilon_0\,\mu_0)}$ can be derived from Maxwell's equations. Using the values for ϵ_0 and μ_0 already known at the time, Maxwell concluded this speed to be $v \approx 3 \cdot 10^8$ m/s. Previously, in 1851, Fizeau had experimentally determined the speed of light to be 315 000 km/s [21]. Such a remarkable agreement with experiment led Maxwell to the conclusion that light *is* electromagnetic radiation [22].

2.2 Propagation of Light

In the realm of optics, one is typically[1] interested in the solutions of Maxwell's equations in a nonmagnetic medium ($\tilde{\mathbf{M}} = 0 \Rightarrow \tilde{\mathbf{B}} = \mu_0\tilde{\mathbf{H}}$) with no free charges ($\rho_f = 0$) or free currents ($\tilde{\mathbf{J}} = 0$). The medium in which the light propagates

[1] These conditions are, however, not always valid. For example, three-photon absorption (3PA) causes electrons to rise from the valence band of a dielectric to the conduction band, so that there are free charges present and $\rho_f \neq 0$. This is likely to be the case with the titania measurements, presented in Section 4.2. For the sake of simplicity, the formulation here ignores this fact.

is, however, allowed to be nonlinear in nature as the relationship between $\tilde{\mathbf{D}}$ and $\tilde{\mathbf{E}}$ described by Equation 2.2a includes polarization $\tilde{\mathbf{P}}$, which in general exhibits a nonlinear dependency on $\tilde{\mathbf{E}}$. Dielectrics such as glass or air are examples of such a material. With these assumptions, a wave equation of the most general kind can be deduced by taking the curl of Equation 2.1c and using Equation 2.1d along with the constitutive relations 2.2a and 2.2b:

$$\nabla \times \nabla \times \tilde{\mathbf{E}} + \frac{1}{c_0^2}\frac{\partial^2 \tilde{\mathbf{E}}}{\partial t^2} = -\mu_0 \frac{\partial^2 \tilde{\mathbf{P}}}{\partial t^2}. \tag{2.4}$$

Here c_0 is the speed of light in vacuum, defined as

$$c_0 \equiv \sqrt{1/(\epsilon_0\, \mu_0)}\,. \tag{2.5}$$

This speed is equal to *exactly* 299,792,458 m/s, as the SI unit of metre is defined as the distance that light travels in vacuum in 1/299,792,458 seconds [23]. By using an identity from vector calculus, the first term on the left hand side of the wave equation 2.4 can be written as

$$\nabla \times \nabla \times \tilde{\mathbf{E}} \equiv \nabla\left(\nabla \cdot \tilde{\mathbf{E}}\right) - \nabla^2 \tilde{\mathbf{E}}. \tag{2.6}$$

The first term on the right hand side of the latter equation can be shown to be vanishingly small in most cases of interest and in particular when the *slowly-varying envelope approximation* (SVEA) is valid [19]. The criteria for validity of the SVEA is that the frequency bandwidth $\Delta\omega$, centered around the carrier frequency ω_0, of the optical field must be narrow, i.e., $\frac{\Delta\omega}{\omega_0} \ll 1$. In the time domain this approximation states that the envelope $\mathcal{A}$ of the electric field does not significantly change in the duration of an optical cycle of the carrier frequency:

$$\left|\frac{\partial \mathcal{A}}{\partial z}\right| \ll |k(\omega_0)\mathcal{A}| \quad \text{or} \quad \left|\frac{\partial \mathcal{A}}{\partial t}\right| \ll |\omega_0 \mathcal{A}|\,, \tag{2.7}$$

where k is the wave number. By ignoring higher-order derivatives on the grounds of these inequalities [18], SVEA allows the simplification of many wave equations. For Equation 2.6 this means that $\nabla\left(\nabla \cdot \tilde{\mathbf{E}}\right) \approx 0$. With this approximation we have $\nabla \times \nabla \times \tilde{\mathbf{E}} = -\nabla^2 \tilde{\mathbf{E}}$, so now the wave equation 2.4 becomes

$$\nabla^2 \tilde{\mathbf{E}} - \frac{1}{c_0^2}\frac{\partial^2 \tilde{\mathbf{E}}}{\partial t^2} = \mu_0 \frac{\partial^2 \tilde{\mathbf{P}}}{\partial t^2}\,. \tag{2.8}$$

This wave equation can be further refined by splitting the polarization into a linear

and a nonlinear part according to Equation 2.3,

$$\tilde{\mathbf{P}} = \tilde{\mathbf{P}}_{\mathrm{L}} + \tilde{\mathbf{P}}_{\mathrm{NL}} = \epsilon_0 \chi^{(1)} * \tilde{\mathbf{E}} + \tilde{\mathbf{P}}_{\mathrm{NL}} \,, \tag{2.9}$$

where $\tilde{\mathbf{P}}_{\mathrm{L}}$ depends linearly on the electric field $\tilde{\mathbf{E}}$. Insertion to Equation 2.8 gives

$$\nabla^2 \tilde{\mathbf{E}} - \frac{1}{c_0^2} \frac{\partial^2 \tilde{\mathbf{E}}}{\partial t^2} = \mu_0 \frac{\partial^2 \tilde{\mathbf{P}}_{\mathrm{L}}}{\partial t^2} + \mu_0 \frac{\partial^2 \tilde{\mathbf{P}}_{\mathrm{NL}}}{\partial t^2} \,. \tag{2.10}$$

In a similar manner, the electric displacement $\tilde{\mathbf{D}}$ can be broken in two,

$$\tilde{\mathbf{D}} = \tilde{\mathbf{D}}_{\mathrm{L}} + \tilde{\mathbf{P}}_{\mathrm{NL}} \,, \tag{2.11}$$

where the linear electric displacement is defined as

$$\tilde{\mathbf{D}}_{\mathrm{L}} = \epsilon_0 \tilde{\mathbf{E}} + \tilde{\mathbf{P}}_{\mathrm{L}} \,. \tag{2.12}$$

With Equation 2.12 and the identity 2.5, the second time derivatives of $\tilde{\mathbf{E}}$ and $\tilde{\mathbf{P}}_{\mathrm{L}}$ present in Equation 2.10 can be combined as

$$\frac{1}{c_0^2} \frac{\partial^2 \tilde{\mathbf{E}}}{\partial t^2} + \mu_0 \frac{\partial^2 \tilde{\mathbf{P}}_{\mathrm{L}}}{\partial t^2} = \mu_0 \frac{\partial^2 \tilde{\mathbf{D}}_{\mathrm{L}}}{\partial t^2} \,. \tag{2.13}$$

With these quantities, the wave equation 2.10 may now be written as

$$\nabla^2 \tilde{\mathbf{E}} - \mu_0 \frac{\partial^2 \tilde{\mathbf{D}}_{\mathrm{L}}}{\partial t^2} = \mu_0 \frac{\partial^2 \tilde{\mathbf{P}}_{\mathrm{NL}}}{\partial t^2}. \tag{2.14}$$

For illustrative purposes, a lossless, dispersionless, isotropic medium is considered. For such a medium it holds that the linear part of electric displacement $\tilde{\mathbf{D}}_{\mathrm{L}}$ is related to the electric field $\tilde{\mathbf{E}}$ by a dimensionless, scalar material constant $\epsilon^{(1)}$, and the vacuum permittivity ϵ_0,

$$\tilde{\mathbf{D}}_{\mathrm{L}} = \epsilon_0 \epsilon^{(1)} \tilde{\mathbf{E}} \,. \tag{2.15}$$

Insertion of this relation into the wave equation 2.14 yields

$$\nabla^2 \tilde{\mathbf{E}} - \epsilon_0 \epsilon^{(1)} \mu_0 \frac{\partial^2 \tilde{\mathbf{E}}}{\partial t^2} = \mu_0 \frac{\partial^2 \tilde{\mathbf{P}}_{\mathrm{NL}}}{\partial t^2} \,. \tag{2.16}$$

By using the identity 2.5 for the vacuum speed of light, Equation 2.16 becomes

$$\nabla^2 \tilde{\mathbf{E}} - \frac{\epsilon^{(1)}}{c_0^2} \frac{\partial^2 \tilde{\mathbf{E}}}{\partial t^2} = \frac{1}{\epsilon_0 c_0^2} \frac{\partial^2 \tilde{\mathbf{P}}_{\mathrm{NL}}}{\partial t^2} \,. \tag{2.17}$$

This is a wave equation, where the nonlinear polarization induced in the material acts as a source term for the electric field, i.e., the nonlinear response of a medium to an optical field generates new electromagnetic radiation. The Fourier transform of Equation 2.17 gives

$$-\nabla^2\mathbf{E} = \frac{n^2\omega^2}{c_0^2}\mathbf{E} + \frac{\omega^2}{\epsilon_0 c_0^2}\mathbf{P}_{\text{NL}}\,, \tag{2.18}$$

where ω is the angular frequency and n the index of refraction of the medium, satisfying the condition $n^2 = \epsilon^{(1)}$. Note that this is now a wave equation in the frequency domain, a description which can be useful on many occasions. Defining the vacuum wave number as $k_0 = \frac{\omega}{c_0}$ and the wave number in a medium as $k(\omega) = n(\omega)k_0$—a frequency dependent quantity in a dispersive medium—allows for the above wave equation to be written as

$$-\nabla^2\mathbf{E} = k^2(\omega)\mathbf{E} + \frac{k_0^2}{\epsilon_0}\mathbf{P}_{\text{NL}}\,. \tag{2.19}$$

Despite the assumptions made, Equation 2.19 is a highly general wave equation able to explain many an optical phenomena.

In the absence of nonlinear polarization, this equation has solutions of freely propagating waves with the velocity c_0/n. Setting $\mathbf{P}_{\text{NL}} = 0$ gives

$$\nabla^2\mathbf{E} + k^2(\omega)\mathbf{E} = 0\,, \tag{2.20}$$

which is in fact the ordinary *Helmholtz equation*.

2.3 Nonlinear Effects

Nonlinear optics, the branch of optics concerned in light-matter interaction of nonlinear nature, can be said to have begun with the first observation of second-harmonic generation (SHG) [24] shortly after the first laser was introduced in 1960 [25]. Nonlinear optical effects are typically observed where high optical intensities are involved, as with laser light, or with long propagation distances encountered in fiber-optic communication, where light is used to transmit information over transatlantic distances within optical fibers [26].

Nonlinear optical effects are categorized by the order of nonlinear polarization $\tilde{\mathbf{P}}_{\text{NL}}$ responsible for their existence. The magnitude of second-order, or $\chi^{(2)}$ effects such as second-harmonic generation and sum-frequency generation [27], for example, depend on the square of the electric field amplitude and the relevant elements of the $\chi^{(2)}$ tensor of the medium. $\chi^{(3)}$ nonlinearities are of special interest in this thesis, as

the lifetime of the nonlinear third-order polarization is studied in Chapter 6, and because third-harmonic generation (THG) is employed in the pulse characterization technique described in Chapter 4. The third-order nonlinear polarization $\tilde{\mathbf{P}}(t)^{(3)}$ is the term $n = 3$ of Equation 2.3 and can be written as [18]

$$\tilde{\mathbf{P}}(t)^{(3)} = \epsilon_0 \iiint_{-\infty}^{+\infty} \chi^{(3)}(t-t_1, t-t_2, t-t_3) \vdots \tilde{\mathbf{E}}(t_1)\tilde{\mathbf{E}}(t_2)\tilde{\mathbf{E}}(t_3)\, \mathrm{d}t_1 \mathrm{d}t_2 \mathrm{d}t_3 \; . \tag{2.21}$$

This representation of polarization in the time domain does not illuminate the origin of various nonlinear effects. A more comprehensive representation in the frequency domain can be given as [19]

$$\begin{aligned} \mathbf{P}_i^{(3)}(\omega_o + \omega_n + \omega_m) = \epsilon_0 D \sum_{ijkl} \chi_{ijkl}^{(3)}(\omega_o + \omega_n + \omega_m; \omega_o, \omega_n, \omega_m) \\ \times \mathbf{E}_j(\omega_o)\mathbf{E}_k(\omega_n)\mathbf{E}_l(\omega_m) \; , \end{aligned} \tag{2.22}$$

where the frequency dependent susceptibility tensor $\chi^{(3)}$ interacts with the three electric fields oscillating at arbitrary frequencies along any of the three Cartesian axes x, y, or z, producing polarization $\mathbf{P}_i^{(3)}$ along the axis specified by the index i. The essential part is that the third-order polarization produced by the three electric fields produces oscillates—and produces new light—at an angular frequency equal to the sum of ω_o, ω_n and ω_m. If all the frequencies are identical, $\omega_o = \omega_n = \omega_m \equiv \omega$, then light at 3ω is created, i.e., third-harmonic generation takes place. In the case where all three frequencies are unique, light on a fourth frequency is born, an effect referred to as four-wave mixing (FWM) [28]. As the electric fields are complex, negative angular frequencies are allowed as well. Setting the three frequencies equal in magnitude but making one negative while leaving the other two positive yields a sum of $\omega + \omega - \omega = \omega$. This describes the optical Kerr effect [29], where the refractive index experienced by an optical field is affected by the field's own intensity. The intensity dependent refractive index n obeys the relation [19]

$$n = n_0 + n_2 \, I \; , \tag{2.23}$$

where I is the optical intensity of the incident field, n_0 is the linear refractive index and n_2 is the second-order nonlinear refractive index, related to the third-order susceptibility $\chi^{(3)}$ by

$$n_2 = \frac{3}{4\, n_0^2\, \epsilon_0\, c} \cdot \chi^{(3)} \; . \tag{2.24}$$

The Kerr effect is responsible for the $\chi^{(3)}$ effects of self-phase modulation (SPM) [30] and self-focusing [31], a phenomena facilitating the generation of ultrashort pulses through Kerr-lens mode-locking (KLM) [32].

2.4 Laser Pulses

An introduction to the physics of laser pulses is given here, closely following the theory section of the excellent thesis of Christian Grebing [33].

Laser pulses, which are essentially compact packets of electromagnetic radiation, are possible solutions to the wave equations presented earlier in this chapter. A complete representation of a laser pulse is given by its electric field as a function of time and space, $\tilde{\mathbf{E}}(t, \mathbf{r})$, or equivalently by its frequency domain counterpart, $\mathbf{E}(\omega, \mathbf{r})$. The description of linearly polarized light can be reduced to a scalar form, $E(t, z)$, with light propagating along the z-direction. Elliptically polarized light can then be described by considering its individual components separately. The electric field of laser pulse can also be seen as a superposition of plane waves of different frequencies with a fixed phase relationship,

$$\tilde{E}(t, z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} \mathcal{E}(\omega, z) e^{-i\omega t} \mathrm{d}\omega \,, \tag{2.25}$$

where $\mathcal{E}(\omega, z)$ is the complex spectral amplitude of the wave with the frequency ω. Calligraphic symbols are used here to denote complex quantities. Equation 2.25 is in fact the Fourier transform of the electric field in the frequency domain, i.e., $\mathscr{F}\{E(\omega, z)\} = \tilde{E}(t, z)$. Since the electric field $\tilde{E}(t, z)$ is a physical quantity and therefore real valued, the properties of the Fourier transform compel the complex spectral amplitude $\mathcal{E}(\omega, z)$ to be self-adjoint, i.e., $\mathcal{E}(\omega, z) = \mathcal{E}^*(-\omega, z)$. In other words, all the information of the physical field is contained in either side of the spectrum! The complex field amplitude $\tilde{\mathcal{E}}^+(t, z)$ is obtained by integrating over positive frequencies

$$\tilde{\mathcal{E}}^+(t, z) = \frac{1}{\sqrt{2\pi}} \int_{0}^{+\infty} \mathcal{E}(\omega, z) e^{-i\omega t} \mathrm{d}\omega, \tag{2.26}$$

so that the physical electric field of the pulse can be written as $\tilde{E}(t, z) = 2\Re\left[\tilde{\mathcal{E}}^+(t, z)\right]$. The one-sided spectrum can be defined as

$$\mathcal{E}^+(\omega, z) = \begin{cases} \mathcal{E} & , \omega \geq 0 \\ 0 & , \omega < 0 \,, \end{cases} \tag{2.27}$$

so that it holds that

$$\tilde{\mathcal{E}}^+(t, z) = \mathscr{F}\left\{\mathcal{E}^+(\omega, z)\right\}. \tag{2.28}$$

In most practical cases, and also in the context of this thesis, the spectral amplitude

is centered around the carrier frequency ω_0 with contributions only within a frequency interval $\Delta\omega$, which is small compared to ω_0. By substituting $\omega \to \omega_0 + \Delta\omega$ and using the frequency shift property of the Fourier transform for Equation 2.26, it is possible to introduce the carrier frequency ω_0 also in the time domain

$$\begin{aligned}\tilde{\mathcal{E}}^{+}(t,z) &= e^{-i\omega_0 t} \times \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} \mathcal{A}(\Delta\omega, z)\, e^{i\Delta\omega t}\, \mathrm{d}(\Delta\omega) \\ &= \mathcal{A}(t,z)\, e^{-i\omega_0 t} \\ &= A(t,z)\, e^{i\varphi(t,z)}\, e^{-i\omega_0 t}\ . \end{aligned} \tag{2.29}$$

Here $\mathcal{A}(t,z)$ is the inverse Fourier transform of the spectrally shifted complex amplitude $\mathcal{A}(\Delta\omega, z) = \tilde{\mathcal{E}}^{+}(\omega_0 + \Delta\omega, z)$. The quantities $\varphi(t,z)$ and $A(t,z)$ describe the temporal phase and the envelope of the electric field, respectively. This relation is especially important for the analysis of pulse characterization techniques, and in the framework of the pulse retrieval software discussed in latter chapters. For future reference, a simplified notation outside the scope of this section is adopted,

$$\tilde{\mathcal{E}}(t) = E(t)\, e^{-i\omega_0 t} = A(t)\, e^{-i(\,\omega_0 t + \varphi(t)\,)}\ . \tag{2.30}$$

Here $\tilde{\mathcal{E}}(t)$ is the complex electric field, and $E(t)$ and $A(t)$ are the complex and real electric field amplitudes, respectively, the former containing the complex phase $e^{i\varphi(t)}$. The spatial dependence has been omitted along with the use of calligraphic symbols for complex quantities, as both $\tilde{\mathcal{E}}(t), E(t) \in \mathbb{C}$ while $A(t) \in \mathbb{R}$.

By using the slowly-varying envelope approximation, the inequalities in Equation 2.7 allow the wave equation 2.10 to be written in the form of a simplified description for the spatial evolution of a pulse,

$$\frac{\partial}{\partial z}\mathcal{A}(\Delta\omega, z) + i\Delta k\, \mathcal{A}(\Delta\omega, z) = 0\ . \tag{2.31}$$

The use of SVEA to simplify equations describing pulse evolution, most notably the nonlinear Schrödinger equation (NLS) [18], makes both the analytical studying and the numerical simulation of pulse propagation and the phenomena involved possible. In the context of fiber-mode propagation, NLS can be used to describe SPM, FWM, SHG and various other nonlinear effects.

SVEA is not explicitly used in the calculations presented in this thesis, as the studied ultrashort pulses are *not* narrowbanded as demanded by the approximation, in fact, they have an extremely wide frequency bandwidth. This is a fact which can be explained by a fundamental property of the Fourier transform: the more pronounced a

temporal (spatial) feature is, the more frequencies (spatial-frequencies) are required to describe it. That is, to generate the shortest pulse, one must generate the broadest spectrum [34]. A related quantity is the time-bandwidth product (TBP), which sets a lower limit to the pulse duration a certain spectrum can support. If the pulse is considered to be as short as TBP allows it to be, the pulse is said to be *transform limited*. The exact value of TBP depends on the shape of the pulse envelope, e.g., a Gaussian pulse has a minimum TBP of ≈ 0.441 while a bandwidth-limited hyperbolic secant squared (sech^2) pulse has a TBP of ≈ 0.315 [35]. These values hold when both temporal and spectral width are measured using the full width at half maximum (FWHM) criteria. The shortest pulses must also have a simple shaped intensity envelope, like sech^2, with preferably no satellite structures and a flat phase, so that there is no frequency *chirp* present [34]. Chirp is the time dependence of the instantaneous frequency of a pulse, defined as the negative first time derivative of the phase $\varphi(t)$ [18]

$$\delta\omega(T) \equiv -\frac{\partial\varphi}{\partial T} , \tag{2.32}$$

where T is the time coordinate relative to the frame moving at the speed of the pulse envelope, the *group velocity* $v_g \equiv \partial\omega/\partial k$.

Various schemes to generate ultrashort laser pulses have been developed, and generally the fastest pulses have been created through *mode-locking* [36]. Mode-locked lasers can be divided in to active and passive types according to their operation principle. Active types rely on the periodic modulation of the losses experienced by optical fields circulating in the laser cavity, or on changing the phase of the round-trip. As active modulation relies on electronics, the inherent slowness of electronic drivers limits the possibilities to create pulses less than a picosecond in length. Passive mode-locking uses light itself to create the suitable cavity conditions. A slight fluctuation in intensity can cause more losses to some modes than others so that eventually a single (or a group of) mode prevails and consumes virtually all the available optical energy for itself. This can be achieved for example with saturable absorbers such as a semiconductor saturable absorber mirror (SESAM) for bulk lasers [37], or with erbium doped fiber lasers [38], also sometimes in conjunction with SESAMs [39]. The most prominent technology is the solid state titanium-sapphire (Ti:sapphire) laser employing passive Kerr-lens mode-locking, capable of producing pulses sub-6 fs in the near-infrared region, corresponding to less than two optical cycles in length, at MHz repetition rates, i.e., the number of pulses emitted per second is in the millions [40, 41]. In this context the term ultrafast is simply too *slow*, as it can be used in context of picosecond pulses as well. Instead, the attribute *few-cycle* is used to describe pulses approaching the fundamental limit of a single optical cycle. By using the infrared pulses created by a Ti:sapphire laser

to generate high-harmonics, attosecond (10^{-18} s) pulses in the extreme ultraviolet (XUV) wavelength range have been demonstrated [42, 43]. Recently, the possibility to create laser pulses in the staggeringly short time scale of zeptosecond (10^{-21} s), i.e., less than an attosecond in length, has been discussed [44].

In the following chapter, the focus is moved from creating ultrashort pulses to measuring them. Relevant techniques and related theory are discussed.

3. PULSE CHARACTERIZATION

Ultrashort laser pulses are an essential tool in the exploration of a myriad of elementary processes in nature [41]. For instance, photosynthesis, vision and protein folding in biology, molecular vibrations and re-orientations in chemistry, and electron-hole relaxation in physics are all events that occur on femtosecond time scale [34]. Time resolution of these events demands for the ability to create and measure light pulses of even shorter duration. As a bare minimum requirement in ultrafast measurements, the length of the pulse must be known in order to determine the temporal resolution of an event. In many experiments precise knowledge of the amplitude and phase profiles of the pulse is of extreme importance, as they can have a significant effect on the outcome [45]. For example, much greater molecular photodissociation is observed for a chirped pulse in comparison to an unchirped one [46]. Many material characterization techniques rely on measuring the effect of the material on ultrashort pulses [47–50].

The generation of ultrashort laser pulses requires deep knowledge of the underlying physical phenomena, the verification of which is accomplished through precise measurements. Furthermore, pulse-distorting effects must be quantified in order to make further reduction of the pulse duration possible. Today, laser pulses of sub-10 fs duration corresponding to only a few optical cycles can routinely be produced with mode-locked solid state lasers [51]. Shaped femtosecond pulses, having numerous applications, e.g., in biomedical imaging [52] and enhancing of laser-electron interaction in generation of plasma waves [53], must naturally be verified to be of the correct form by pulse characterization. Precisely characterized femtosecond pulses are the foundation of optical frequency metrology [54], and used extensively for time resolving chemical reactions [55].

As laser pulses are among the shortest events created by man, no other even shorter event exists that could be used to measure them. Instead, a pulse can be characterized by using the pulse to measure *itself*. In order to fully characterize a pulse, its electric field in time ($E(t)$) or equivivalently in frequency domain ($E(\omega)$) must be measured, i.e, both the intensity and the phase of the pulse must be measured completely in either domain [45]. In this chapter different methods for pulse char-

acterization are discussed starting from historical predecessors of *frequency-resolved optical gating* (FROG) and progressing towards a novel technique dubbed as *third-harmonic interferometric frequency-resolved optical gating* (THIFROG), which will be the subject of chapter 4.

3.1 Optical Autocorrelation

The fastest optoelectronic sensor [51], the streak camera, has a time resolution of roughly one picosecond, whereas the length of a few-cycle pulse is on the order of femtoseconds. Since even the fastest measurement devices fall three orders of magnitude behind in time resolution, the direct measurement of the intensity of ultrashort pulses as a function of time is impossible.

The autocorrelator, employed in all early pulse characterization methods, surpasses this limitation by using the pulse to measure itself. Specifically, the pulse is split into two, one of which is delayed by an adjustable delay τ with respect to the other. The two pulses then meet in a medium with an instantaneous nonlinear response, resulting in a delay-dependent temporal overlap of the two pulses, and eventually, a measurable signal. The nonlinear medium can be, for example, a second-harmonic generation crystal [45]. The intensity of the generated frequency-doubled light, proportional to the product of the intensities of the two input pulses, is measured by a photodetector as a function of the pulse delay. The peak of the signal is located at zero delay, where there is maximum temporal overlap. At large delays, there is no temporal overlap and the generated SHG signal is significantly smaller, if not zero, as is the case with the so called background-free autocorrelation signal,

$$\mathrm{AC}(\tau) \propto \int_{-\infty}^{+\infty} I(t)\, I(t-\tau)\, \mathrm{d}t \;, \tag{3.1}$$

achieved with a noncollinear intensity autocorrelator setup, schematically illustrated in Figure 3.1. Here, the two pulses cross each other at a finite angle. Clearly, information of the pulse duration is obtained with an autocorrelation measurement.

The use of a noncollinear setup can, however, result in geometrical smearing of the beam and eventually in too large of an estimate for the pulse duration [56, 57]. Therefore, for short pulses, a collinear setup avoiding this effect is preferred, yielding a signal [51]

$$\mathrm{IAC}(\tau) \propto \int_{-\infty}^{+\infty} \left| \left[E(t) + E(t-\tau)\right]^2 \right|^2 \mathrm{d}t \;. \tag{3.2}$$

In collinear setups the measured signal is composed of several different components,

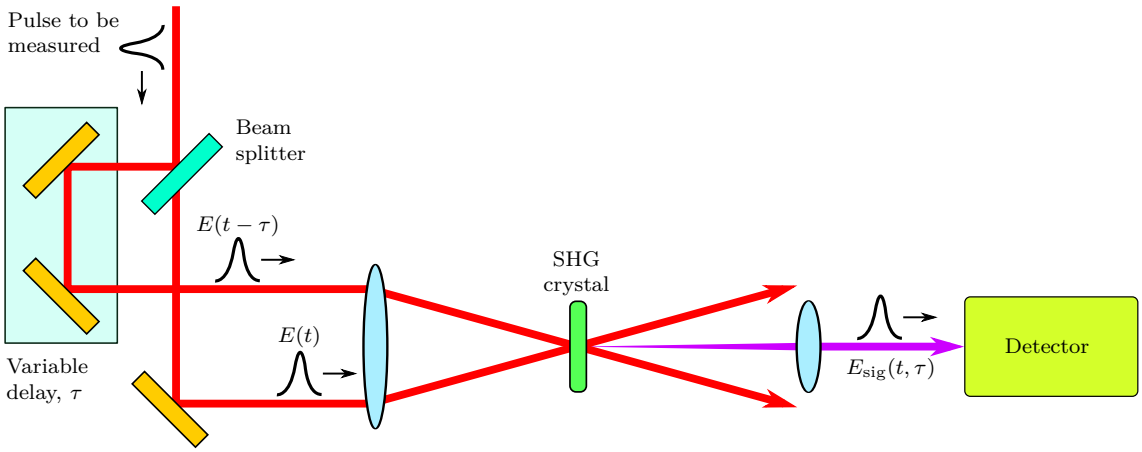


Figure 3.1: Experimental layout for a background-free intensity autocorrelation (AC) measurement. The pulse to be measured is split in two, one of which is variably delayed. The two pulses meet in a second harmonic generation crystal producing a frequency-doubled field, whose intensity is measured as a function of the delay τ [34].

resulting from the expansion of the above and other similar expressions. These components interfere with each other, hence the title interferometric intensity autocorrelation (IAC). The difference of collinear and noncollinear experimental setups is illustrated in Figure 3.2.

As an additional benefit, the degree of interferometric modulation in an IAC measurement provides additional information about the chirp of a pulse. One disadvantage of a collinear setup is the stronger presence of unwanted residual fundamental light [58]. An example of an interferometric autocorrelation measurement of a few-cycle pulse is presented in Figure 3.3.

Unfortunately, autocorrelation does not allow for the unambigious retrieval of the structure of a pulse without additional knowledge [51]. Assumptions of the pulse shape can be made, e.g., on the basis of a theoretical description of the mode-locking process in order to estimate the pulse width. This, however, is not possible in the sub-10 fs regime, where the complex structures of ultrashort pulses prohibit the use of simple analytical functions in pulse reconstruction. Moreover, the use of overly optimistic pulse shapes often results in a too short of an estimate for the pulse duration [45]. Ambiguity is a serious problem as different pulse shapes of equal power spectrum may result in nearly identical autocorrelation traces [60].

Decorrelation methods have been developed to retrieve the pulse shape from an autocorrelation measurement with the help of additional experimental information, such as the simultaneous measurement of the power spectrum of the laser [51]. Computer optimization strategies can then be used to find a pulse shape that satisfies both constraints given by the two measurements. Reliable operation, however, requires

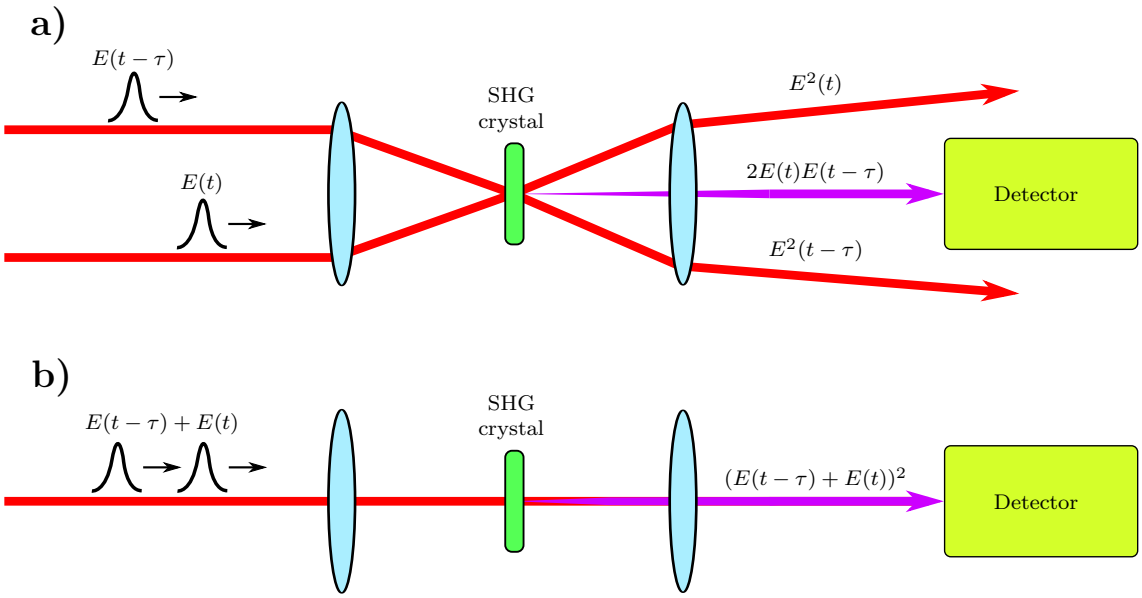


Figure 3.2: SHG signal from (a) noncollinear (AC) and (b) collinear (IAC) measurement setups. Collinear configuration produces a more complex signal, which provides additional means for detection of measurement errors. Figure adapted from Amat-Roldán *et al.* [59].

data with excellent signal-to-noise ratio, and even then ambiguities are not fully removed [60].

The phase of the pulse can be retrieved, e.g., with the *Gerchberg–Saxton algorithm* (GSA) [61] developed in the seventies. The GSA involves an iterative error-reduction procedure where the objective is to find a phase profile for the pulse matching the measurement data in both, the temporal (intensity) and the spectral domain. Fourier transforms back and forth between the two domains are made, resulting in convergence from an initial guess for the phase toward the (usually) unique solution [34]. The use of GSA, however, requires that both the spectrum and the intensity profile of the pulse are known, which not is the case here as autocorrelation does not unambiguously determine the temporal intensity.

The iterative method of *temporal information via intensity* (TIVI), described by Rundquist and Peatross in 1998 [62], can be used to estimate the intensity profile of a pulse through autocorrelation without making spesific assumptions of the pulse shape. By using TIVI and simultaneously measuring the power spectrum of the laser, phase retrieval becomes possible with GSA. Unfortunately, yet again, the outcome suffers from ambiguities [34] arising from both TIVI and GSA, which prohibits the use of this method in retrieval of complex structures of ultrashort pulses.

Another iterative method is the *phase and intensity from correlation and spectrum only* (PICASO) [63]. This technique requires that the spectrum of the pulse is mea-

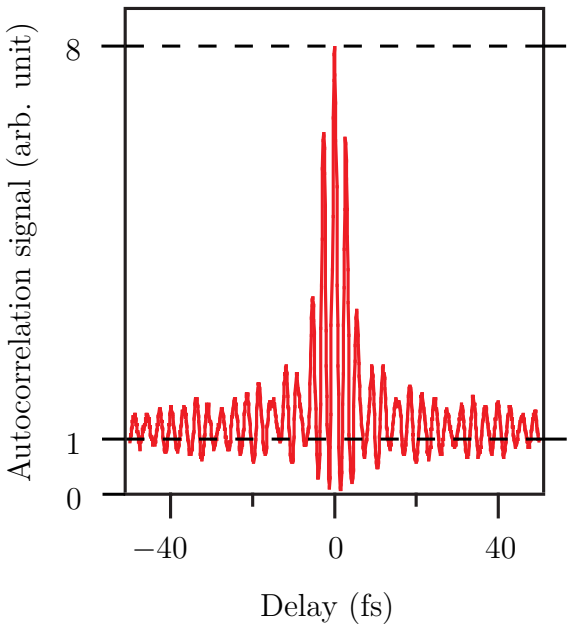


Figure 3.3: Interferometric intensity autocorrelation (IAC) measurement of a 6 fs pulse [51]. The characteristic 1 : 8 ratio between the background and the peak intensities is observed.

sured along with one or more interferometric autocorrelations. Baltuška *et al.* [41] have also described a similar technique. Because of the underlying ambiguities in IAC measurements accompanied with a separate spectrum measurement [60], ultimately neither of these methods can be considered to be a reliable method for pulse retrieval.

3.2 Complete Characterization Methods

To overcome the aforementioned limitations, several techniques have been developed, of which perhaps the most widespread [58] are *frequency-resolved optical gating* (FROG) [34, 45] and *spectral phase interferometry for direct electric-field reconstruction* (SPIDER) [64]. The two methods allow for the reconstruction of both the amplitude and the phase profiles of ultrashort pulses and have their respective advantages and disadvantages [12].

As an interferometric method, SPIDER is more suited to measure the spectral phase of a pulse, advantageous in tracking the influence of dispersion in a short pulse, whereas FROG is better able to resolve the exact satellite structure of a pulse [12]. SPIDER is generally faster, offering acquisition and reconstruction rates of several tens of hertz, and can be used as an online method to aid in aligning femtosecond systems. FROG, on the other hand, uses a time consuming optimization strategy for pulse retrieval, which makes it slower but also more robust. The built-in consistency

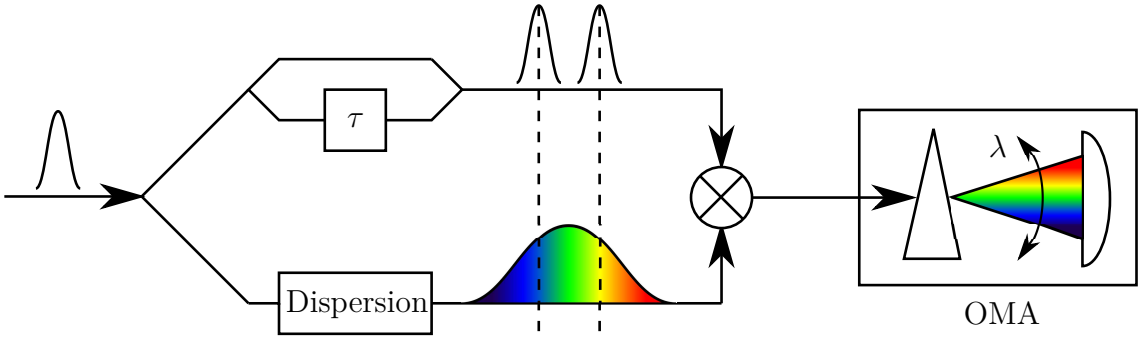


Figure 3.4: Schematic picture of a SPIDER experimental setup. The input pulse on the left is initially duplicated so that one pulse proceeds to the upper path and the other to the lower path. In the upper path, the pulse is again duplicated, and one of the replica pulses is delayed in respect to the other. The pulse propagating in the lower path is heavily chirped, e.g., with a highly dispersive glass block, so that red frequencies are shifted to the front of the pulse while blue components are shifted to the back. Sum-frequency generation takes place where the two paths cross, and the chirped pulse causes a frequency-shift in the two replica pulses from the upper path. The frequency shift is different for the two pulses separated by the delay τ because of the chirp in the third pulse. The spectral interferogram of the up-converted replicas is measured by the optical multichannel analyzer (OMA), which samples the relative phase delay as a function of frequency. This information can be used to reconstruct the spectral phase $\varphi(\omega)$, and when combined with an independent amplitude spectrum measurement, the pulse is fully characterized [51].

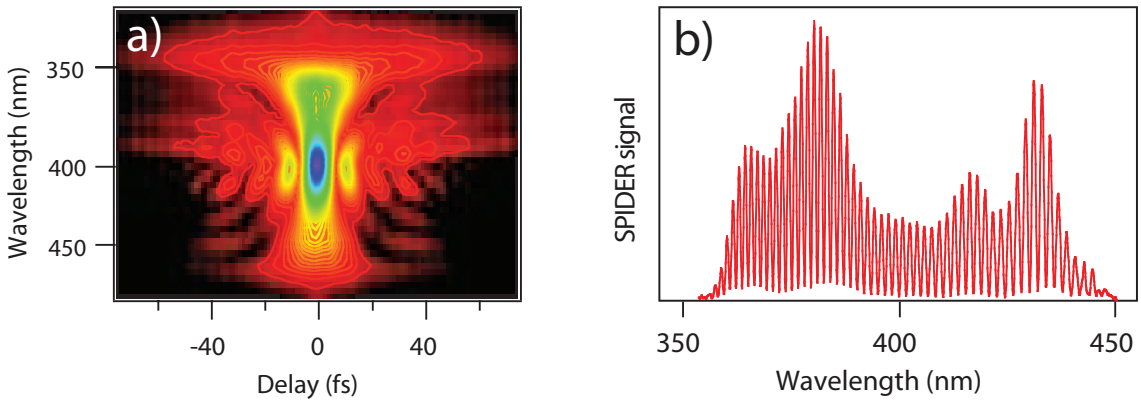


Figure 3.5: Data from (a) FROG and (b) SPIDER measurements of 6 fs pulses nearly identical to the one that was used in the IAC measurement presented earlier in Figure 3.3 [51].

checks of FROG allow for detection of experimental flaws such as spectral filtering. Apart from a few exceptions, a FROG reconstruction is effectively unambiguous [51]. Because of the iterative nature of the FROG algorithm, however, it possible that pulse retrieval fails to converge. This is especially likely when complex pulse shapes are involved, or when there is a high amount of noise in the measurement data, e.g., when weak pulses are measured. Example data sets produced by FROG and SPIDER are illustrated in Figure 3.5.

The selection of a pulse characterization method should always be based on the requirements and constraints of a specific measurement. Since a special variant of FROG is being investigated in this thesis, we move to discuss FROG in greater detail while leaving the SPIDER technique to lesser attention.

3.3 Frequency-Resolved Optical Gating

Introduced in 1993 by R. Trebino and D. J. Kane [8,65], the FROG method involves the measurement of a spectrogram for a pulse,

$$S(\omega,\tau) \propto \left| \int_{-\infty}^{+\infty} E(t)\, g(t-\tau)\, \exp(-i\omega t)\, \mathrm{d}t \right|^2 , \tag{3.3}$$

where $g(t-\tau)$ is some gate function with a variable delay [45]. In FROG, an optical nonlinearity is used to perform the gating. A *trace* of a FROG measurement consists of spectrally resolved autocorrelations for each of the delays τ, i.e., a two-dimensional array sampled at a (τ,ω) grid is produced [51]. The trace is often plotted as a rectangular image with colors representing the relative magnitude of data points, as was illustrated in Figure 3.5a.

Many variants of the FROG technique exist. One of them, employing the second-harmonic generation nonlinearity, is called the SHFROG and yields a trace of the form

$$I_{\mathrm{SHFROG}}(\omega,\tau) \propto \left| \int_{-\infty}^{+\infty} E(t)\, E(t-\tau)\, \exp(-i\omega t)\, \mathrm{d}t \right|^2 . \tag{3.4}$$

The use of different autocorrelation geometries and nonlinearities for the gating result in FROG traces of slightly different forms. Other variants include the self-diffraction FROG [8] and polarization gating FROG [65]. A very general schematic picture of a FROG setup is presented in Figure 3.6. For implementations of different FROG variants, see Figures 3.8, 3.10, and 4.1.

Excluding an (irrelevant) absolute phase factor, it is possible to completely deter-

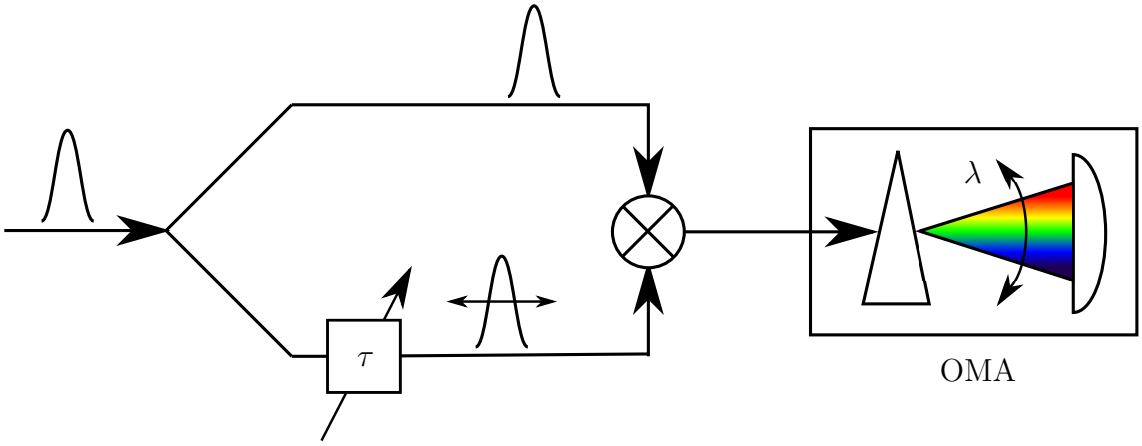


Figure 3.6: Schematic picture of a FROG experimental setup. Similarly to Figure 3.4 with SPIDER, the input pulse on the left is duplicated so that one pulse proceeds to the upper path and the other to the lower path. The replica pulse propagating on the lower path is subjected to a variable delay τ, exactly as with the autocorrelator in Figure 3.1. A nonlinear interaction whose intensity depends on τ, e.g., SHG, occurs when the two pulses meet, and the result is recorded with a spectrometer (OMA) as a function of delay. An optimization strategy can then be employed to completely characterize the pulse using this information [51].

mine $E(t)$ from its spectrogram with the aid of inversion algorithms [66]. This, however, requires knowledge of the gate function, which is, of course, unknown since the pulse is being used to gate itself. For this reason, the problem is restated as the *two-dimensional phase-retrieval problem.*

3.3.1 Trebino's Approach and the Generalized Projections Algorithm

One formulation of the problem, described by Trebino *et al.* [45], involves the use of a *signal field* $E_{\text{sig}}(t,\tau)$. For an SHG autocorrelator, the signal field is $E(t)\,E(t-\tau)$. The signal field is defined as the Fourier transform of a new quantity $\hat{E}_{\text{sig}}(t,\Omega)$ with respect to the *delay-frequency* Ω, which is the conjugate variable of the time delay τ. The object is then to find $\hat{E}_{\text{sig}}(t,\Omega)$ that will yield the original pulse field $E(t) = \hat{E}_{\text{sig}}(t,\Omega=0)$.

Now the SHFROG trace 3.4 is rewritten in terms of $\hat{E}_{\text{sig}}(t,\Omega)$ as

$$I_{\text{SHFROG}}(\omega,\tau) \propto \left| \iint_{-\infty}^{+\infty} \hat{E}_{\text{sig}}(t,\Omega)\, \exp(-i\omega t - i\Omega\tau)\, \mathrm{d}t\, \mathrm{d}\Omega \right|^2 . \tag{3.5}$$

This expression is clearly a two-dimensional Fourier transform of $\hat{E}_{\text{sig}}(t,\Omega)$. One also finds that the spectrogram measurement yields information solely of the magnitude

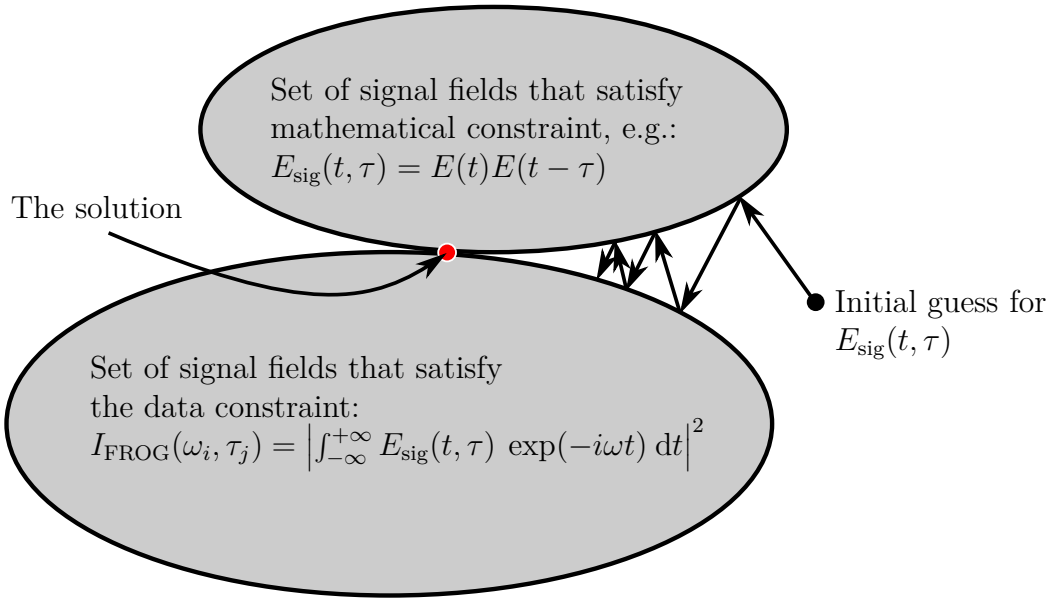


Figure 3.7: Geometrical interpretation for the generalized projections algorithm, which finds a solution to a problem iteratively by projecting between the sets of constraints. Here, the two sets are convex, resulting in convergence, but for non-convex sets, as in FROG, convergence is not guaranteed. Figure adapted from Trebino *et al.* [45] and DeLong *et al.* [68].

of the signal field, leaving its phase unknown. The 2D phase-retrieval problem is then to find the phase of the Fourier transform of $\hat{E}_{\mathrm{sig}}(t,\Omega)$, i.e., the phase of the signal field.

Provided that the mathematical form of $\hat{E}_{\mathrm{sig}}(t,\Omega)$ is known, as is the case with SHFROG, the problem can be solved with the *generalized projections algorithm* (GPA) [67]. The principle of the GPA method is graphically illustrated in Figure 3.7. The signal field, that is to be found, must satisfy two constraints. The first one is the data constraint, i.e., the squared magnitude of the 1D Fourier transform of the signal field must be equal to the measured trace,

$$I_{\mathrm{FROG}}(\omega_i,\tau_j) = \left|\int_{-\infty}^{+\infty} E_{\mathrm{sig}}(t,\tau)\ \exp(-i\omega t)\ \mathrm{d}t\right|^2 . \tag{3.6}$$

The second constraint is the mathematical form of the signal field in terms of the pulse field $E(t)$. As previously stated, this depends on the exploited nonlinearity of the measurement. One must then search for a signal field that satisfies both constraints by making *projections* from one set of signal fields satisfying one of the constraints unto another set of signal fields satisfying the other constraint and vice versa. Provided that the both sets are convex, convergence towards a solution satisfying both constraints is guaranteed.

Unfortunately, this is not the case for FROG and hence convergence is uncertain. Furthermore, not every FROG trace can be expressed in terms of a signal field E_{sig}, as is the case with the *interferometric FROG* (IFROG) methods, which will be discussed in greater detail later on. The projection step of the FROG algorithm is also difficult to implement in IFROG, rendering this approach useless in such cases [58].

3.3.2 Stibenz and Steinmeyer's Approach

A more general algorithm suitable for a wider class of spectrotemporal distributions is described in a 2006 paper by Stibenz & Steinmeyer [58]. This pulse retrieval algorithm will be referred to as the *Stibenz–Steinmeyer algorithm* (SSA) in the context of this thesis. Based on a gradient method often used for local optimization strategies, the SSA can also be seen as a modification of the traditional GPA approach of Trebino *et al.* suitable for regular FROG traces.

The goal of the SSA is to iteratively minimize the functional distance

$$Z = \sum_{i,j=1}^{N} \mid I_{\text{FROG}}^{\text{meas}}(\Delta\omega_i, \tau_j) - I_{\text{FROG}}^{(\text{n})}(\Delta\omega_i, \tau_j) \mid^2 \ , \tag{3.7}$$

where $I_{\text{FROG}}^{\text{meas}}(\Delta\omega_i, \tau_j)$ is the measured FROG trace and $I_{\text{FROG}}^{(\text{n})}(\Delta\omega_i, \tau_j)$ is the trace calculated for the nth iteration of the electric field $E^{(n)}(t_k)$. Such a goal is also present in the common FROG algorithm with the distinction that in the SSA the traces are evaluated in frequency-delay domain (ω, τ) rather than in time-delay domain (t, τ). The next iteration for the electric field, $E^{(n+1)}(t_k)$, is found by line minimization along the gradient of Z. A factor μ, which minimizes Z in the next iteration must be found according to

$$\tilde{\mathbf{E}}^{(n+1)} = \mathbf{E}^{(n)} + \mu \mathbf{g} \ , \tag{3.8}$$

where the $\mathbf{g}$ is the gradient of Z and defined via

$$g_k = \left(\frac{\partial Z}{\partial \Re \left[E(t_k) \right]} \right) + i \left(\frac{\partial Z}{\partial \Im \left[E(t_k) \right]} \right) \ . \tag{3.9}$$

3.3.3 FROG Error

The quality of a pulse reconstruction is assessed by calculating the *FROG error*, G, of a FROG trace [34, 45]. Specifically, it is the rms difference between the measured

trace I_{FROG}, with the peak value normalized to unity, and the k-th iteration of the trace, $I^{(k)}_{\mathrm{FROG}}$, computed from the reconstructed pulse,

$$G = \sqrt{\frac{1}{N^2}\sum_{i,j=1}^{N}\left| I_{\mathrm{FROG}}(\omega_i,\tau_j) - \alpha I^{(k)}_{\mathrm{FROG}}(\omega_i,\tau_j)\right|^2}\,, \tag{3.10}$$

where α is the real number that minimizes G. Typically, a FROG error of $<1\%$ is achieved in what is consider to be a successful reconstruction of experimental data. It should be noted that while FROG errors of different FROG variants are not directly comparable, they still offer a general image of the relative quality of a pulse reconstruction and act as a measure of experimental error for measured FROG traces [58].

The normalization constant α is calculated for each of the iteration steps k. Since the sum in the expression for FROG error in 3.10 is clearly positive, the square root and the $1/N^2$ factor can be left out when searching for α. Here the indices i, j and arguments for FROG intensities have been omitted for the sake of clarity.

$$\begin{aligned}
\arg\min_{\alpha} G &= \arg\min_{\alpha} \sum\left| I_{\mathrm{FROG}} - \alpha I^{(k)}_{\mathrm{FROG}}\right|^2 \\
&= \arg\min_{\alpha} \sum\left(I_{\mathrm{FROG}} - \alpha I^{(k)}_{\mathrm{FROG}}\right)^2 \\
&= \arg\min_{\alpha} \sum\left[I^2_{\mathrm{FROG}} - 2\alpha I_{\mathrm{FROG}} I^{(k)}_{\mathrm{FROG}} + \alpha^2\left(I^{(k)}_{\mathrm{FROG}}\right)^2\right] \\
&= \arg\min_{\alpha} \sum I^2_{\mathrm{FROG}} - 2\alpha\sum I_{\mathrm{FROG}} I^{(k)}_{\mathrm{FROG}} + \alpha^2\sum\left(I^{(k)}_{\mathrm{FROG}}\right)^2 \\
&= \arg\min_{\alpha} \alpha^2\sum\left(I^{(k)}_{\mathrm{FROG}}\right)^2 - 2\alpha\sum I_{\mathrm{FROG}} I^{(k)}_{\mathrm{FROG}}\,.
\end{aligned} \tag{3.11}$$

Differentiating the function to be minimized from equation 3.11 with respect to α yields

$$2\alpha\sum\left(I^{(k)}_{\mathrm{FROG}}\right)^2 - 2\sum I_{\mathrm{FROG}}\, I^{(k)}_{\mathrm{FROG}} = 0\,, \tag{3.12}$$

so that the equation for α is seen to be

$$\alpha = \frac{\sum_{i,j=1}^{N} I_{\mathrm{FROG}}(\omega_i,\tau_j)\; I^{(k)}_{\mathrm{FROG}}(\omega_i,\tau_j)}{\sum_{i,j=1}^{N}\left(I^{(k)}_{\mathrm{FROG}}(\omega_i,\tau_j)\right)^2}\,. \tag{3.13}$$

3.4 Interferometric FROG

While conventional FROG and SPIDER techniques have proven to be excellent tools in the characterization of pulses in the range of 10–100 fs, the extremely short pulses of sub-10 fs duration require particular care because of the enormous bandwidths

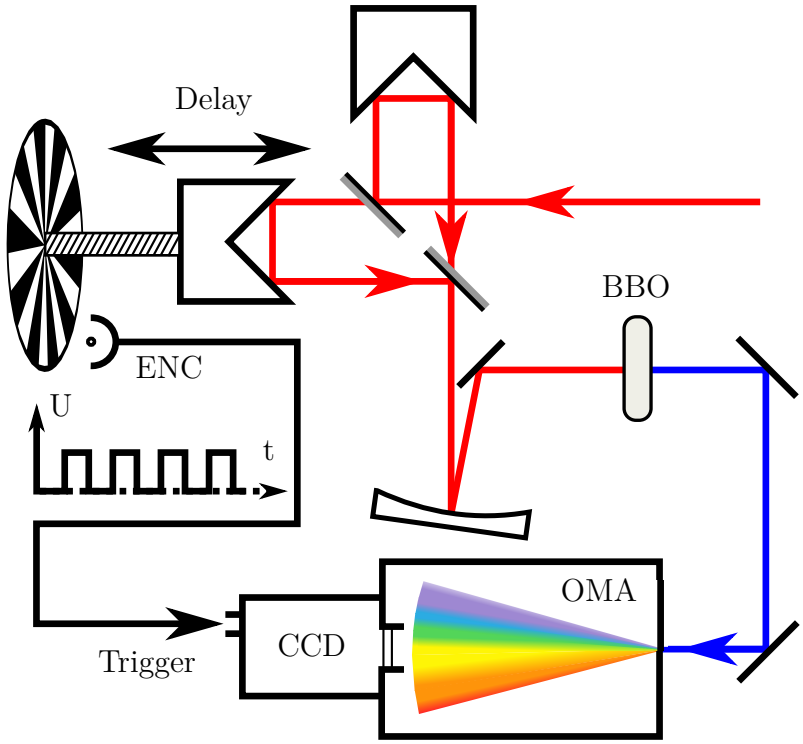


Figure 3.8: IFROG setup used by Stibenz and Steinmeyer in their 2005 paper [12]. The input beam is split in two by the interferometer while a constantly moving stage induces a variable delay on one of the interferometer arms. The two beams join together when leaving the interferometer, propagating collinearily to a BBO crystal, where second-harmonic generation takes place. The frequency doubled beam is guided to an optical multichannel analyzer (OMA), which essentially divides the input spectrum to several channels, which in turn are recorded by a fast line-scan CCD camera. The camera is triggered to take shots at constant delay steps of 225 attoseconds, i.e., 1/12 of the wavelength for 800 nm. This corresponds to roughly thrice the Nyquist limit of the second-harmonic field. Figure adapted from Stibenz and Steinmeyer [12].

involved and the complex structures of the pulses [12, 58]. Pulse shapes can become more complex, e.g., due to leading or trailing satellites produced in pulse compression by supercontinuum generation. This, in turn, leads to complicated spectral structures, which can pose a problem for both FROG and SPIDER. FROG has an edge over SPIDER when dealing with complex pulse shapes, as sampling on a large two-dimensional grid offers some redundancy. In FROG, the frequently employed noncollinear geometry can distort the measured pulse profile as a finite crossing angle causes a dependance of the zero delay position on the lateral dimension [41]. In certain cases this can lead to an overestimation of the pulse duration by some 5–10%. This can be compensated for by the use of special algorithms or by aperturing of the central part of the beam. Another alternative is to use collinear geometry, which avoids the problem in the first place. As an example, the IFROG measurement setup used by Stibenz and Steinmeyer in [12] is illustrated in Figure 3.8.

Collinear geometry can be implemented in FROG by replacing the background-free autocorrelation with interferometric autocorrelation. This variant of FROG, referred to as interferometric FROG (IFROG), was widely avoided because of the overwhelming amount of data needed. This leads to slow convergence during reconstruction and possibly even difficulties in meaningful interpretation of the measurement. A trace of second-harmonic IFROG (SHIFROG) measurement, employing SHG as the nonlinearity, contains modulation at both, the fundamental and the second-harmonic periods. The Nyquist rate f_N, which is the minimum sampling rate required to measure a signal containing frequency components less or equal to a frequency $f_{\max}$ without aliasing [69], is defined as

$$f_N = 2\, f_{\max} \,. \tag{3.14}$$

Thus a pulse with a carrier frequency f_0 has a maximum frequency component of $f_{\max} = 2f_0$ after SHG, setting the Nyquist rate at $4f_0 = 1/4T_0$, i.e., the IFROG measurement must be executed with a delay step $\Delta\tau$ of less than a quarter of an optical cycle. For a 800 nm pulse this translates to a delay step size of less than 0.67 femtoseconds, so for a delay range of only $\pm$100 fs, 300 spectra must be measured. The measurement must also be carried out quickly in order to avoid interferometer drift, or alternatively the interferometer must be actively stabilized, adding to the complexity of the experiment.

The SHIFROG trace can be written as [59]:

$$I_{\text{SHIFROG}}(\omega,\tau) \propto \left| \int_{-\infty}^{+\infty} \left(\mathcal{E}(t) + \mathcal{E}(t-\tau)\right)^2 \exp(-i\omega t)\, \mathrm{d}t \right|^2 . \tag{3.15}$$

Recalling that the complex electric field $\mathcal{E}(t)$ can be written as a product of the complex amplitude $E(t)$ and modulation term at the carrier angular-frequency ω_0

$$\mathcal{E}(t) = E(t)\, e^{i\omega_0 t} \,,$$

Equation 3.15 may now be expanded. Fourier analysis of the resulting equation

leads to the following form [58], where the substitution $\Delta\omega = \omega - 2\omega_0$ is used,

$$I_{\text{SHIFROG}}(\omega,\tau) \propto \left|E_{\text{SH}}(\Delta\omega)\right|^2 \tag{3.16a}$$

$$+ \cos\left[(2\omega_0 + \Delta\omega)\,\tau\right] \left|E_{\text{SH}}(\Delta\omega)\right|^2 \tag{3.16b}$$

$$+ \; 4\cos\left[\left(\omega_0 + \frac{\Delta\omega}{2}\right)\tau\right] \times$$

$$\Re\left[E_{\text{SHFROG}}(\Delta\omega,\tau)\; E^*_{\text{SH}}(\Delta\omega)\exp\left(i\frac{\Delta\omega}{2}\tau\right)\right] \tag{3.16c}$$

$$+ \; 2\left|E_{\text{SHFROG}}(\Delta\omega,\tau)\right|^2 \; . \tag{3.16d}$$

Here $E_{\text{SH}}(\Delta\omega)$ is the second-harmonic field of a pulse

$$E_{\text{SH}}(\Delta\omega) \equiv \int_{-\infty}^{+\infty} E^2(t)\exp(-i\Delta\omega t)\,\mathrm{d}t \; , \tag{3.17}$$

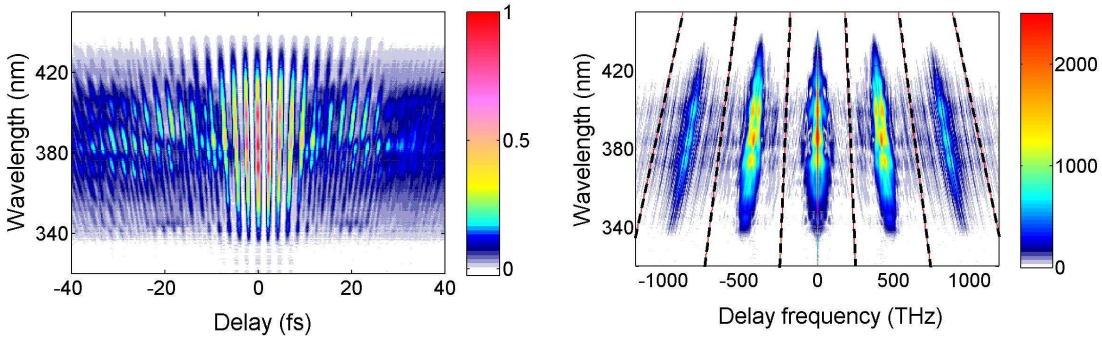
and $E_{\text{SHFROG}}(\Delta\omega,\tau)$ is the SHFROG field.

$$E_{\text{SHFROG}}(\Delta\omega,\tau) \equiv \int_{-\infty}^{+\infty} E(t)E(t-\tau)\exp(-i\Delta\omega t)\,\mathrm{d}t \; . \tag{3.18}$$

The first term 3.16a describes the interaction of the pulse with itself and is responsible for the background signal in IAC measurements employing the SHG nonlinearity [59]. The second term 3.16b is the cross term of the two interfering SHG pulses, containing the information already found in the first term, but it is modulated by 2ω in the delay-frequencies. However, unlike the first term, the second term cannot be used for pulse retrieval. The interaction of the frequency-doubled $E(t)E(t-\tau)$ field, the only field measured in AC, with the SHG fields of the two individual pulses is expressed by the third term 3.16c modulated at the fundamental frequency ω_0. The fourth and final term 3.16d gives the ordinary SHFROG signal of Equation 3.4, as $I_{\text{SHFROG}} = |E_{\text{SHFROG}}|^2$. The different modulational bands are clearly visible in Figure 3.9b, which is a Fourier transform of an IFROG trace measured by Stibenz and Steinmeyer with the setup illustrated in Figure 3.8.

3.4.1 Retrieval from the IFROG Trace

By recognizing that IFROG is essentially a spectrally resolved IAC, integration of the trace with respect to frequency recovers the autocorrelation similar to that depicted in Figure 3.3. Thus, the integrity of the measured IFROG trace can be easily verified during acquisition by confirming that the IAC displays the characteristic 1:8 ratio between the background level and the zero delay peak. Measurement of the fringes



(a) An interferometric FROG trace

(b) FT of (a) along the delay axis

Figure 3.9: (a) On the left, an interferometric FROG trace measured by Stibenz and Steinmeyer with the setup that was illustrated in Figure 3.8. (b) On the right, a Fourier transform of (a) along the delay axis reveals the four different modulational sidebands and the DC baseband (zero delay-frequency component) of the IFROG trace at equal distances from each other. The different bands reside on delay frequencies equal to multiples of the carrier frequency ω_0, with the baseband centered on zero delay frequency. The two closest sidebands on both sides of the baseband correspond to fundamental modulation at $\pm\omega_0$, while the two outer sidebands at $\pm 2\omega_0$ correspond to the second-harmonic component [12].

in IAC allows for self calibration of the delay axis, further improving the means to detect experimental errors during the IFROG measurement.

The different modulational terms (3.16b, 3.16c) and the unmodulated DC part, i.e, the zero delay-frequency component, (3.16a, 3.16d) can be extracted from the IFROG trace by means of Fourier filtering. In this procedure the trace is Fourier transformed with respect to the delay so that the the different components become clearly visible in the delay-frequency domain. The desired components can then be identified by their location on the delay-frequency axis and separated by setting all the other components, along with either one of the two mirror bands of the relevant modulational component found on equal distances from the origo in both positive and negative frequencies, as zero and subsequently taking the inverse Fourier transform back to the delay-time domain.

The unwrapping procedure described by Amat-Roldán *et al.* [59] used two dimensional Fourier transform instead, reducing the error introduced by the *fast Fourier transform* (FFT) algorithm because of assuming periodicity in the applied direction. The DC part was separated from the trace, and the term 3.16a was removed by subtracting the background level of the DC part, which can be either measured or averaged over several samples at the very edges of the delay axis, where the SHFROG signal is negligible. This leaves only the noncollinear SHFROG trace 3.16d, for which one can use standard retrieval algorithms to reconstruct the pulse.

While Amat-Roldán *et al.* described how to acquire a standard SHFROG trace

from IFROG measurements, Stibenz and Steinmeyer [12] went a step further by recognizing that the different modulational components provide an independent way to reconstruct the pulse, and that the information they provide can be used as an additional cross check, especially valuable in the measurement of few-cycle pulses. Here the pulse reconstruction relies on the retrieval of the phase $\varphi_{\mathrm{mod}}(\omega,\tau)$ of the cosine in 3.16b, i.e., the phase of the SH sidebands. This is only possible if the fringe structure of the sidebands is intact, which is not always the case. The retrieval of the noncollinear SHFROG trace is, however, still possible even if the fringes are smeared.

The phase $\varphi_{\mathrm{mod}}(\omega,\tau) \equiv (2\omega_0 + \Delta\omega)\tau$ can be extracted from the Fourier filtered SH sideband, arising from 3.16b, by retrieving its phase. One can also use this modulated SH signal to independently reconstruct the delay τ by identifying the marker fringes of the SH sidebands in the Fourier domain. Once acquired, $\varphi_{\mathrm{mod}}(\omega,\tau)$ is used to extract the fundamental modulation term 3.16c from the IFROG trace by erasing all but the fundamental sidebands in the delay-frequency domain, taking an inverse Fourier transform back to the delay-time domain and multiplying the result with $\cos(\varphi_{\mathrm{mod}}/2)$. This multiplication eliminates the cosine in 3.16c and leaves only the $\Re[]$ expression, dubbed as the fundamental modulation FROG (FM-FROG) trace by Stibenz and Steinmeyer, which can also be expressed in the form

$$I_{\text{FM-FROG}} \propto |E_{\text{SHFROG}}(\Delta\omega,\tau)|\,|E_{\text{SHFROG}}(\Delta\omega,\tau=0)| \times \cos\left(\varphi_{\text{SHFROG}}(\Delta\omega,\tau) - \varphi_{\text{SHFROG}}(\Delta\omega,\tau=0) + \frac{\Delta\omega}{2}\tau\right). \tag{3.19}$$

Here $E_{\text{SHFROG}}(\Delta\omega,\tau=0) = E_{\text{SH}}(\Delta\omega)$ and $\varphi_{\text{SHFROG}}(\Delta\omega,\tau)$ and $\varphi_{\text{SHFROG}}(\Delta\omega,\tau=0)$ are the phases of the SHFROG and the SH fields, respectively.

The correct sign of the FM-FROG trace, vital to a meaningful interpretation, is ensured by this phase-sensitive technique. As the real part of the product of two complex numbers can be negative, the FM-FROG trace can in fact exhibit negative portions. The cosine term connects the sign of the trace to its phase. For a transform-limited Gaussian pulse the FM-FROG is almost entirely positive. In the presence of chirp the phase $\varphi_{\mathrm{mod}}(\omega,\tau)$ is noticeably altered, and strong oscillations in the FM-FROG can arise. Furthermore, satellite pulses can give rise to significant negative portions in the trace. With the FM-FROG retrieved from the IFROG trace, the modified GPA described in Section 3.3.2 can now be used to reconstruct the pulse. Standard FROG strategies are not applicable here, as the projection step present in most of these algorithms requires the knowledge of a signal field, which is difficult to define for the trace in question.

3.5 FROG for Nonlinearity Lifetime Measurement

The applications for FROG are not limited to pulse characterization. In their Nano Letters paper of 2010 [13], Anderson *et al.* describe the use of IFROG to measure the plasmon dephasing time of a metallic nanostructure.

Electron dephasing, which defines the initial interplay of an optical field with the metal, is one of the fastest processes in metals with a time scale of but a few femtoseconds. It also governs the temporal evolution of surface plasmon polaritons (SPPs)—coherent charge density oscillations of the quasi-free conduction band electrons of the metal—which may both propagate or be localized. As the electron dephasing time T_2 is proportional to the local field enhancement of metallic nanoparticles, it plays an important role in the functionality of plasmonic nanostructures and optical antennas. Understanding of the ultrafast dynamics involved provides valuable insight to theoretical models of electron behaviour in metals.

The extremely short time scale of the phenomena demands for an optical measurement scheme involving ultrashort pulses. Operation in these extreme circumstances makes signal analysis difficult, and because of this, previous attempts had relied on several assumptions such as a transform-limited spectral phase of the driving optical pulse, resulting in inaccuracies of the measured response. The use of FROG, however, allows for the complete characterization of the optical response function of a single metal nanostructure without specific assumptions of the resonance effect involved.

Anderson *et al.* made two separate second-harmonic IFROG measurements, where 9.5 fs pulses with a center wavelength of 780 nm from a Ti:sapphire oscillator were focused to a 20 μm spot and subjected first to a 50 μm thick BBO crystal, then a gold nanotip with a tip radius of roughly 10–20 nm. The latter was chosen as its behaviour is of great interest in many applications, such as tip-enhanced and tip-scattering scanning near-field optical microscopy. The two IFROG traces were analyzed with the Fourier filtering strategy for the DC band, described in Section 3.4.1, after which commercial FROG pulse retrieval software was used to reconstruct the electric field $E(t)$ of the original pulse from the BBO measurement, and the plasmon dephasing induced optical polarization $P(t)$ from the gold nanotip measurement. By assuming that the SHG response from the BBO crystal is *nearly* instantaneous, $P(t)$ can be described by the convolution of $E(t)$ with the response function $R(t)$ of the gold nanotip

$$P(t) = \int_{-\infty}^{+\infty} R(t-\tau)E(t)\,\mathrm{d}\tau\ . \tag{3.20}$$

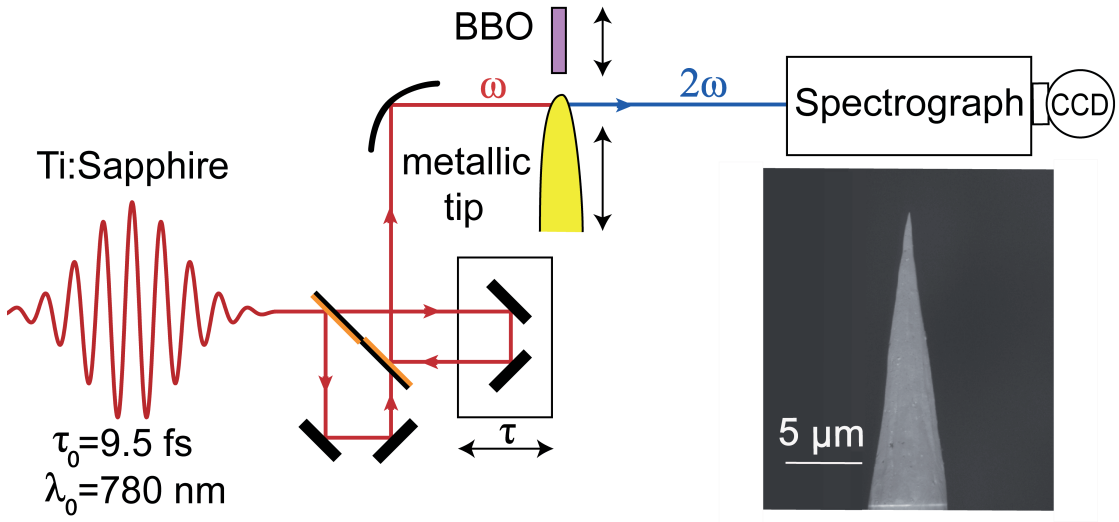


Figure 3.10: Test setup used by Anderson *et al.* [13]. An IFROG trace of 9.5 fs laser pulses from a Ti:sapphire oscillator is recorded after the pulses have interacted with one of the samples. The BBO and metallic tip samples are movable, and used one at a time. An image of the gold nanotip produced with a scanning electron microscope measurement is presented in the lower right corner.

The response function may then be determined by applying standard deconvolution strategies to the reconstructed fields $E(t)$ and $P(t)$.

The application of the procedure described above to the measurement data yielded a response function with a shape bearing striking resemblance to that of an damped harmonic oscillator

$$R_{\text{sim}}(t) = A \exp(i\omega_{\text{pl}}t - \gamma t/2) \,, \tag{3.21}$$

where A is the effective oscillator strength, ω_{pl} the plasmon resonance frequency, and $\gamma t/2$ the line width. Numerical fit of the damped harmonic oscillator model to the measured response function gave a spectral line width of $\Gamma = \hbar\gamma = 65$ meV, finally defining the measured plasmon dephasing time of the gold nanotip as $T_2 = 2\hbar/\Gamma = 20 \pm 5$ fs. The experimental error of ± 5 fs was based on the FROG errors of the two IFROG traces. A schematic description of the measurement setup used by Anderson *et al.* is shown in Figure 3.10.

A similar strategy can in principle be applied not only for plasmon dynamics, but also for determining the lifetime of any optical response in a medium, provided that the laser pulses used are sufficiently short, i.e., roughly in the same time scale as the nonlinearity. By using a deconvolution approach, it is possible to gain information of complex processes occurring on time scales even faster than the duration of the laser pulse. In principle the time resolution is limited only by the signal-to-noise

ratio of the experiment. The second-harmonic IFROG can be replaced with another FROG technique, which can, e.g., remove the direction-of-time ambiguity of the reconstructed pulses. In the following chapter an IFROG technique based on third-harmonic generation is explored, and in Chapter 6 it is harnessed for estimating the lifetime of an ultrafast $\chi^{(3)}$ nonlinearity in a dielectric medium.

4. THIRD-HARMONIC INTERFEROMETRIC FROG

Although the FROG geometries based on SHG are the most sensitive and frequently employed [70], they have a number of weaknesses. Firstly, very thin, fragile, and expensive noncentrosymmetric crystals must be used in order to assure phase-matching for the entire spectrum of the pulse. Secondly, they have a limited capability to measure asymmetric pulses because of the direction-of-time ambiguity.

In contrast, third-order FROG techniques do not suffer from this ambiguity [45, 71]. Instead, their use has been limited by the lack of materials with sufficiently strong third-order nonlinearities such as the third-harmonic generation (THG). The weak THG signals generated in materials such as fused silica require that sensitive detection systems and high energy pulses are used.

For infrared laser wavelengths, surface third-harmonic generation (STHG) from an air-glass interface has been a standard for third-order characterization of femtosecond pulses with third-harmonic FROG and related techniques [72, 73]. Its many advantages include an independence of wavelength and bandwidth, relatively high THG conversion efficiency and strongly reduced phase-matching requirements.

Recently, both organic [70] and inorganic [14] thin films have been found to possess a strong third-order nonlinearity, enabling the characterization of unamplified, low energy pulses. Because of the high nonlinearity, extremely thin material layers may be used, essentially removing all phase-matching constraints normally present in nonlinear optics. This enables the conversion of broadband spectra without spectral filtering and temporal broadening through dispersion, making FROG geometries based on third-order processes particularly suited for the measurement of few-cycle optical pulses. Such short pulses are especially vulnerable to the aforementioned effects, which are capable of corrupting the shape and duration of the measured pulse. In addition to being immune to the direction-of-time ambiguity and the capability of pulse asymmetry detection, it is also possible to make a distinction between pre- and post-pulses with techniques exploiting the third-order autocorrelation [74], such

as FROG.

While previous examples of FROG techniques based on third-order nonlinear effects used noncollinear geometries, there is no fundamental reason prohibiting the collinear approach. And so it came to pass that the palette of available FROG variants was recently augmented with a novel technique employing third-harmonic generation as the nonlinearity in interferometric FROG.

4.1 The Technique

The first demonstration of the third-harmonic interferometric FROG (THIFROG) technique was given by Das *et al.* in 2011 [14], who used it to characterize 15 fs pulses from a Ti:sapphire oscillator. A commercial pulse retrieval software for THG FROG was used to analyze the unmodulated DC kernel of the THIFROG trace, leaving the modulational sidebands unused in the absence of a software capable of reconstructing a third-order interferometric trace. Such a software will be presented in this thesis, later on in Chapter 5.

One objective of the paper was to determine the optimal thickness for a TiO_2 thin film with the goal of producing a strong, wide band THG signal. Through third-order autocorrelation measurements, it was found that the optimal layer width of 180 nm was capable of producing a THG signal up to 26 times stronger than that of a surface third-harmonic generation (STHG) from a air-glass interface, which was also studied for comparison. This experimentally determined thickness is close to the theoretically predicted value of 160 nm, which is the coherence length of the material in question, given by

$$L_C = \frac{\lambda}{6\,(n_3 - n_1)} \tag{4.1}$$

for the THG process, approximately confirming the result. The values used here are $n_1 = 2.23$ for the refractive index of TiO_2 at the fundamental wavelength of 800 nm and $n_3 = 3.06$ for that of the third-harmonic field at 266 nm, calculated using a formula for the refractive index in the interband region as a function of energy [75] with the experimental coefficients for the formula measured by Kim [76]. Typically, nonlinear optical processes exhibit the highest frequency conversion efficiency with layers as thick as the coherence length. By comparing the THG signals from a thin film and an air–glass interface, the third-order nonlinearity of the thin film can be quantified [77]. This method yielded an absolute value of $\chi^{(3)} = 9.9 \cdot 10^{-19}$ $\mathrm{m^2V^{-2}}$ for the TiO_2 thin film nonlinearity, exceeding the bulk TiO_2 nonlinearity of $\chi^{(3)} = 4.8 \cdot 10^{-21}$ $\mathrm{m^2V^{-2}}$ for 1064 nm by two orders of magnitude. Although the latter value

was defined for a different wavelength, the authors argued that the corresponding value for 800 nm is close to this. The high nonlinearity was attributed to surface effects, as surface enhancement of third-order nonlinearity have previously been reported in other materials [78, 79]. The above values for $\chi^{(3)}$ were converted from their reported values in the gaussian system to SI units with the relation [19]

$$\chi^{(3)}(\mathrm{m^2\,V^{-2}}) = \frac{4\,\pi}{(3\cdot 10^4)^2}\ \cdot\ \chi^{(3)}(\mathrm{esu})\ . \tag{4.2}$$

The experiments corroborated the fact that titania thin films are a well suited medium for obtaining a strong THG signal over the entire bandwidth of a Ti:sapphire oscillator, and when combined with THIFROG, they offer a viable alternative in the characterization of ultrashort pulses. In comparison to most of the previously known third-order pulse characterization techniques, THIFROG measurements from easily fabricated titania thin films offer a significantly enhanced THG efficiency with a relatively simple experimental setup and little restrictions on the suitable wavelength region. A schematic description of the measurement setup used by Das *et al.* is illustrated in Figure 4.1.

4.2 THIFROG Measurements for Silica and Titania Thin Films

In this thesis a pair of THIFROG measurements for silica (SiO_2) and titania (TiO_2) thin films from a similar setup to that described in the previous section is being investigated. The measurements were conducted in the Max Born Institute for Nonlinear Optics and Short Pulse Spectroscopy of Berlin, Germany by Das and coworkers in fall 2011. The measured traces are presented in Figure 4.2. In the next chapter, these measurements will be analyzed by performing simulations with a custom made program, with the goal of reconstructing the measured THIFROG traces and the pulses that conceived them. The originally identical pulses interact differently with the two samples, and may expand in time during the nonlinear interaction with the material. The reconstructed pulses are subsequently used to estimate the lifetime of the third-order optical nonlinearity associated with the titania thin film sample, as the difference in the durations of the frequency-tripled pulses for the two samples serves as a measure for the lifetime of the nonlinear response.

This is walking on uncharted territory, as an entire THIFROG trace (Das *et al.* only used the baseband for conventional THG pulse retrieval) has never been used for a pulse retrieval—before now. This also constitutes the world's first measurement of the astonishingly short lifetime of a $\chi^{(3)}$ nonlinearity in a dielectric.

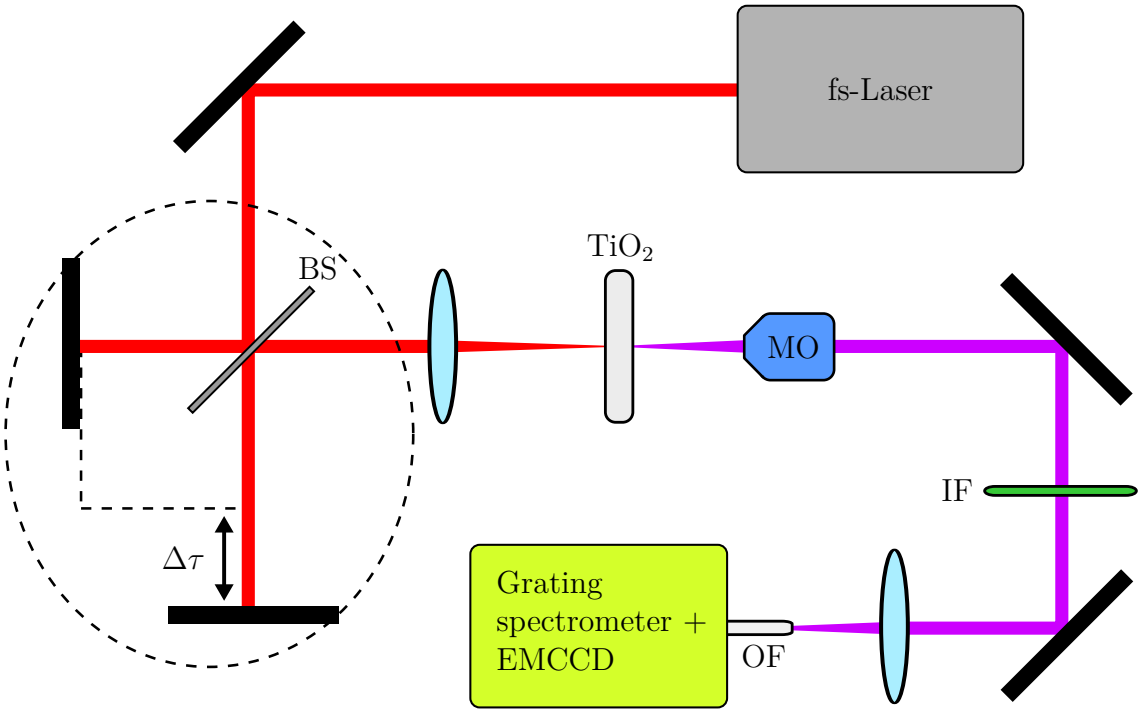


Figure 4.1: Experimental setup used by Das *et al.* for the detection of the THG signal generated in a titania thin film, and for characterization of the 15 fs pulses with the energy of 4 nJ and central wavelength of 800 nm produced by a Femtolasers Femtosource Scientific Ti:sapphire oscillator with a 75.3 MHz repetition rate. An autocorrelation is produced with the aid of the Michelson interferometer, composed of two mirrors and a beam splitter (BS) circled on the left. The third-harmonic field is collected by a UV transmitting microscope objective (MO) and separated from the residual fundamental field by interference filters (IF). The beam is guided by an optical fiber (OF) into a grating spectrometer and captured by an electron multiplying charge-coupled device (EMCCD) providing gain for the measurement. Figure adapted from [14].

While the experimental setup used here follows the layout of Figure 4.1 closely, the Ti:sapphire oscillator has been upgraded since the 2011 paper [14]. The new device is a PULSE : ONE BASIC laser system by VENTEON [80], capable of delivering sub-8 fs, possibly ≈ 6 fs transform limited pulses of 2.5 nJ energy at 80 MHz repetition rate. Sample spectrum and an intensity envelope provided by the manufacturer are presented in Figure 4.3 for reference. The hyperbolic secant squared, or sech^2, intensity envelope of the pulse in 4.3b is used as a model for the seed pulse in the pulse retrieval software discussed in Chapter 5.

The two samples were prepared in Lazer Zentrum Hannover e.V. of Germany by Dr. Marco Jupé and Prof. Dr. Detlev Ristau with a rugate filter manufacturing system, where ion-beam sputtering is used to deposit layers of tunable width and composition of source materials onto a substrate [81]. For these samples, a single 180 nm layer of pure titania and silica were deposited onto their own 1 mm thick glass substrate of the brand "Suprasil 1 mm standard".

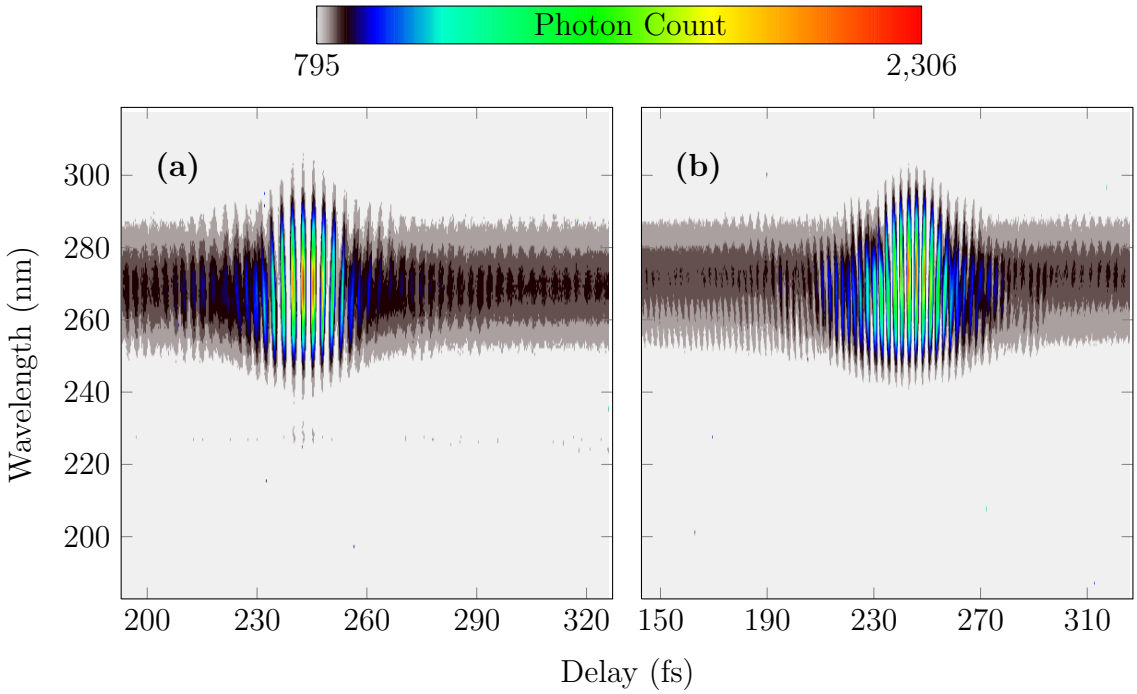
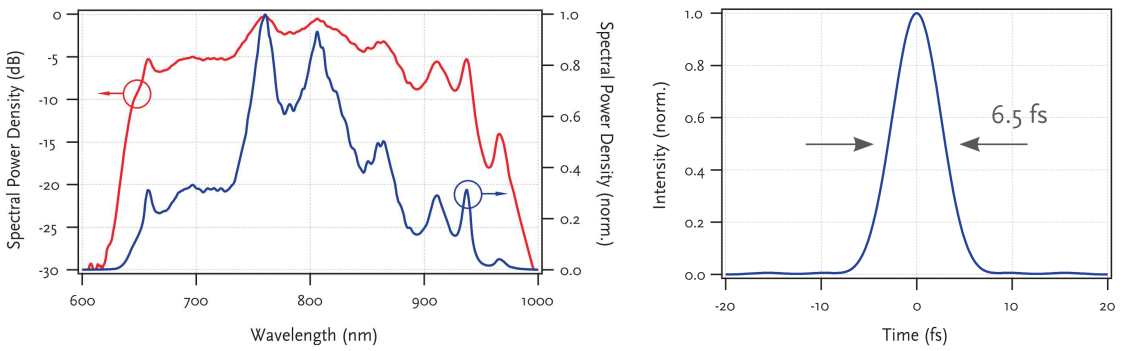


Figure 4.2: Experimental data from THIFROG measurements of pulses which have traversed (a) silica and (b) titania thin film samples. The bulk of the intensity distribution with photon count of > 900 is the actual THIFROG signal, which is found near the 266 nm wavelength of the third-harmonic field for the pulse centered at roughly 800 nm. The black trails on either side of the intensity maxima in delay is the background signal, caused by a single pulse which does not overlap in time with the second pulse. It should be noted that the zero-point for delay is arbitrary in these measurements, so the fact that the maxima of the traces are located in different delay coordinates is irrelevant.



(a) Spectrum (b) Intensity envelope

Figure 4.3: Sample spectrum (a) and intensity envelope (b) for a pulse produced by the VENTEON | PULSE : ONE BASIC laser system employed in the THIFROG measurements for silica and titania thin films. Screen captures from an advertisement [80].

Because the optical properties and exact crystal structure of these thin films have not been measured, literary sources are used to estimate the essential quantities. There are three common polymorphs of titania found in nature: rutile, brookite and anatase [82]. In addition to these, five synthetic crystal structures have been discovered. Optical and physical properties of titania thin films are known to vary widely even for samples prepared with nominally the same deposition techniques [83]. Measurements for the third-order nonlinear susceptibility of both anatase and rutile titania thin films have yielded absolute values of $\chi^{(3)} = 9.9 \cdot 10^{-19}$ m^{2}V^{-2} and $\chi^{(3)} = 5.9 \cdot 10^{-19}$ m^{2}V^{-2}, respectively [84]. As the titania thins film samples in [14] were reported to have a susceptibility of $\chi^{(3)} = 9.9 \cdot 10^{-19}$ m^{2}V^{-2}, which is exactly the same as the previous value reported for anatase thin films, and since the samples presented here have been fabricated in the same manner, it is concluded that the titania thin film sample at hand is most likely anatase or at least shares similar optical properties with anatase. The only quantity for the two samples required in this thesis is the direct band gap. This value is subsequently used to evaluate multiphoton absorption in the samples. For anatase phase, there have been reported band gap values of 2.86 to 3.34 eV in the literature [85], with the majority of reported values for similarly prepared samples found close to 3.2 eV [84]. This value of $E_g = 3.2$ eV for the direct band gap is concluded to be representative for the inspected titania thin film. For the silica sample, the value of $E_g = 9$ eV is used [86].

4.3 Structure of the THIFROG Trace

With the objective of gaining a deeper understanding of the structure and appearance of a THIFROG trace, an analytical discussion on the formation of trace follows. To the best of the author's knowledge, this is the first analytical derivation of a equation describing the different components of a THIFROG trace ever published.

4.3.1 Derivation

Analogously to Equation 3.15, the THIFROG trace can be written as

$$I_{\text{THIFROG}}(\omega, \tau) \propto \left| \int_{-\infty}^{+\infty} \left(\mathcal{E}(t) + \mathcal{E}(t-\tau)\right)^3 \, e^{-i\omega t} \, \mathrm{d}t \right|^2 . \tag{4.3}$$

We will now proceed to expand this equation in order to arrive at a decomposition of $I^{\text{TH}}_{\text{IFROG}}(\omega, \tau)$ similar to what was presented for $I_{\text{SHIFROG}}(\omega, \tau)$ in Equation 3.16. First,

by recalling that $\mathcal{E}(t) = E(t)\exp(i\omega_0 t)$, the cubic term in 4.3 is expanded

$$\begin{aligned}(\mathcal{E}(t)+\mathcal{E}(t-\tau))^3 =& \Big[E(t)e^{i\omega_0 t} \quad + E(t-\tau)\,e^{i\omega_0(t-\tau)}\Big]^3 \\ =& e^{i3\omega_0 t}\Big[E(t) \;+ E(t-\tau)\,e^{-i\omega_0\tau}\Big]^3 \\ =& e^{i3\omega_0 t}\Big[E^3(t) + 3\,E^2(t)E(t-\tau)\,e^{-i\omega_0\tau} \\ & \qquad + 3\,E(t)E^2(t-\tau)\,e^{-i2\omega_0\tau} \\ & \qquad + E^3(t-\tau)\,e^{-i3\omega_0\tau}\Big]. \end{aligned} \tag{4.4}$$

Inserting Equation 4.4 back into Equation 4.3 gives

$$\begin{aligned} I_{\text{THIFROG}}(\omega,\tau) \propto \Bigg| \int_{-\infty}^{+\infty} \Big[& E^3(t) + 3E^2(t)E(t-\tau)\,e^{-i\omega_0\tau} \\ & +3\,E(t)E^2(t-\tau)\,e^{-i2\omega_0\tau} \\ & +E^3(t-\tau)\,e^{-i3\omega_0\tau}\Big] e^{-i\Delta\omega t}\mathrm{d}t \Bigg|^2, \end{aligned} \tag{4.5}$$

where the substitution $\Delta\omega = \omega - 3\omega_0$ was made. Writing the integrals of the different terms in the sum separately yields

$$I_{\text{THIFROG}}(\omega,\tau) \propto \Bigg| \int_{-\infty}^{+\infty} E^3(t)e^{-i\Delta\omega t}\,\mathrm{d}t \tag{4.6a}$$

$$+ 3e^{-i\omega_0\tau} \cdot \int_{-\infty}^{+\infty} E^2(t)E(t-\tau)\,e^{-i\Delta\omega t}\,\mathrm{d}t \tag{4.6b}$$

$$+ 3e^{-i2\omega_0\tau} \cdot \int_{-\infty}^{+\infty} E(t)E^2(t-\tau)\,e^{-i\Delta\omega t}\,\mathrm{d}t \tag{4.6c}$$

$$+ \;\; e^{-i3\omega_0\tau} \cdot \int_{-\infty}^{+\infty} E^3(t-\tau)\,e^{-i\Delta\omega t}\,\mathrm{d}t \Bigg|^2. \tag{4.6d}$$

This equation may be expressed in a more compact way by identifying the four integrals it contains. The first of the integrals 4.6a is defined as the third-harmonic field of a single pulse,

$$E_{\text{TH}}(\Delta\omega) \equiv \int_{-\infty}^{+\infty} E^3(t)\,e^{-i\Delta\omega t}\,\mathrm{d}t\,. \tag{4.7}$$

The second integral found in 4.6b is the standard third-harmonic FROG [34] field,

present also in noncollinear third-harmonic FROG measurements,

$$E_{\text{THFROG}}(\Delta\omega,\tau) \equiv \int_{-\infty}^{+\infty} E^2(t)E(t-\tau)\,e^{-i\Delta\omega t}\,\mathrm{d}t\ . \tag{4.8}$$

By using the translation property of the Fourier transform, the fourth integral in 4.6d becomes

$$\begin{aligned}\int_{-\infty}^{+\infty} E^3(t-\tau)e^{-i\Delta\omega t}\,\mathrm{d}t &= e^{-i\Delta\omega\tau}\int_{-\infty}^{+\infty} E^3(t)\,e^{-i\Delta\omega t}\,\mathrm{d}t \\ &= e^{-i\Delta\omega\tau}\ E_{\text{TH}}(\Delta\omega)\ .\end{aligned} \tag{4.9}$$

Similarily, the third integral in 4.6c can be written as

$$\begin{aligned}\int_{-\infty}^{+\infty} E(t)E^2(t-\tau)\,e^{-i\Delta\omega t}\,\mathrm{d}t &= e^{-i\Delta\omega\tau}\int_{-\infty}^{+\infty} E^2(t)E(t+\tau)\,e^{-i\Delta\omega t}\,\mathrm{d}t \\ &= e^{-i\Delta\omega\tau}\ E_{\text{THFROG}}(\Delta\omega,-\tau)\ .\end{aligned} \tag{4.10}$$

Assuming that the third-harmonic FROG field is symmetric about the delay τ, that is, $E_{\text{THFROG}}(\Delta\omega,\tau)$ is an even function for τ

$$E_{\text{THFROG}}(\Delta\omega,-\tau) = E_{\text{THFROG}}(\Delta\omega,\tau)\ ,\ \forall\,\tau\ , \tag{4.11}$$

equation 4.6 can now, with the help of Equations 4.7–4.11, be written as

$$I_{\text{THIFROG}}(\omega,\tau) \propto \Big| E_{\text{TH}}(\Delta\omega)\ \left[1+e^{-i(\Delta\omega+3\omega_0)\tau}\right] \tag{4.12a}$$

$$+\,3\,E_{\text{THFROG}}(\Delta\omega,\tau)\ \left[1+e^{-i(\Delta\omega+\omega_0)\tau}\right]\cdot e^{-i\omega_0\tau}\Big|^2\ . \tag{4.12b}$$

This is a squared norm of two complex entities, for which it holds true that

$$|z_1+z_2|^2 = (z_1+z_2)^*\cdot(z_1+z_2) = |z_1|^2+|z_2|^2+2\,\Re[z_1^* z_2]\ , \tag{4.13}$$

where z_1, z_2 are complex. Applying Equation 4.13 on Equation 4.12 yields

$$
\begin{aligned}
I_{\text{THIFROG}}(\omega,\tau) \propto \;& |E_{\text{TH}}(\Delta\omega)|^2 \times \left| \left[1 + e^{-i(\Delta\omega+3\omega_0)\tau}\right] \right|^2 && \text{(4.14a)}\\
& + 9\,|E_{\text{THFROG}}(\Delta\omega,\tau)|^2 \times \left| \left[1 + e^{-i(\Delta\omega+\omega_0)\tau}\right] \right|^2 && \text{(4.14b)}\\
& + 3\cdot 2\Re\Big\{ E^*_{\text{TH}}(\Delta\omega) E_{\text{THFROG}}(\Delta\omega,\tau) \;\times && \text{(4.14c)}\\
& \qquad \left[1 + e^{i(\Delta\omega+3\omega_0)\tau}\right] \;\times && \text{(4.14d)}\\
& \qquad \left[1 + e^{-i(\Delta\omega+\omega_0)\tau}\right] \;\times && \text{(4.14e)}\\
& \qquad e^{-i\omega_0\tau} \Big\} . && \text{(4.14f)}
\end{aligned}
$$

Taking a closer look at the third term in the above sum, the three exponential containing terms of 4.14d, 4.14e and 4.14f inside the $\Re\{\ \}$ expression are multiplied

$$
\begin{aligned}
& \left[1 + e^{i(\Delta\omega+3\omega_0)\tau}\right] \times \left[1 + e^{-i(\Delta\omega+\omega_0)\tau}\right] \times e^{-i\omega_0\tau} \\
=& \left[1 + e^{i2\omega_0\tau} + e^{i(\Delta\omega+3\omega_0)\tau} + e^{-i(\Delta\omega+\omega_0)\tau}\right] \times e^{-i\omega_0\tau} \\
=& e^{-i\omega_0\tau} + e^{i\omega_0\tau} + e^{i(\Delta\omega+2\omega_0)\tau} + e^{-i(\Delta\omega+2\omega_0)\tau} \,, \qquad \text{(4.15)}
\end{aligned}
$$

and since the sum of a complex number and its complex conjugate is $z + z^* = 2\Re[z]$, we see that the four exponential terms in 4.15 can be written as a sum of two cosines

$$
\begin{aligned}
& e^{-i\omega_0\tau} + e^{i\omega_0\tau} + e^{i(\Delta\omega+2\omega_0)\tau} + e^{-i(\Delta\omega+2\omega_0)\tau} \\
=& 2\Big[\cos(\omega_0\tau) + \cos\big(\,(\Delta\omega + 2\omega_0)\tau\,\big) \Big] \,, \qquad \text{(4.16)}
\end{aligned}
$$

which is a real quantity. Noticing that terms of the form $\left|1 + e^{i\phi}\right|^2$ can be expanded using Equation 4.13 as

$$
\begin{aligned}
\left|1 + e^{i\phi}\right|^2 &= |1|^2 + \left|e^{i\phi}\right|^2 + 2\,\Re[1^* \cdot e^{i\phi}] \\
&= 1 + 1 + 2\cos\phi \\
&= 2\,(1 + \cos\phi) \,, \qquad \text{(4.17)}
\end{aligned}
$$

the first two terms in Equation 4.14 can be written anew so that 4.14a becomes

$$
\begin{aligned}
& |E_{\text{TH}}(\Delta\omega)|^2 \times \left| \left[1 + e^{-i(\Delta\omega+3\omega_0)\tau}\right] \right|^2 \\
=& 2\,|E_{\text{TH}}(\Delta\omega)|^2 \times \Big[1 + \cos\big((\Delta\omega + 3\omega_0)\tau\big)\Big] \,, \qquad \text{(4.18)}
\end{aligned}
$$

and similarily for 4.14b,

$$\begin{aligned} &9\,|E_{\mathrm{THFROG}}(\Delta\omega,\tau)|^2 \times \left|\left[1+e^{-i(\Delta\omega+\omega_0)\tau}\right]\right|^2 \\ =&2\cdot 9\,|E_{\mathrm{THFROG}}(\Delta\omega,\tau)|^2 \times \left[1+\cos\left((\Delta\omega+\omega_0)\tau\right)\right]. \end{aligned} \tag{4.19}$$

Now, by inserting Equations 4.16, 4.18 and 4.19 into 4.14 we have

$$I_{\mathrm{THIFROG}}(\omega,\tau) \propto 2\cdot|E_{\mathrm{TH}}(\Delta\omega)|^2 \times \left[1+\cos\left(\,(\Delta\omega+3\omega_0)\tau\,\right)\right] \tag{4.20a}$$

$$+\,2\cdot 9\,|E_{\mathrm{THFROG}}(\Delta\omega,\tau)|^2 \times \left[1+\cos\left(\,(\Delta\omega+\omega_0)\tau\,\right)\right] \tag{4.20b}$$

$$+\,2\cdot 2\cdot 3\,\Re\left[E^*_{\mathrm{TH}}(\Delta\omega)\,E_{\mathrm{THFROG}}(\Delta\omega,\tau)\right] \times \tag{4.20c}$$

$$\left[\cos(\omega_0\tau)+\cos\left(\,(\Delta\omega+2\omega_0)\tau\,\right)\right], \tag{4.20d}$$

where the cosines of Equation 4.16 were ejected from the $\Re\{\,\}$ expression as they are real. Dividing by 2 and rearraging finally yields the dissected form of the original Equation 4.3,

$$I_{\mathrm{THIFROG}}(\omega,\tau) \propto\ |E_{\mathrm{TH}}(\Delta\omega)|^2 + 9\,|E_{\mathrm{THFROG}}(\Delta\omega,\tau)|^2 \tag{4.21a}$$

$$+\,6\cos(\omega_0\tau) \times \Re\left[E^*_{\mathrm{TH}}(\Delta\omega)E_{\mathrm{THFROG}}(\Delta\omega,\tau)\right] \tag{4.21b}$$

$$+\,9\cos\left(\,(\Delta\omega+\omega_0)\tau\,\right) \times |E_{\mathrm{THFROG}}(\Delta\omega,\tau)|^2 \tag{4.21c}$$

$$+\,6\cos\left(\,(\Delta\omega+2\,\omega_0)\tau\,\right) \times \Re\left[E^*_{\mathrm{TH}}(\Delta\omega)E_{\mathrm{THFROG}}(\Delta\omega,\tau)\right] \tag{4.21d}$$

$$+\quad\cos\left(\,(\Delta\omega+3\,\omega_0)\tau\,\right) \times |E_{\mathrm{TH}}(\Delta\omega)|^2\ . \tag{4.21e}$$

4.3.2 Interpretation

The unmodulated first two terms in 4.21e are the fundamental third-harmonic field and the regular third-harmonic FROG trace, respectively, together forming the DC baseband of the THIFROG trace. The third (b) and fourth terms (c) are both modulated at the carrier frequency ω_0, although only (b) has a modulation which stays constant at different frequencies of the trace. The terms (c–e) are modulated at 1, 2, and 3 times ω_0, respectively. Because of the presence of the term $\Delta\omega\cdot\tau$ in the cosines of (c–e), the modulation frequency of these terms at THIFROG frequencies above $3\,\omega_0$ is increased, while the modulation frequency below it is decreased. As can be observed from the example THIFROG trace in Figure 4.4a, this gives the fringes around the center of the trace their skewed appearance, as they experience ever faster modulation when moving to higher frequencies (lower wavelengths) of the trace. Since the first and the last terms are the only terms without a factor of 6 or 9, it is expected that these coincidentally only terms including the plain THG intensity are

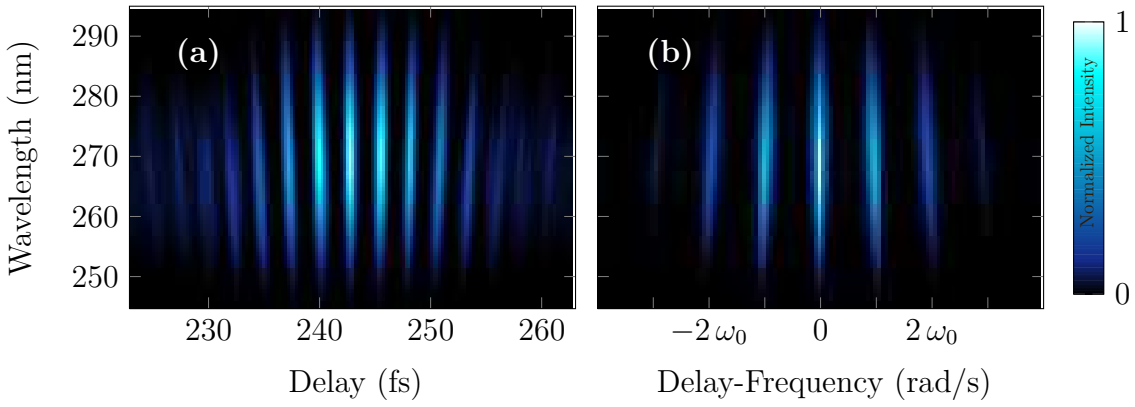


Figure 4.4: THIFROG trace (a) and its Fourier transform along the delay axis (b), where the different delay components are clearly visible as separate bands. Sidebands at $\pm\,\omega_0$ correspond to the fundamental modulation component of the trace, i.e., the FM-band, while sidebands at $\pm 2\,\omega_0$ and $\pm 3\,\omega_0$ correspond to the higher harmonic sub-traces.

left in the shadow of the stronger terms, all of which contain the THFROG electric field. As the fundamental modulation consists of two terms, (b) and (c) with factors of 6 and 9, respectively, it is though to be the strongest as the $2\,\omega_0$ modulated term has only a factor 6 and $3\,\omega_0$ has none. This finding is also supported by inspecting Figure 4.4b, the Fourier transform of a THIFROG trace in the delay-frequency space, where the first two sidebands corresponding to modulation at the fundamental frequency are the strongest, and subsequent sidebands are always weaker than the one before. As predicted by the equation 4.21, exactly three pairs of sidebands are found, as can be observed in the figure.

4.3.3 Evaluation

The formulation of Equation 4.21 relies on two assumptions. Firstly, the two pulses must have identical amplitude profiles, that is, the pulses have identical electric fields at zero delay

$$E_{\text{undelayed}}(t) = E_{\text{delayed}}(t-\tau)\ ,\ \tau = 0\ . \tag{4.22}$$

This is assumed in the formulation of all the traditional FROG traces described by Trebino [34]. Such an assumption, however, does not hold if the power of the incident beam is not equally divided unto the two interferometer arms, e.g., because the beam splitter might not have a perfect 50/50 division ratio, or even worse, the ratio might not be the same for the entire frequency bandwidth of the pulse resulting in distortions. As this fact is not considered a problem in the formulation of other analytical FROG equations, there is no reason why this should be a matter of concern with THIFROG either. The second assumption stated in Equation 4.11,

however, is not always justified, as THG FROG traces are generally *not* perfectly symmetric with respect to delay [9,45]. The use of Equation 4.21 to separate different components of the trace is therefore unjustified in the general case, and instead the untreated Equation 4.3 must be used for pulse retrieval. Nevertheless, the dissected form of Equation 4.21 is a valuable tool in understanding the structure of a THIFROG trace. It clearly states the presence of a DC-term and modulational terms at three distinct frequencies, responsible for the fringe structure seen in THIFROG traces, the skewness of the fringes, and the decreasing magnitude of sidebands with increasing modulation frequency.

5. PULSE RETRIEVAL SOFTWARE

In this chapter the Reconstruction program—a pulse retrieval software constructed and designed specifically for the purposes of this thesis—is discussed. Implemented on MATLAB, the program consists of over twenty functions, revolving around the main function dubbed as **Reconstruction**. The source code is provided in its entirety in Appendix A. The different aspects and key features of the program are divided into the following sections. The general strategy for pulse reconstruction 5.1, computation of the trace 5.2, use of Fourier filtering to obtain the sub-traces 5.3, and the simplex algorithm used in the optimization 5.4 are discussed. Afterwards, the actual use of the program is outlined in Section 5.5, and discussed more thoroughly in Sections 5.6 and 5.7.

5.1 Strategy for Pulse Reconstruction

In its core, the Reconstruction program seeks a pulse whose THIFROG trace best matches the experimental data provided to the program. Beginning with a seed pulse, the amplitude and the phase of the pulse is iteratively adjusted at a preconfigured number of data points to produce a better fitting trace.

In practice the program uses a built-in MATLAB function **fminsearch** to optimize the pulse. This function is an implementation of the *Nelder–Mead simplex algorithm*, which will be discussed in detail in Subsection 5.4.2. The **fminsearch** function tries to minimize the FROG error (see Equation 3.10), which is calculated from the current iteration of the pulse by following a procedure described in the flowchart of Figure 5.1. The traces *per se* are not used to calculate the FROG error, which is instead composed of the weighed sum of the errors of the different delay-frequency bands of the traces, which are discussed in Section 5.3. The user may decide which bands to use and how to weigh them.

Once the FROG error has diminished to a certain level, the algorithm terminates. In practice, however, this might never happen if the local minimum the algorithm is converging toward is simply too far off from the optimal solution, making it impossible to decrease the error beyond a certain degree. Another possibility is that

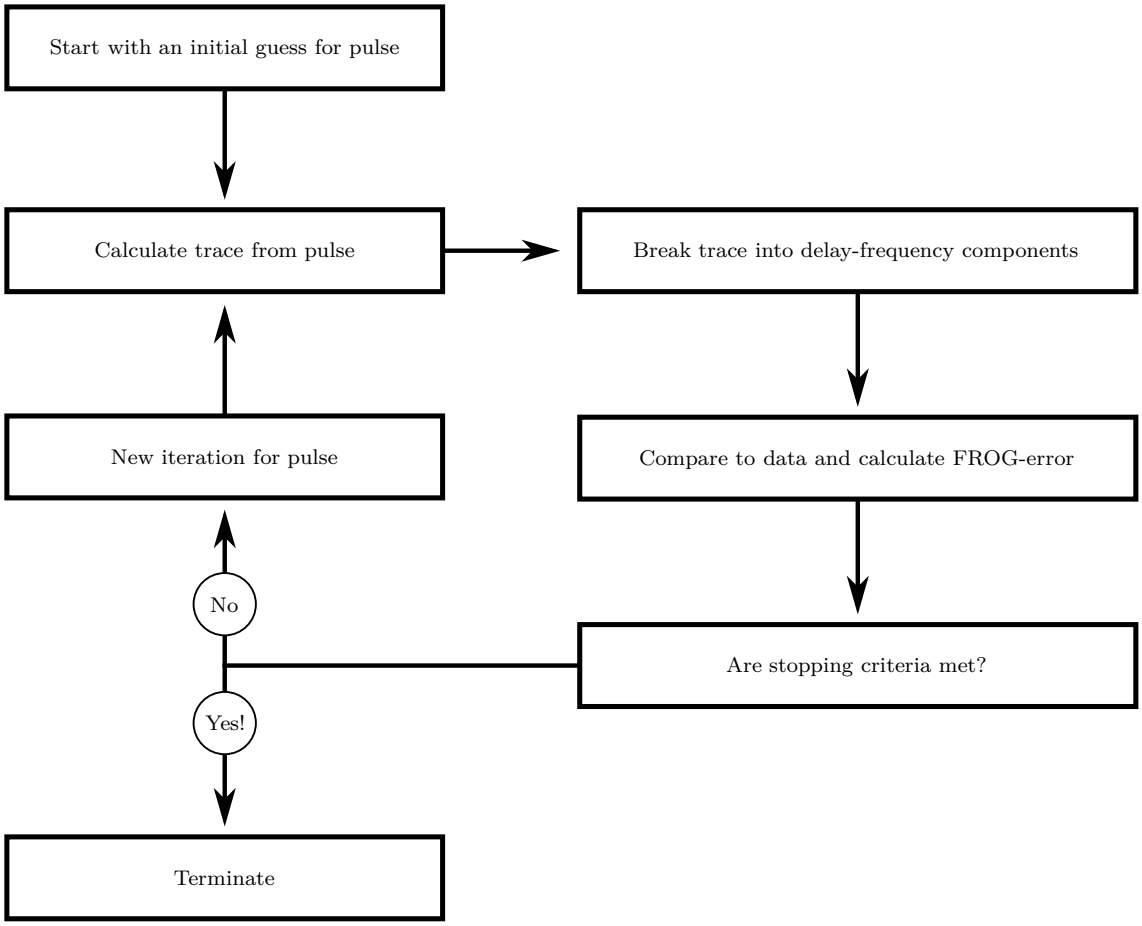


Figure 5.1: Flowchart for the pulse reconstruction strategy followed in the program. First, the program creates a complex-valued seed pulse based on the user provided parameters. The trace of the seed pulse is computed with a procedure further described in Section 5.2. The trace is broken into its selected delay-frequency components, or sub-traces, by Fourier filtering discussed in Section 5.3. The measure for the goodness of the fit is a weighed sum of the FROG errors of the sub-traces in comparison to the measurement data. Each of the FROG errors are calculated as is described in Subsection 3.3.3. The program uses a simplex algorithm, the subject of Subsection 5.4.2, to minimize the weighed error by adjusting the pulse so that the trace it produces better fits the measurement data. The algorithm will converge towards a local minimum of the FROG error and produce a possible solution—the reconstructed pulse—for the inverse problem at hand.

the convergence speed stagnates so severely that it would take an impractical amount of computation time to reach the desired magnitude of error. For these reasons, the algorithm will terminate after a preconfigured time has elapsed. For the simulations presented in Chapter 6 this time limit was 24 hours. It should be noted that speed was not a priority when the program was designed, as the goal is to successfully reconstruct the traces as accurately as possible. Since only a few simulations were needed, it is perfectly reasonable to allow a full day for each simulation to finish.

5.2 A Pulse and its Trace

The THIFROG trace of a laser pulse is measured by recording the third-harmonic field generated by the pulse and its copy with a spectrometer, so that the replica pulse is variably delayed in respect to the original. The result is a two dimensional, frequency versus delay image—a spectrogram. This measurement can be readily simulated. The procedure for computing the trace of a pulse is illustrated in Figure 5.2, and will be discussed in detail next.

5.2.1 Computation of a Trace

Beginning with the complex-valued seed pulse sampled at twice the frequency of the delay step, but with half the length in time than that of the delay axis, two matrices are formed. The first matrix is for the undelayed pulse. The delay axis τ for the matrices ranges from $\tau = -\gamma$ to $\tau = +\gamma$, where γ is the length of the time axis for the pulse. The delay axis is identical to the one of the measured trace, after it has been prepared for the program. The time axis is $3\,\gamma$ long, or thrice the length of the pulse. The reason for this becomes obvious when the second matrix is introduced. The undelayed pulse is copied onto each delay coordinate with zero padding of length γ on both sides, producing the first matrix depicted in Figure 5.2b.

The second matrix of Figure 5.2c is for the delayed pulse. On the two extremes of delay, the pulse is delayed by an amount equal to its length in time so that on the left side at $\tau = -\gamma$, the delayed pulse actually precedes the undelayed pulse in time *exactly* enough so that the two pulses do not overlap in time at all. At the center of the delay axis lies the zero delay point $\tau = 0$, where the two pulses are perfectly aligned. On the right at $\tau = +\gamma$ the delayed pulse is lagging right behind the original pulse.

Each complex element of the first matrix is added to the corresponding element of the second matrix, and each element of the resulting sum matrix is raised to the third power. This represents the physical third-harmonic generation process, where the cube of the electric field produces light at new frequencies. The resulting matrix in Figure 5.2d is the trace of the two pulses in time-delay (t, τ) space. The more familiar frequency-delay (ν, τ) picture is formed by taking the Fourier transform of the sum matrix along the time axis, yielding the simulated THIFROG trace of Figure 5.2e. This entire procedure is essentially a discretization of Equation 4.3.

If the delay axis were any longer (or the pulse any shorter), no new, relevant information would be obtained. The THIFROG trace on larger delays would look

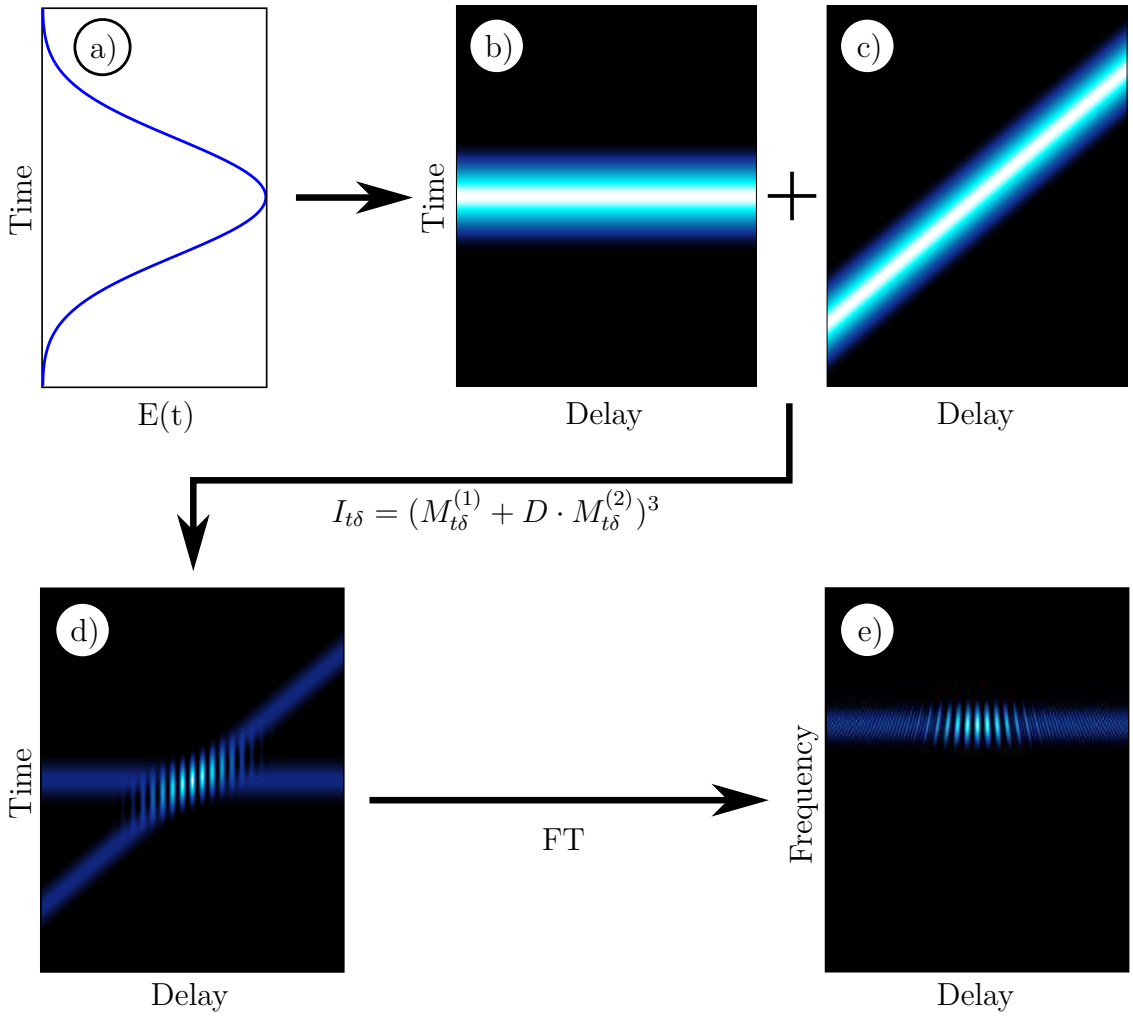


Figure 5.2: Schematic presentation, explaining how the trace of a pulse is calculated in the program. The pulse (a) is used to generate two zero-padded matrices: (b) one with no delay imposed upon the pulse and (c) a second with a variable delay. The respective matrices $M^{(1)}$ and $M^{(2)}$ are inserted into the equation $I_{t\delta} = (M^{(1)}_{t\delta} + D \cdot M^{(2)}_{t\delta})^3$, where D is the division factor of the beam splitter, to produce (d) the trace $I_{\mathrm{THIFROG}}(t, \delta)$. Fourier transform along the time axis yields (e) the THIFROG trace in the frequency domain. All of the plotted quantities are absolutes of the complex-valued quantities.

exactly the same as on current the edges at $\tau = \pm\gamma$, as the two pulses would not overlap in time at all, producing a third-harmonic signal of twice the third-harmonic signal of a single pulse, regardless of how large the delay would be. Any shorter delay axis (or a longer pulse) and the trace would not be fully computed, as part of the trace produced by overlapping pulses would reside in larger delays than fit the picture. The choice for the maximum pulse length (γ) of half the delay axis length (2γ) is therefore optimal.

During the reconstruction, a new pulse is formed several times in every iteration by adjusting the pulse of the previous iteration at preconfigured number of optimiza-

tion points. Typically there are less of these points than there are time samples of the pulse. Therefore, the pulse to be adjusted has to be sampled at the optimization points before altering any values. This is achieved by linear interpolation of amplitude and phase of the pulse. The interpolated values are altered by the optimization algorithm, after which the pulse is again sampled at the original time axis by linear interpolation. The complex-valued equivalent of the pulse is computed, and only after this is the trace of the adjusted pulse computed to determine, whether it is a better fit than the trace produced by the previous pulse. Should this be the case, the new pulse takes the place of the previous pulse, and a new iteration cycle begins.

5.2.2 The Seed Pulse

The very first pulse, referred to as the *seed pulse*, is formed by using the following formula to produce a bell like shape for the amplitude profile of the pulse,

$$A(t) = \frac{2}{\exp(t/t_{\mathrm{f}}) + \exp(-t/t_{\mathrm{r}})} \,, \tag{5.1}$$

where t_{r} and t_{f} the rise and the fall times, respectively. The rise and fall times (from 10% to 90% intensity and vice versa) for a particular THIFROG trace are estimated by the user with the assistance of the program, as is explained in Subsection 5.6. Essentially, Equation 5.1 is a hyperbolic secant shaped amplitude profile, making the intensity profile sech^2 shaped, with asymmetric rise and fall times. This corresponds to the assumed shape of the pulses produced by the VENTEON laser system — used in the experimental setup for the THIFROG traces analyzed in Chapter 6— as was presented in Figure 4.3b, with the added possibility for asymmetry. Such asymmetry can be induced in the pulse when it interacts with the samples, so allowing the software to start with an asymmetric seed pulse can lead to a faster initial convergence and decrease overall computation time. To reduce the presence of often irrelevant intensity distribution near the edges of the time range for the simulated pulse, a Blackman window function [87] is used to force the electric field amplitude of the pulse to zero at the very edges. A flat phase of π radians is used. Another alternative would be to use random noise for the phase, but the flat phase proved to be a more reliable option during testing. The fact that the phase level is set at π radians instead of zero is because the optimization algorithm functions more reliably when working with finite numbers instead of values close to zero. The pulse in Figure 5.2a is formed in this manner with equal rise and fall times and a flat phase.

5.2.3 Pulse Representations

Although the figures produced by the program always present the pulse by its intensity and phase profiles, and the optimization is accomplished by modifying the amplitude and phase instead of the complex electric field, the pulse is always converted into complex form in order to compute its THIFROG trace. This is accomplished simply by using Equation 2.30, i.e.,

$$\tilde{\mathcal{E}}(t) = A(t)e^{-i(\,\omega_0 t+\varphi(t)\,)}\,.$$

In order to go back to the amplitude and phase representation from the complex field $\tilde{\mathcal{E}}(t)$, the amplitude is obtained by

$$A(t) = \sqrt{\left|\tilde{\mathcal{E}}(t)\right|^2}\,, \tag{5.2}$$

and the phase by using a built-in MATLAB function **phase**.

There is no fundamental reason why the optimization could not be realized by adjusting the real and imaginary parts of the complex electric field $\tilde{\mathcal{E}}(t)$ instead of the amplitude $A(t)$ and phase $\varphi(t)$. This choice was simply a matter of convenience in the programming, and of personal preference, as the author finds the amplitude and phase representation of the physical pulse to be the more intuitive one out of the two options.

5.3 Fourier filtering

An essential role in the pulse reconstruction sequence is played by the method of *Fourier filtering*. In general, the filtering procedure involves a transform to the spectral domain of a quantity, applying a desired filtering function to mitigate the presence of certain frequency components, and finally an inverse Fourier transform back to the temporal domain. Fourier filtering is used extensively in the fields of digital image processing [88] and signal processing [89]. Typical uses include removal of high frequency noise from a signal through *low-pass filtering*, image sharpening or compression, or isolation of a certain frequency band through *band-pass filtering* to study it without the interfering presence of other frequency components. The last of the examples is applied also in this thesis.

5.3.1 Fourier Filtering in Pulse Retrieval

The reconstruction program uses Fourier filtering to extract different delay-frequency bands or *sub-traces* from both the measured THIFROG trace as well as the simulated trace, so that these sub-traces may be compared and used as a measure for the goodness of the fit. The reason for this is that because the fringes of the THIFROG trace are quite narrow, less than a femtosecond wide, it can be difficult to match the positions of the calculated and measured traces along the delay-axis so that the fringes of the two traces would overlap correctly from the very start of the reconstruction. Should this positioning fail, the algorithm might not be able to correct this delay shift, making convergence to a possible solution impossible. The *DC-baseband* or simply *baseband* (zero delay-frequency component) of a trace, however, does not contain any fringes, so a slight offset in delay can eventually be corrected by slowly adjusting the pulse so that the basebands of the two traces match. One could optionally use the fundamental modulation or the higher harmonic sidebands, i.e., the delay-frequency components of the trace that are modulated at the carrier frequency ω_0 or at $2\omega_0$ for the fitting procedure. The program allows for the use of any combination of the sub-traces excluding the third-harmonic sideband modulated at $3\omega_0$, which should not contain any useful information for the pulse retrieval.

The optimization of the pulse is a computationally intensive process, largely due to the fact that after each adjustment of the phase and the amplitude of the pulse, the trace has to be calculated first (which already involves one Fourier transform), its bands have to be extracted (2 transforms for each band, one forward and one inverse) and only then can the error of the fit be calculated to establish whether the adjustments were for better or for worse. When both the baseband and the FM-band are used in the reconstruction, each iteration of the pulse requires 5 Fourier transforms. The first transform to compute the trace is executed with the well established *fast Fourier transform* (FFT) algorithm [90], which is a computationally very effective method for the task. Unfortunately, FFT in MATLAB computes the *discrete Fourier transform* (DFT) for *all* the frequency samples in the range $-\omega_{\max} \ldots + \omega_{\max}$, where $\omega_{\max}$ is the maximum frequency, dictated by the number of points the FFT is evaluated at. In the sub-band extraction this information is not required, as the approximate location of the sought after band in the delay-frequency domain is well established to be a multiple of the carrier frequency ω_0. The data corresponding to the second-harmonic sideband, for example, resides within small stretches around the delay-frequencies $+2\,\omega_0$ and $-2\,\omega_0$. Furthermore, the Fourier transform has the property that for real valued data, the transform is self-adjoint [91], i.e., the data in the negative frequencies equals to the complex conjugate of the corresponding positive frequencies. Since the trace is real valued measurement

data, its transform at the negative frequencies can in fact be derived from the signal at the positive frequencies. This information is, however, useless for our purposes—except for the baseband computation, where half of the band resides in negative frequencies and is obtained from the positive half using the self-adjoint property. Otherwise the computation of the baseband follows the same principle as the harmonic sub-band extraction described next.

5.3.2 The Filtering Procedure

In order to extract a harmonic band, it is sufficient to calculate the Fourier transform for the positive frequencies in the vicinity of the appropriate multiple of ω_0. This is accomplished by using a custom build function, which is essentially a variation of the standard DFT formula [91],

$$H_n = \sum_{k=0}^{N-1} h_k e^{2\pi i k n/N} \ . \tag{5.3}$$

The equation above maps N complex variables, h_k, from the original domain to N complex variables, H_n, in the Fourier domain. k and n are indices for the samples in their respective domains. The indices n for the samples H_n corresponding to the relevant frequencies can be calculated, and k takes all the values from 0 to $N-1$, where N is the number of delay samples. The fact that the FT is always calculated for the same frequencies, makes the job much easier. The exponential term of Equation 5.3 needs to be calculated only once for each sub-band, whereas the built-in FFT function would do this every single time. The only variable which varies on each iteration of the reconstruction is the the trace data h_k, so the program multiplies the new h_k with the already known exponential terms and computes the sum of these to obtain H_n. Note that this function only performs the DFT for the strip of data at a single frequency, so the function has to be run for each frequency sample separately.

No filtering of the transformed data is required as only relevant frequencies were transformed. Instead, an inverse DFT is executed, following a similar formula to the forward DFT of Equation 5.3,

$$h_k = \frac{1}{N} \sum_{n=0}^{N-1} H_n e^{-2\pi i k n/N} \ . \tag{5.4}$$

The exponential term in the above equation is identical to that of the previous equation aside for the minus sign, and again it is necessary to calculate this term only once for each sub-band. After applying the inverse DFT, the sub-band has

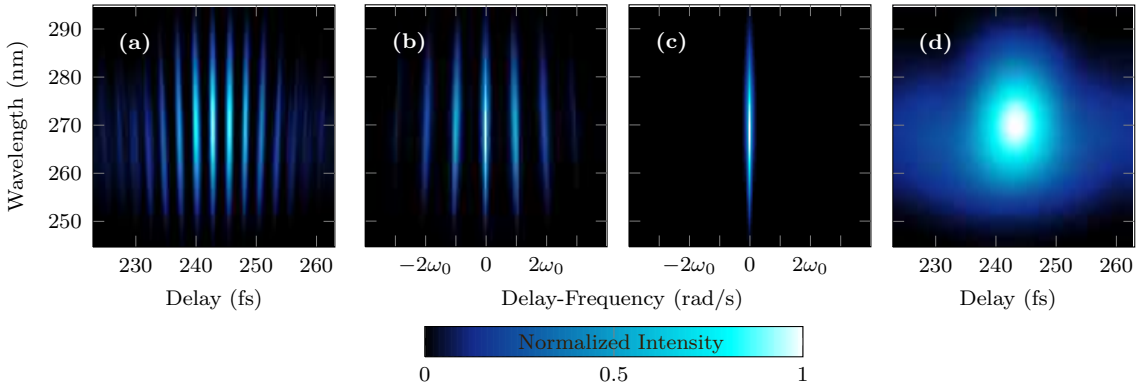


Figure 5.3: Fourier filtering of the baseband. The trace (a) is Fourier transformed along the delay axis to produce (b) the trace in delay-frequency domain, where the different delay components are clearly visible as separate bands. Sidebands at $\pm\omega_0$ correspond to the fundamental modulation component of the trace, i.e., the FM-band, while sidebands at $\pm 2\,\omega_0$ and $\pm 3\,\omega_0$ correspond to the higher harmonic sub-traces. In order to preserve only the DC baseband, all but the band around zero delay-frequency are removed (c). Taking an inverse Fourier transform back to delay space yields the baseband (d). The THIFROG trace in (a) is a close-up of actual experimental data for the silica thin film after some initial processing, seen in Figure 6.1c with a different color map.

been extracted. The procedure of Fourier filtering the baseband from a trace is demonstrated in Figure 5.3. FFT is still used for the calculation of the trace, because all the positive frequencies beginning from 0 are used, negating the gain for using the custom built DFT function as one of its main advantages is the selection of only a handful of frequencies. The built-in FFT function is also much more simple to implement. The decision to use it made the already quite complex program a bit tidier, and encountering programming errors a slightly less likely event during development.

5.4 Optimization

Optimization is the act of minimizing or maximizing a function. These two are trivially connected as $\min[f(x)] = -\max[f(x)]$, so only minimizing is referred to from here on. Different kinds of functions, or different kinds of problems, have their own peculiarities, and a poor choice of algorithm can mean that the problem stays unsolved. There is a plethora of optimization algorithms available. Each of them have their strengths and weaknesses, and some are better suited to a particular problem than others. The reasons why Nelder–Mead simplex algorithm was chosen for the problem at hand, the pulse retrieval, is discussed next.

5.4.1 Choice of Algorithm

Optimization algorithms can be categorized by a few qualities [91]. First one is the scope, or whether the algorithm is equipped to search for a local, or a global minimum. Second is the use of derivatives, i.e., whether the algorithm requires their computation or not. Third are the constraints, whether the algorithm makes use of *a priori* limits for the function or its form, e.g., a variable to be optimized might be required to be positive. Fourth is whether the function is linear or not. Fifth is the dimension, are the variables scalar, or are they vectors? One factor is also the use of memory: if the algorithm requires a lot of memory with many variables, its use may be effectively prohibited for problems with large amount of variables. Speed of convergence is always wanted, but it is not essential for every problem.

Let us go through the different qualities one by one for the case of the pulse retrieval problem.

1. Global minimum is desired, of course, but this can be extremely difficult to find. Since an approximate solution can be devised, i.e., the seed pulse can be formed by evaluating the width of the baseband and the carrier frequency can also be approximately defined, finding a local minimum should be sufficient.

2. The derivatives cannot be computed for the function used in the pulse retrieval, i.e., Equation 4.3.

3. The problem is unconstrained. The complex electric field can, in principle, have any values. However, it is possible to limit the normalized intensity values to $0 \ldots 1$, but this was not considered necessary.

4. The function is nonlinear, as the relation between the pulse and the trace, given by Equation 4.3, is certainly not linear.

5. The problem is multidimensional. There are as many variables as there are evaluation points for the pulse envelope *times two*, since both phase and amplitude are adjusted. For the simulations of Chapter 6, there were 50 points where the pulse was evaluated, so there were 100 variables to be optimized. This is a fairly large quantity.

These qualities narrow down the list of possible choices. Gradient methods cannot be used as the derivatives are unknown. Instead, a direct search method, where only the function values are adjusted, is required. Linear programming is not possible because of the nonlinearity of the function. Monte Carlo method is a global

method, but due to the complexity of the problem, the use of this method would result in poor convergence, so that finding a global minimum would be hopeless. Only two viable options are left: Nelder–Mead and Powell's method [92]. Both are local strategies that are also unconstrained, direct search methods suitable for multidimensional problems. Powell's method should be faster, but it is more difficult to implement [91]. The Nelder–Mead, on the other hand, is *already* implemented on MATLAB. Because speed is not the main concern here, the simpler but somewhat slower Nelder–Mead is chosen.

5.4.2 Simplex Algorithm

The MATLAB–function **fminsearch** is based upon the simplex algorithm described by Lagarias *et. al* [93], which is, in turn, an improved version of a famous and widely used method for nonlinear unconstrained optimization, the Nelder–Mead simplex algorithm, first published in 1965 [94]. The algorithm, also known as the *amoeba algorithm* or the *downhill simplex method* [91], is used to minimize a scalar–valued nonlinear function of n real valuables without any knowledge about the derivatives of the said function. Since the functions that are to be minimized in the context of this thesis, i.e., the laser pulses, are complex-valued, the pulse is converted in to its phase and amplitude representation following the procedure described in Subsection 5.2.3, so that the algorithm only has to deal with real valued data. The Nelder–Mead algorithm is a direct-search method as only the function values are adjusted in the course of an iteration cycle of the optimization.

The *simplex* is a geometrical figure of non-zero volume formed by $N + 1$ points or *vertices* in N-dimensional space. Furthermore, the simplex is the *convex hull*, i.e., the smallest convex set that contains all of the $N + 1$ vertices. The concept of a convex hull can be visualized in two dimensions by imagining a rubber band flexed around all the vertices as tightly as possible, the area enclosed by the rubber band thus forming the convex hull. A *convex set* is a set in which any pair of two points can be connected with a straight line segment so that the line segment itself is also contained within the set. An example of a *nonconvex set* could be a crescent shape or any shape with a hole in it.

Each iteration of the Nelder–Mead algorithm begins with a simplex, the first one being the initial guess hopefully close to the minimum to be found. The algorithm then decides which of the five procedures, illustrated in Figure 5.4, is to be used to obtain a new vertex to replace the vertex with the highest function value, thus forming a simplex for the next iteration cycle. The simplex adjusts itself to the "local landscape" in the N-dimensional space by contracting in the neighbourhood

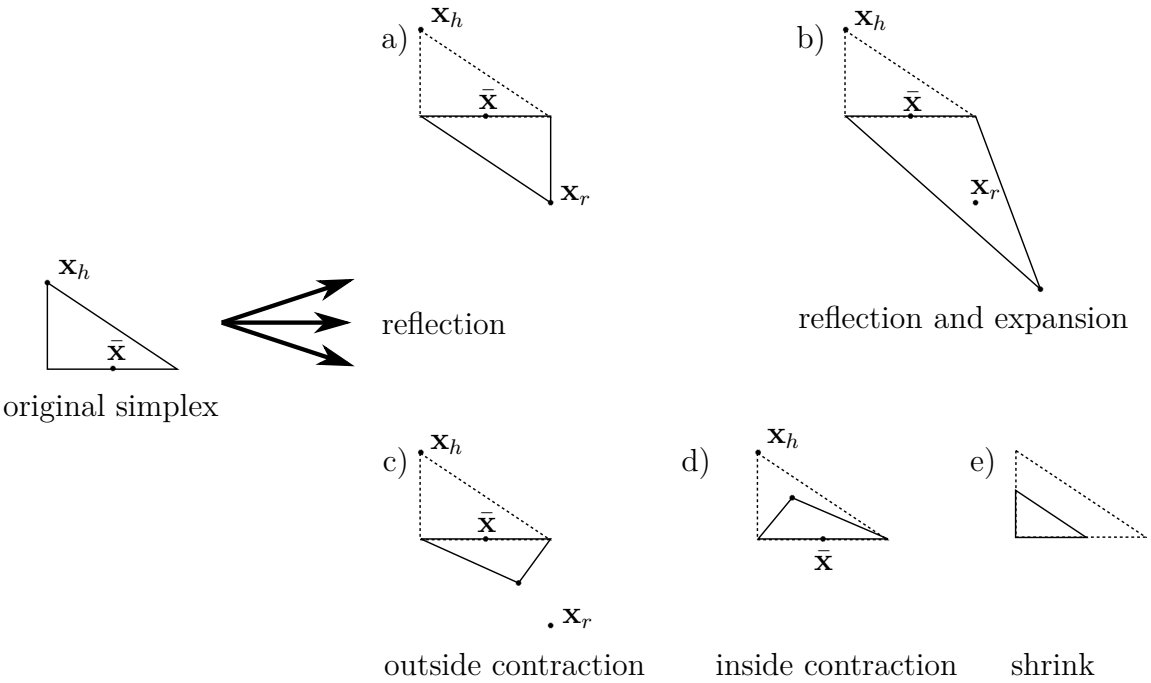


Figure 5.4: The five possible steps to modify a simplex in an iteration cycle of the Nelder–Mead algorithm. The original simplex, in this case a triangle, is shown on the left, while the outcomes of the different operations are on the right along with the original simplex (dashed line). These operations are: (a) reflection away from the high point, (b) a reflection and an expansion away from the high point, (c) outside contraction, (d) inside contraction and (e) shrinking toward the low point (the point with the lowest function value) along every dimension. The point $\bar{\mathbf{x}} = \sum_{i=1}^{N} \mathbf{x}_i / N$ is the centroid of all vertices excluding the high point $\mathbf{x}_h$, i.e., the point which gives the highest value of the minimizing function, and $\mathbf{x}_r$ is the reflection of the high point through $\bar{\mathbf{x}}$. By choosing an appropriate sequence of these steps a convergence toward a local minimum will result in most cases [91, 93].

of a minimum, stretching down along inclined planes and changing direction when a valley is encountered at an angle.

The Nelder–Mead algorithm does not always converge for general nonconvex functions, such as Equation 4.3, but it has been found to work well in practice and to provide a rapid initial decrease in function values [93].

5.5 Work Flow

The pulse retrieval software consists of three parts relevant to the user: the function to prepare the data, the script to run the main function, and the graphical user interface for controlling the pulse retrieval. The operation of the three is outlined in the following steps.

1. Start by running the **PrepareData** function to prepare experimental data for the use of the program. This will create two files, labelled with the letters D and P for data and parameters, respectively. See Section 5.6 for details.

2. Use a script to provide the necessary parameters for the **Reconstruction** function, and to start the simulation. The parameters are explained in Subsection 5.7.1.

3. Use the graphical user interface of the main program to monitor and control the pulse retrieval. This discussed in Subsection 5.7.3.

4. Once the pulse retrieval is terminated, either automatically or by the user, the data produced by the simulation is saved to a predefined location.

An example of a script for step (2) is provided in the file *DataGenerator.m* in Appendix A. This script enables the use of an automated sequence to run several simulations in succession and saving the produced data in a convenient location, e.g., in the subfolders of the program. Although the user might wish to observe the progress of the optimization and possibly decide to terminate the simulation once convergence begins to stagnate, it can be a good idea to let the simulations run their course and see the results once everything is finished, as a single simulation might take quite some time to finish. The simulations presented in the following chapter had a time limit of 24 hours each, taking all together almost a week to finish!

5.6 Preparations for Experimental Data

Before the measurement data is used for pulse retrieval, one must provide the program with important parameters about the data with the custom build **PrepareData** function. The function will assist the user in this procedure with a graphical user interface (GUI). The parameters that need to be defined are: minimum wavelength, maximum wavelength, minimum delay, maximum delay, rise time, fall time, optical cycle length, delay step size for optimization, power division of the beam splitter and the coordinates for the point of zero delay and center frequency. These will be discussed next.

5.6.1 Parameters

The minimum and maximum wavelengths and delays are parameters used in the original measurement and are provided along the data by the Max Born Institute. The rise and fall times and the optical cycle length are graphically estimated by the

user with the aid of the **PrepareData** function and are used to make an estimate for the width of the pulse envelope for the seed pulse.

The optical cycle length is deduced from the fringe spacing of the trace as the fringes are known to be separated by a delay $\Delta\tau = \lambda_0/c$. This is simply because there is a local intensity maximum of the trace whenever the two pulses overlap perfectly, which occurs periodically after a delay equal to a full optical cycle. As the length of the optical cycle depends on the wavelength, and the pulses contain a range of wavelengths, the THIFROG trace is in turn spread over a range of wavelengths corresponding to the sum of the frequencies of the frequency–tripled photons. Since only a single value of the optical cycle is used for the initial estimate of the width of the pulse envelope in the program, the selection the wavelength for which the optical cycle is deduced represents an ambiguity. The user should select a wavelength close to the center of the THIFROG as it is the most representative wavelength possible. This ambiguity, however, is ultimately only a minor concern, as the carrier frequency is deduced separately, and this value of the optical cycle length is used only for forming the seed pulse envelope.

In order to determine the location of the zero delay point, i.e., the value of delay that corresponds to the maximal overlap of the two pulses, and of the central frequency, from which the carrier frequency used in the slowly varying envelope approximation is obtained, the user is assisted graphically to Fourier filter the baseband of the trace. The baseband is extracted for the reason that it can be difficult to estimate the sought after point from the original, highly modulated trace, where a single fringe is represented by only a couple of delay coordinates, making the pinpointing of the peak of a fringe problematic. The unmodulated baseband contains no fringes, so the intensity maximum can easily be located. The intensity maximum of the baseband is a good, unambiguous estimate for both the central frequency and the zero delay and used in all the simulations of this thesis. Still, one has an option to select any other point as well, should it be necessary.

The user is also asked to select a wavelength range in which the THIFROG trace resides, so that irrelevant measurement data—wavelength ranges where virtually no signal is measured—can be excluded. In all of the simulations conducted in this thesis, the power division of the beam splitter is assumed to be exactly $1:1$, i.e., the beam splitter of the interferometer splits the incident optical power perfectly in half for the entire frequency range of the pulse. This, however, might not always be the case. It is up to the user to decide whether to adjust the ratio so that the program gives one of the pulses more power than the other. In practice this is achieved by multiplying the complex amplitude of the delayed pulse by the power division

of the beam splitter variable before the resulting THIFROG trace is calculated. Wavelength dependency of the power division is currently not taken into account. Including a support for user-provided information of the wavelength dependency of the power division, possibly independently measured, is a plausible improvement for the program.

The size of the delay step given by the user can be the same as the actual delay step of the measurement, which was 0.25 fs for both of the Max Born Institute samples, but this is not absolutely necessary. For pragmatical reasons, a new delay axis is constructed so that it contains the previously mentioned "zero delay point", i.e., the delay where the two pulses supposedly perfectly overlap, a point which almost certainly does not coincide with any of the original delay points. Interpolation is therefore required to fit the data to the new set of delay points. And since interpolation is carried out in any case, sampling of the data at a slightly different delay spacing is not very detrimental. Delay step size used in the simulations here was 0.2 fs, half a femtosecond less than the original sampling time. This value was found to work well for the simulations, and it was selected as it is a nice even number quite close to the original value.

5.6.2 Removal of Artefacts

In addition to providing parameters, possible artifacts in the data are removed by the user with a GUI if deemed necessary. The **PrepareData** function initiates this GUI after other parameters have been provided. These artifacts occur as small, intense spots in the measured trace, often only a single pixel wide. A suitable level of measured intensity is selected near the artifact, which is "painted over" with the cursor so that the area becomes flat. These artifacts can appear in any region of the measurement data and their removal is easily justified if a single intense pixel is found in a sea of background noise. Occasionally an artifact might reside right next to or within actual THIFROG signal data. In this case the user has to carefully consider whether to remove the supposed artifact and risk tampering with an actual data point of the real signal, or to leave it at its place which will in turn result in high delay-frequency components of the trace that can interfere with the reconstruction process. If possible, one should try to choose a measurement with little or no possible artifacts within the area of the measured signal to avoid ambiguities. This of course is only possible if several measurements of the same setup were made in the first place.

5.6.3 Data and Parameter Files

After these steps have been carried out, the **PrepareData** function produces two files, which include the possibly edited THIFROG trace in a form suitable for the main program, and the necessary parameters. These .mat files share the name of the original text file containing the measurement data, and are labelled with the letters D and P (for Data and Parameters), respectfully.

The **PrepareData** function allows the user to select and process multiple data files at once. If consecutive data sets are measured with the same parameters, the user may use the same parameters entered only once for the following datasets. A previously created P file containing the parameters can also be selected. These features are included to avoid unnecessary manual inputting of parameters in a situation where multiple samples with similar properties have been measured, and no intermediate adjustments to the measurement setup were required, so that all the samples have identical parameters. It is also possible to use a single parameter file for multiple samples, but, for the sake of clarity, it is advisable to use a unique parameter (P) and data (D) file for each sample.

5.7 The Main Program

The core of the software is the **Reconstruction** function, responsible for further processing of the data and the actual optimization resulting in the reconstructed pulse.

5.7.1 Parameters for the Main Program

The **Reconstruction** function must be provided with several parameters that guide its behaviour. The first set of parameters specify file paths and identifiers for output files. Their description and names are provided in Table 5.1. The location of the D and P files and a path where the results of the reconstruction are to be saved. The user must also provide a name for the optimization run, in the *scenario* variable. Several datasets may be reconstructed in succession, and each must have its own *scenario* name. There is also a *tag* which is included in every file. Typically one can use the scenario to include identification of the original data or the physical sample used in the measurement, and the tag to give a name for the simulation run, e.g., "testrun1".

Table 5.1: Parameters defining the location of the relevant files, where to save the data generated by the software, and identifiers for the output files

Parameter	Description
pathD	Location of the D file containing the preprocessed trace
pathP	Location of the P file containing the parameters of preprocessed trace
savePath	Path for the output files
scenario	Name of the sample, included in output filenames
tag	Additional identifier, included in output filenames

Second set of parameters control the optimization, and must be selected carefully. These are: *delayRange*, specifying the portion of the data along the delay axis to be used, *frequencyResolution*, which defines the number of distinct frequencies calculated for the trace (in the frequency range where the original trace resides), and *optimizationPoints*, giving the number of points the pulse is evaluated at. Typical values for these are given in Table 5.2. Additionally, the combination of sub-traces used in the optimization, and how large their weights are when the total FROG error defining the goodness of the fit is computed, can be selected with the parameters *bands* and *weights*, respectively. The values for both of these parameters are given in a vector, where the first element corresponds to the baseband, the second element to the FM-band, and the third element to the second-harmonic modulation band. The parameter *optimize* must be set to 1 for the optimization to start, otherwise the program only prints figures.

Table 5.2: Typical values for the three parameters that define the delay range and the resolution of simulation.

Parameter	Value	Unit	Description
delayRange	$60 \dots 100 \cdot 10^{-15}$	second	Specifies the length of the delay axis
frequencyResolution	$20 \dots 60$	–	Number of frequency samples in the main trace
optimizationPoints	$20 \dots 60$	–	Number of points the pulse is adjusted at
bands	[1/0; 1/0; 1/0]	–	Selects the sub-traces used in optimization
weights	$[0 \dots 1; 0 \dots 1; 0 \dots 1]$	–	Set the weights for the different sub-traces' FROG errors
optimize	1/0	–	Optimization on/off

5.7.2 Before the Simulation Starts

Since computation is carried out in frequencies instead of wavelengths, the measured data has to be interpolated to fit a new set of frequency and delay axes, where the former does not share the same spacing as the original wavelength axis. This is because when the wavelengths, originally sampled at equal intervals, are converted into frequencies, the spacing between two adjacent frequency points varies and is

in fact frequency dependent: $\Delta\lambda = \lambda_2 - \lambda_1 = c/\nu_2 - c/\nu_1 = c\Delta\nu/(\nu_1\nu_2)$, or $\Delta\nu = \Delta\lambda \cdot (\nu_1\nu_2/c)$. An uneven frequency sampling makes life with Fourier transforms along the frequency space difficult, so the data is interpolated to fit a set of new, evenly spaced frequencies. As was mentioned earlier in Subsection 5.6, the delay axis is also engineered to fit the needs of the program, and also requires interpolation of the data. Interpolation is, however, only carried out once for the measurement data, simultaneously for the new frequency and delay axes. This is done to prevent unnecessary loss of data. An internal MATLAB routine is used to perform the two dimensional interpolation with a cubic spline fitting procedure.

5.7.3 Operation During Simulation

Once the main program is started, a plot of the trace from the original data along with the calculated trace of the seed pulse appears with the Fourier filtered sub-traces of DC, fundamental modulation and second–harmonic modulation bands. After this the actual optimization begins, progress of which can be observed from the MATLAB command window displaying the weighed FROG error of each iteration and the corresponding action determined by the Nelder–Mead algorithm. In addition, once the algorithm has reached its maximum number of iterations set by the user, a figure with the plots of the original trace, its baseband, the current iteration of the pulse with its intensity and phase profiles and the subsequent fitted trace with its baseband is updated. This is the graphical user interface (GUI) of the main program. A screen capture of the GUI is presented in Figure 5.5.

After the GUI has been updated, the algorithm starts again and runs for the maximum number of iterations, unless the user has pressed one of the buttons on the figure. The *Stop* button terminates the algorithm loop and the program decides the simulation is successfully concluded, saving all the data and exits. Pressing the *Change number of iterations* button causes a pop up to appear, which prompts the user to provide a new maximum number of iterations for each algorithm run. The smaller the number, the more often the user gets a graphical update of the progress of the reconstruction, but every time this happens, the algorithm is prematurely terminated. This might slow the reconstruction process, so a large number of iterations, say a few thousand, is preferred if the user is not present to continuously monitor the progress. It should be stressed that it is impossible to immediately terminate the algorithm at any point without crashing the program. Only when the algorithm has reached its maximum iterations (or alternatively reaches a stopping condition such as a sufficiently small error of 10^{-8}) can the program act and respond to user input, i.e., the pressing of one of the two buttons in the figure.

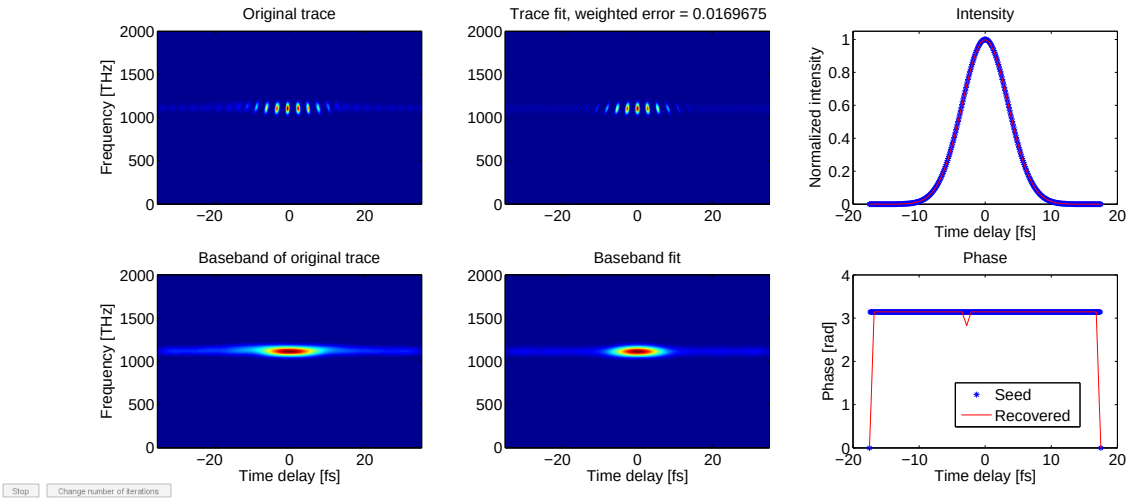


Figure 5.5: Screen capture of the graphical user interface of the Reconstruction program. The progress of the pulse retrieval can be monitored by comparing the measured trace and its baseband, displayed on the left hand side column, to the simulated trace and its baseband, located at the the center column. On the right hand side column, the intensity and phase envelopes of the seed pulse (blue diamonds) and the simulated pulse (red line) are presented. Note that this picture was taken at the very start of the simulation, so the pulse does not differ practically at all from the seed. In the lower left corner are the two buttons that can be used to either terminate the pulse retrieval (Stop) or change the number of iterations the algorithm runs before updating the GUI again (Change number of iterations). The buttons may be pressed at any time, but the program will only respond when the algorithm has reached its maximum number of iterations and the GUI is to be updated once more.

6. SIMULATIONS AND RESULTS

Two different datasets were provided by Max Born Institute from the same experimental setup: one for a sample of silica thin film and a second for a sample of titania thin film, as was presented in Figure 4.2. What is clearly visible in these pictures—even to the naked eye—is the asymmetry of the titania trace along the delay axis and, in contrast, the symmetry of the silica trace. The titania trace is also much wider in the sense that the high intensity center of the trace reaches larger delays in comparison to the silica trace. These observations imply the presence of a noninstantaneous response time of the third-order optical nonlinearity in titania. The situation is similar to that what Anderson *et al.* faced in their plasmon tip response time measurements. By reconstructing the pulses of both, the silica and the titania samples, it is possible to estimate the lifetime of the nonlinearity in titania using a convolution strategy.

6.1 Preparations

Before the simulations can be executed, the experimental data must be prepared for the program following the procedures described in Section 5.6. Several artifacts, circled in magenta in Figure 6.1, were removed, the background level was adjusted and the intensity was normalized. Suitable parameters for the optimization sequence were found by trial and error. These were: delayRange = 80 fs, optimizationPoints = 51 and frequencyResolution = 32. The time limit for each of the six simulations was 24 hours, which was also the realized duration for every simulation, i.e., no other stopping criteria like sufficiently small FROG error was met before the time limit was reached.

The intensity maximum of the baseband for the measured trace was used to define the central frequency ν_c of the trace. These were $\nu_c^{\mathrm{SiO_2}} \approx 1110$ THz for silica and $\nu_c^{\mathrm{TiO_2}} \approx 1096$ THz for titania, corresponding to carrier wavelength of $\lambda_0^{\mathrm{SiO_2}} \approx 810$ nm and $\lambda_0^{\mathrm{TiO_2}} \approx 821$ nm, respectively. The (angular) carrier frequencies ω_0 used by the program are calculated from the central frequencies with the relation $\omega_0 = 2\pi \cdot \nu_c/3$, as ν_c is the third-harmonic frequency, i.e., three times the carrier frequency.

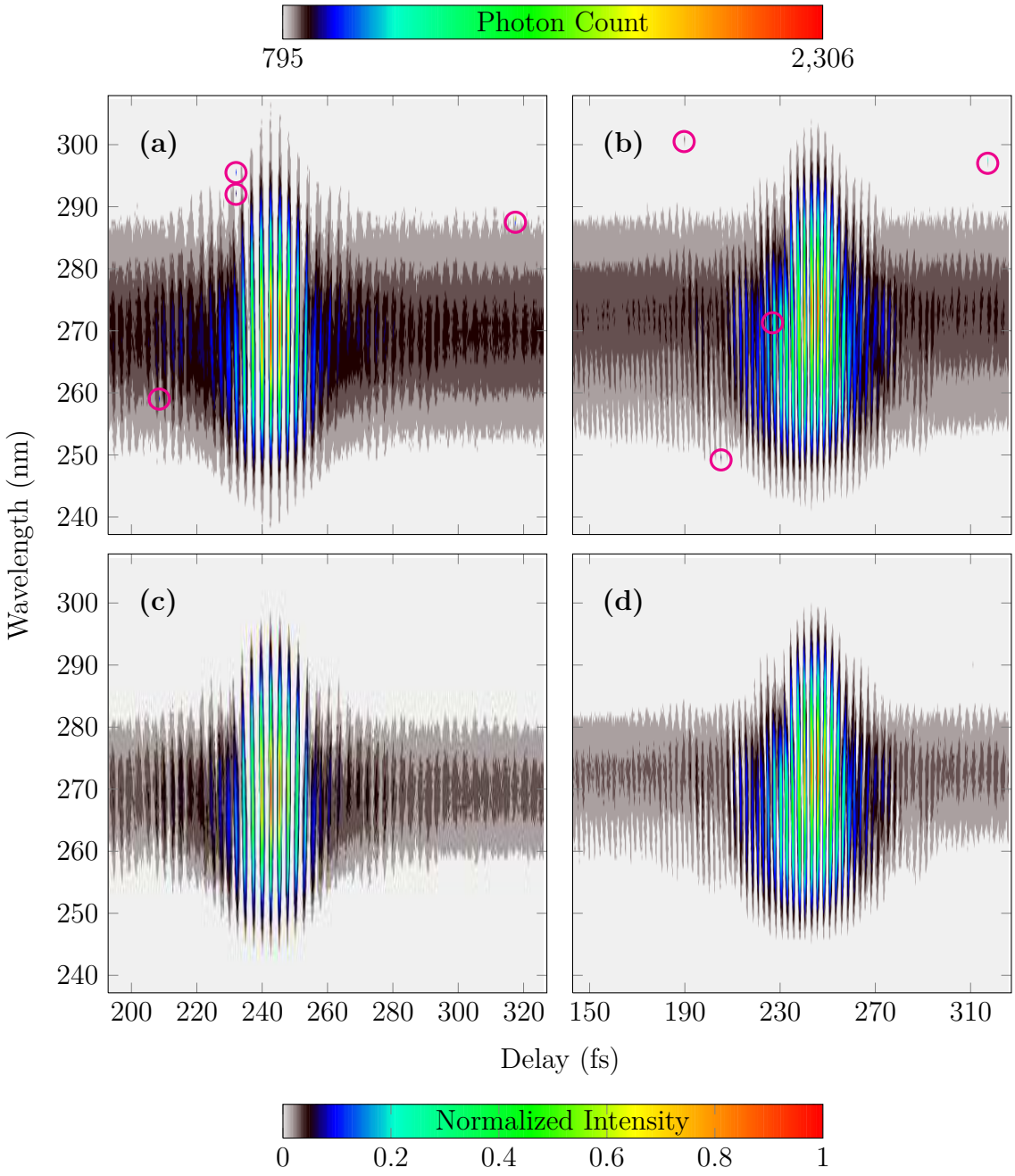


Figure 6.1: Preparing of the measurement data for the Reconstruction program. On top are the unaltered THIFROG traces for (a) silica and (b) titania, while the altered traces for (c) silica and (d) titania are found below. The scale for (a) and (b) is found above the traces, while the scale for (c) and (d) is located at the bottom of the figure. All the above pictures are close-ups of the relevant wavelength range containing the THIFROG signal. Measured data at wavelengths not shown here was discarded. Several artifacts were removed from the two traces (a) and (b), where the data points to be levelled are circled in magenta. Background noise was removed, and the zero intensity level was adjusted so that the background signal on both sides of the principal trace is damped. The effect of this procedure can be seen by comparing the black portions of the unaltered traces on top, and the altered traces below. The intensity was normalized to 1 for (c) and (d).

To avoid confusion, it should be noted that from here on the term "measured trace" is not used for the measured experimental data *per se*, but for the experimental data which has undergone the above mentioned preparations, and the processing of the main program, i.e., interpolation to fit the tailored delay and frequency axes.

Different combinations of the baseband and the fundamental modulation band were used in the reconstructions to calculate the FROG error, which serves as the measure for the goodness of the fit. This means that the program tries to make the sub-bands used in the reconstruction to match the respective sub-bands of the measured trace as closely as possible. In total there are six reconstructions: three for both silica and titania using 1) only baseband, 2) only FM-band, and 3) both baseband and FM-band, for the simulation. These are referred to as simulations 1, 2 and 3, respectively, for either of the two samples. For example, simulation 2 for silica refers to the reconstruction of the silica pulse where the FROG error was calculated from only the FM-band. The measured trace and its sub-traces are referred to with the index 0 in the labelling found in the pictures. The results for the simulations are presented in the following subsections, starting with the reconstructed traces which are followed by the reconstructed pulses.

6.2 Reconstructed Traces

The measured traces, and the reconstructed traces for the three simulations along with their respective basebands and FM-bands are illustrated in Figures 6.2 for silica, and 6.3 for titania. As the simulations used the sub-traces for the fitting procedure instead of the actual trace, most of the discussion here concentrates to evaluate how well the simulated sub-traces fit the sub-traces of measured trace.

The high degree of symmetry for the silica trace (Figure 6.2 (T0), upper left corner) is also evident in its baseband (B0) and FM-band (FM0), where the only significant difference between the two sides of the sub-traces is found at delays $\tau \approx \pm 15$ fs, while the central part of the measured intensity at $\tau \approx -10 \ldots 10$ fs is almost perfectly symmetric. Simulation 1 for silica produced a baseband (B1) which is in good agreement with the measured baseband (B0), but the reconstructed FM-band (FM1) displays symmetric satellite structures at $\tau \approx \pm 20$ fs which are not present in (FM0). Simulation 2 corrected this flaw in the FM-band, although the reconstructed sub-trace (FM2) does not display the small asymmetry of (FM0). This time the baseband (B2) is a less convincing fit for (B0). The overall best agreement with measurement data was achieved—perhaps unsurprisingly—with simulation 3, using both sub-traces for the reconstruction. The simulated baseband (B3) is very similar to the measured one (B0), as is the FM-band (FM3) in comparison to (FM0).

The trace (T3) itself looks convincing as well, although the original (T0) shows finite intensity in a broader area. Retrospectively, this could have been avoided by stronger background deduction for the measured data. Even though the overall agreement for the sub-traces is excellent, each of the three simulations produced almost perfectly symmetric traces, and were unable to account for the slight asymmetry in the original data.

In contrast to the symmetry of the measured silica trace, the measured titania trace in Figure 6.3 (T0) and its sub-traces (B0) and (FM0) display much greater differences in intensity between the two sides of the central maximum. This asymmetry is especially pronounced in the fundamental modulation band (FM0), where significant signal is measured at delays $\tau \approx -30,\ -18$ fs, while very little intensity is seen on the corresponding positive delays. Unlike with silica, where the three simulations produced somewhat different results, the titania reconstructions are strikingly similar to each other. This similarity is also observed in the reconstructed pulses for titania, presented in the next subsection. As was the case with silica, the reconstructed traces (T1–3) and their sub-traces (B1–3, FM1–3) for titania are perfectly symmetric. This results in poor agreement especially in the FM-bands (FM1–3), where a wedge-like shape has been formed. This is most probably a consequence of the wing structure of (FM0) at $\tau \approx -18$ fs. The basebands (B1–3) seem to match the original sub-trace (B0) somewhat better.

Another noticeable feature for both samples is that the measured traces (T0) in Figures 6.2 and 6.3 is *not* perfectly aligned with the zero delay while the reconstructed traces (T1–3) are. The implication of this fact is that while the sub-traces are reconstructed with some success, the actual trace is not, at least not without realignment. The measured traces for both of the samples, however, contain quite a bit of noise, so even if the traces were perfectly aligned the agreement could not be perfect, even if the *true* pulse shape was used to create the simulated trace. The sub-traces, on the other hand, are filtered from most of the noise, therefore making their reconstruction a more reliable process.

Even though the sub-traces were in good agreement, this misalignment of the THIFROG traces suggests that the use of the baseband intensity maximum of the measured trace as the point of zero delay is not perfectly justified. A better method for defining the zero delay would align the centres of the two traces more adequately and therefore reduce the overall FROG error. One such method could be to locate the center fringe of the measured trace with the intensity maximum of the trace, as the center fringe is likely to contain this, and fit a polynomial curve over the fringe. The maximum of the polynomial could then be used as the true location of

the intensity peak and therefore as the zero delay point. This method should not be difficult to implement, making it a plausible improvement if the Reconstruction program is to be developed further. On the other hand, the fact the basebands are perfectly aligned can be considered to be more important than the alignment of the actual traces when the success of the reconstruction relies on the agreement of the basebands. Moreover, one could argue that whether the trace is perfectly reconstructed or not is not that important when the *sub-traces are*, if only a rough estimate for the pulse envelope or width is sought. The above described method for improving the alignment of the trace could also be applied for the baseband alignment, which currently works by simply picking the data point with the highest intensity. The fitting of a parabolic function over the baseband peak would help to pinpoint the location of the actual maximum, not just that the data point nearest to it.

Judging by the reconstructed silica traces in Figure 6.2, the most reliable combination of sub-traces for the reconstruction is using both, the baseband and the FM-band. The use of the remaining two harmonic bands modulated at frequencies $2\omega_0$ and $3\omega_0$ was not investigated, but their use could potentially produce even more reliable results. Although speed was not a priority when the program was being coded, faster convergence could be achieved by using different weights for the FROG errors of the sub-traces, and by experimenting with the precision of the simulation to find a suitable balance between speed and resolution.

To summarize, the reconstruction succeeded reasonably well. Simulated sub-traces for silica were in good agreement with the measurement data, as was the titania sub-traces—aside from the asymmetry. The reconstructed traces were almost perfectly symmetric for reasons unknown. One possibility that the beam splitter power division ratio is not a perfect $1:1$, even though this was assumed in these simulations. This would affect the symmetry of the trace in ways the program could not have accounted for with the settings used. Moreover, it is possible that selecting a wider delay range, and perhaps a more complex seed pulse might eventually result in a better fitting, asymmetric trace.

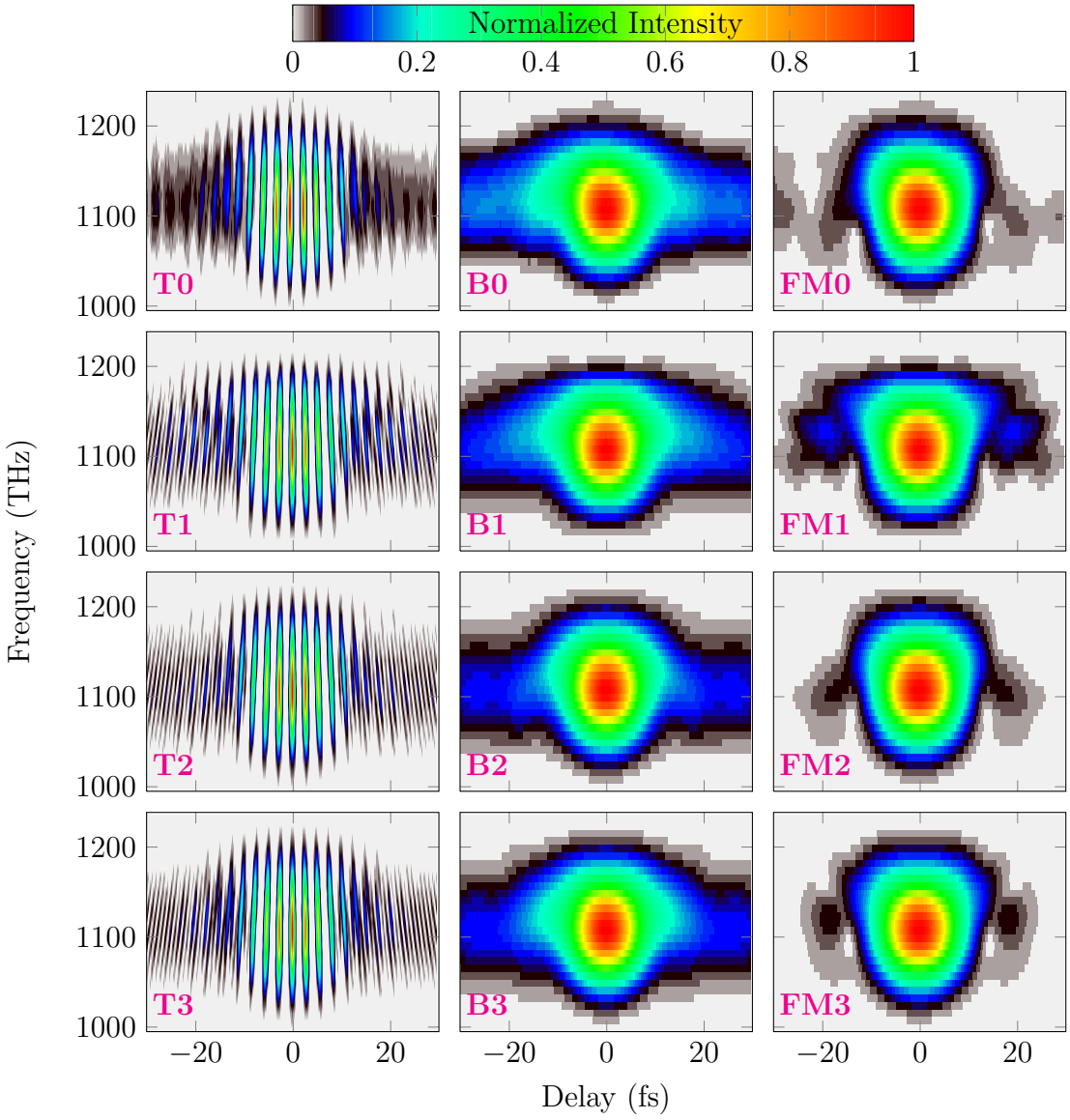


Figure 6.2: Measured and reconstructed traces and their sub-traces for silica. The left hand row contains the THIFROG traces, labelled with T. A zero in the label stands for the measurement data, and numbers 1, 2 and 3 are used to denote reconstructions using 1) baseband, 2) FM-band, and 3) both baseband and FM-band, respectively. The middle row is for the basebands, labelled with B, and on the right hand row are the FM-bands, labelled with FM, of the respective traces on their left. For example, (T0) is the measured trace for silica and (B0) is its baseband, and (FM1) is the FM-band of the reconstructed trace (T1), where the FROG error of the baseband was used as the measure for the goodness of the fit. These images are close-ups of the data, the full computation range for these simulations being frequency $\nu = 0 \ldots 2000$ THz and delay $\tau = -40 \ldots 40$ fs.

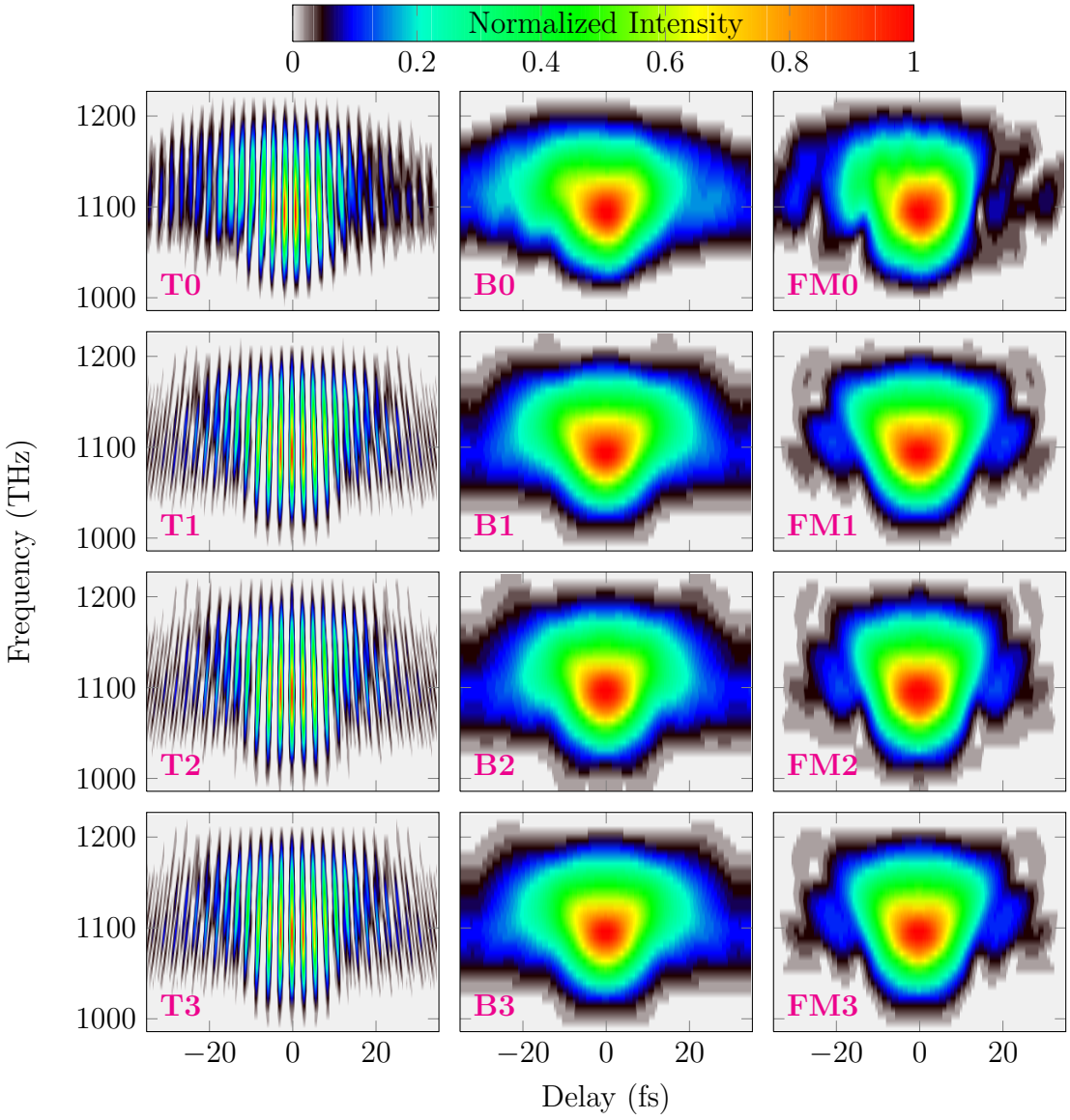


Figure 6.3: Measured and reconstructed traces and their sub-traces for titania. The traces and sub-traces are ordered, displayed and labelled in the same manner as the silica traces in Figure 6.2. Note that a wider range of delays ($\delta = -35 \ldots 35$ fs) is displayed here in comparison to what was displayed with the silica traces ($\delta = -30 \ldots 30$ fs). Therefore, the noticeably wider traces and sub-traces of titania are in fact even wider in comparison to those of silica than they appear here. This translates to a substantially wider pulse envelope for titania, as is evident in Figures 6.4 and 6.6.

6.3 Reconstructed Pulses

The reconstructed pulse intensities and phases for the silica and titania simulations are illustrated in Figure 6.4. Pulses (a–c) are from the silica simulations 1–3, respectively, and pulses (e–g) are likewise for the titania simulations 1–3, respectively. Each of these six reconstructed pulses display a pronounced central peak in the intensity, followed by a trail of residual intensity on one side and a sharp drop on the other side. The smaller the intensity gets, the more fluctuations in both intensity and phase is observed. The phases are seen to be curved one way or the other around the centers of the intensity peaks and evolve quite steadily in regions where significant intensity is present. The phase behaviour becomes erratic in regions of high intensity oscillations, and phase shifts for over 2π are observed, e.g., in (f) at $t \approx 17$ fs. Only the phase in the peak intensity region is considered to bear any physical significance, and the large phase shifts are regarded as irrelevant. Likewise, the highly fluctuating intensities are credited to random noise in the simulations, and only the general trend of the intensity, and the peak itself are considered to contain meaningful information.

6.3.1 Composite Pulses

Since the three simulations for each of the samples aim to reproduce the same pulse, the three reconstructed pulses are used to assemble a single, representative pulse for the sample, henceforth referred to as the *composite pulse*. The procedure used to combine the reconstructed pulses is described next.

Because the reconstructed traces were symmetric in respect to delay, it is impossible to state which side of the reconstructed pulse represents the front and which side the back of the physical pulse. Physically, it makes more sense that the trail follows the pulse and not the other way around, as such a trail behind a pulse can be caused by noninstantaneous response of the medium. It is therefore convenient to agree to a convention, that the time axis is normally oriented in the sense that given two events taking place at t_1 and $t_2 > t_1$, the event at t_1 happens before the event at t_2. The ramifications of this is that the negative time coordinates should represent the sharply rising front of the pulse, and the intensity trail following the pulse must reside in the positive time coordinates. Thus some of the reconstructed pulses must be reversed in time before they are used to make the composite pulses. The time-reversal operation is executed for the silica pulse of Figure 6.4b, and for all three of the titania pulses (e–g). After the time axes of the pulses agree to the same convention, the approximate centers of the three peaks for each sample are aligned

to $t = 0$. Once the peaks coincide, the mean of the intensities and the phases of the three pulses are taken. The resulting single pulse for each sample is then smoothed by averaging each intensity and phase sample in its neighbourhood, after which the intensity is once again centered to $t = 0$, and normalized to 1. These steps yield the composite pulses, shown in Figure 6.4 for silica (d) and titania (h).

6.3.2 Group Delay Dispersion Analysis

To study the retrieved pulses further, the two composite pulses were Fourier transformed to the spectral domain, where the spectral phases $\varphi(\omega)$ were used to calculate the *group delay dispersion* (GDD) for both silica and titania. These are presented in Figure 6.5. GDD or *second-order dispersion* $\varphi(\omega_L)''$ is defined as the second derivative of the spectral phase evaluated at the center angular frequency ω_L of the laser [95]

$$\varphi(\omega_L)'' \equiv \left.\frac{\partial^2 \varphi}{\partial \omega^2}\right|_{\omega_L} . \tag{6.1}$$

By calculating the weighted integral

$$\phi_2 = \frac{\int_{-\infty}^{+\infty} \varphi(\omega)'' \, I(\omega) \, \mathrm{d}\omega}{\int_{-\infty}^{+\infty} I(\omega) \, \mathrm{d}\omega} , \tag{6.2}$$

an estimate for the total group delay dispersion ϕ_2 experienced by the pulse is obtained. These values were $\phi_2^{\mathrm{SiO2}} = -3.9$ fs^2 for silica, and $\phi_2^{\mathrm{TiO2}} = -14.1$ fs^2 for titania, indicating the presence of anomalous dispersion, i.e., $\varphi(\omega_L)'' < 0$. This is indicative of a slight overcompensation of GDD effects by *chirped mirrors* [96] used in the experimental setup. Chirped mirrors can only compensate dispersion in steps of about 30–60 fs^2, making the calculated values for ϕ_2^{SiO2} and ϕ_2^{TiO2} small in comparison.

Negative GDD is, however, unexpected as the setup was carefully constructed to negate the normal dispersion, i.e., $\varphi(\omega_L)'' > 0$, caused by the beam splitter of the Michelson interferometer, and to avoid overcompensation. Another viable option that must be considered is that the MATLAB function for FFT uses a different sign convention than was expected during programming. A change of sign in the exponential of the Fourier transform $F(\omega) = \int_{-\infty}^{+\infty} f(t) \exp(-i\omega t) \, \mathrm{d}t$ would reverse the phase, leading to normal dispersion in our case, thus making the results sensible. Assuming that the above calculated GDD values were correct in magnitude but erroneous in sign, the corresponding thickness of silica in the setup would be 110 μm,

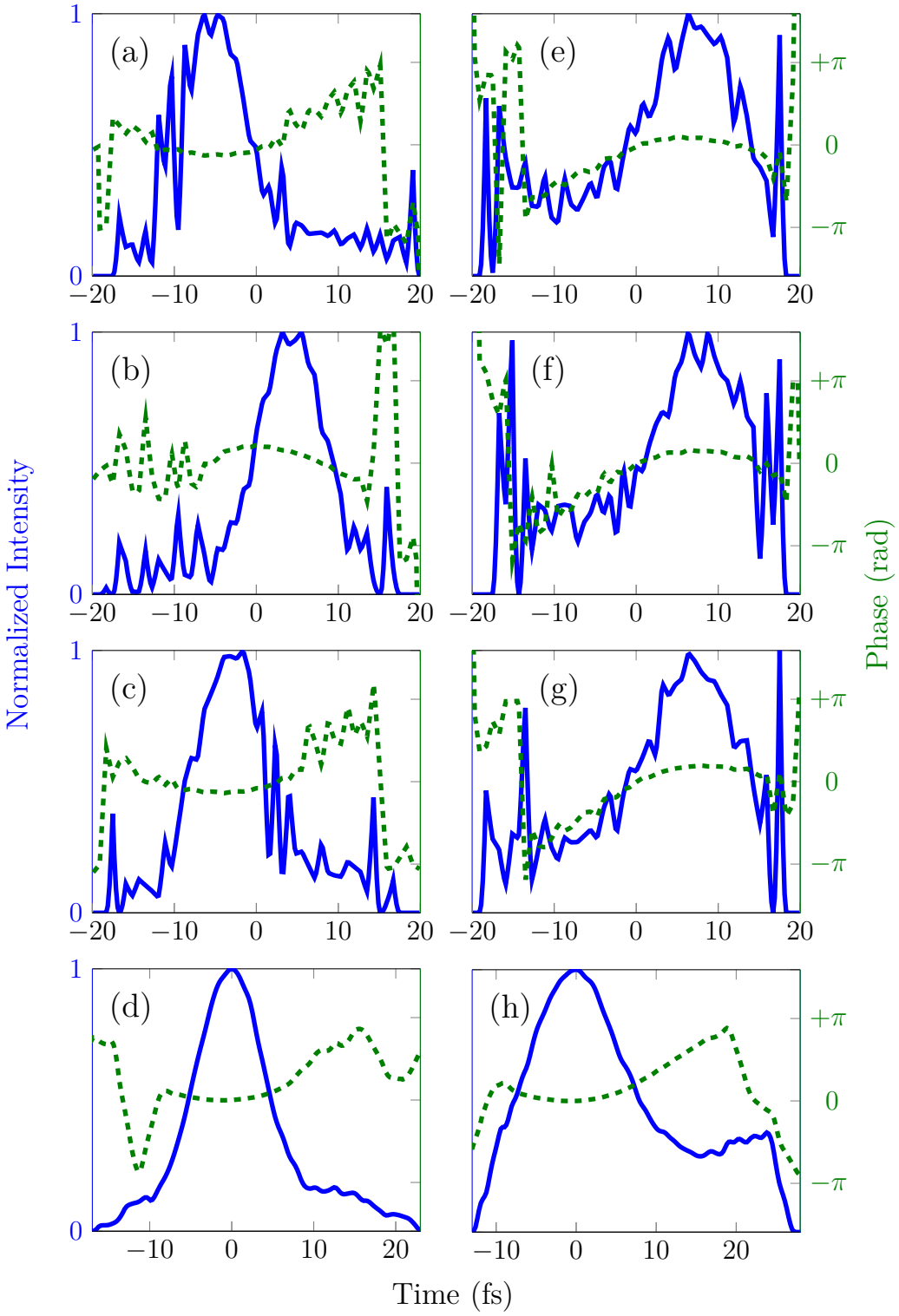


Figure 6.4: Reconstructed pulse intensities (blue line) and phases (dashed green line) for silica (a–c), and titania (e–g). The three reconstructed pulses for each sample were aligned and averaged to build composite pulses for silica (d) and titania (h). Pulses (b) for silica and (e–g) for titania were also subjected to time-reversal so that the intensity "trail" would be located behind the pulse, i.e., on positive time coordinates. Note that the time-reversal operation inverts the phase. This is evident if one compares the silica pulse (b) to the other two pulses (a) and (c), where the curvature of the phase in the vicinity of the intensity peak in (b) is opposite to the phase curvatures of the other two pulses. The composite pulses (d) and (h) were also smoothed by averaging each sample in its neighbourhood.

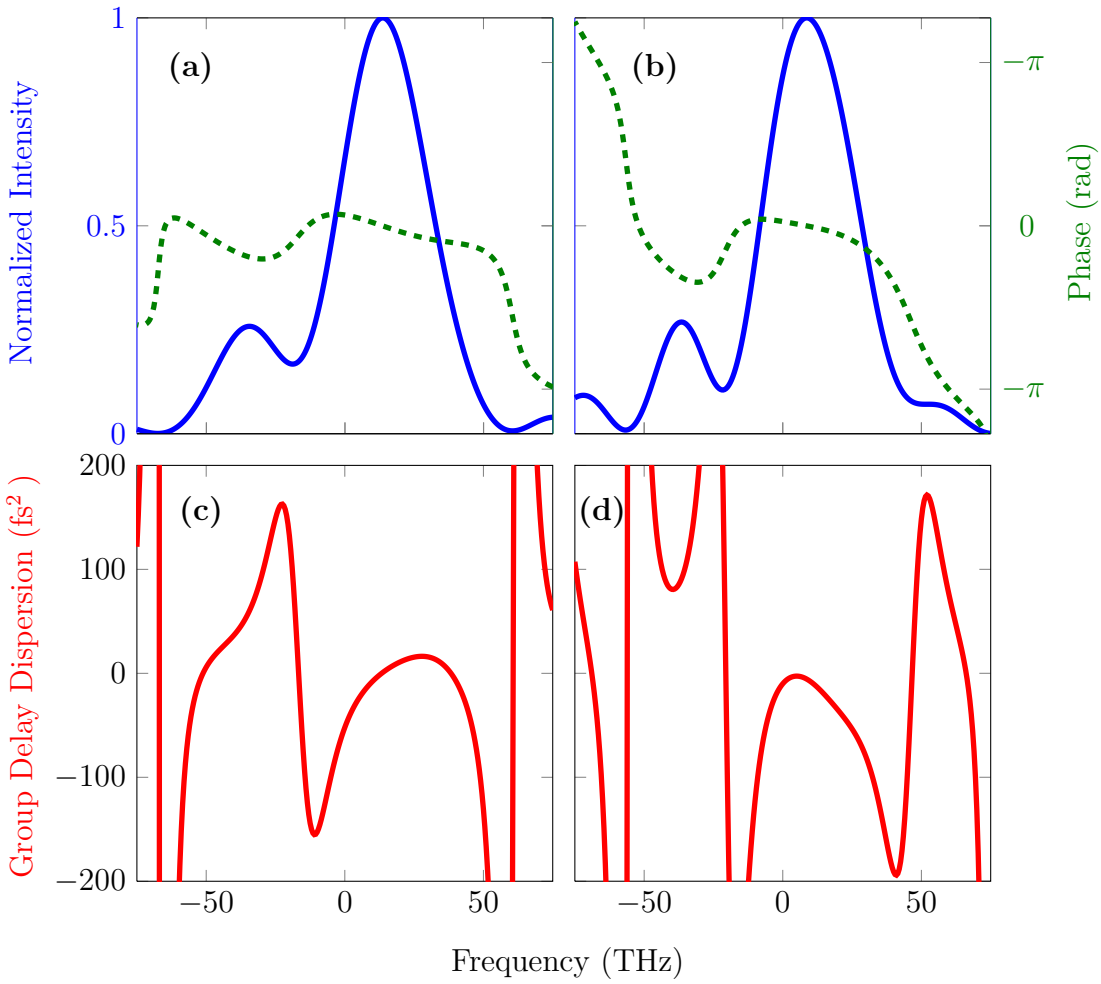


Figure 6.5: Top row: spectral intensities (blue line) and phases (green dashed line) for the composite pulses of (a) silica and (b) titania, presented in Figure 6.4 (d) and (h), respectively. Bottom row: Group delay dispersion (red line) of the same pulses for (c) silica and (d) titania. The frequency axes represent the difference in frequency from the carrier angular frequency ω_0. The apparent roundness of the curves is a consequence of the finite resolution of the simulations.

as silica has a *group velocity dispersion* (GVD) of $\beta_2 = +36$ fs^2/mm at 800 nm. The GVD β_2 is related to the GDD $\varphi(\omega_L)''$ by $\varphi(\omega_L)'' = \beta_2 L$, where L is the length of the medium. The GVD was calculated from the relation [18]

$$\beta_2 \approx -\frac{\lambda^3}{2\pi c^2}\frac{\mathrm{d}^2 n(\lambda)}{\mathrm{d}\lambda^2}\,, \tag{6.3}$$

where the wavelength dependent refractive index $n(\lambda)$ was obtained from the Sellmeier dispersion equation for silica [97]. The 110 μm corresponding thickness is a very small figure and the pulse surely traverses more silica in the experimental setup, as even the glass substrate is 1 mm thick. This simply shows the effectiveness of dispersion compensation in the setup.

The FWHM pulse durations of the composite pulses (d) and (h) in Figure 6.4 are 10.1 fs for silica and 15.7 fs for titania. As the laser system was specified to deliver sub-8 fs pulses, these values indicate temporal broadening of the pulses. The one and a half time pulse length of titania in comparison to the silica pulse cannot be attributed to the measured GDD alone. This substantial difference between the two samples suggests the presence of a finite lifetime nonlinear effect in titania.

6.4 Lifetime of $\chi^{(3)}$ Nonlinearity in Titania Thin Film

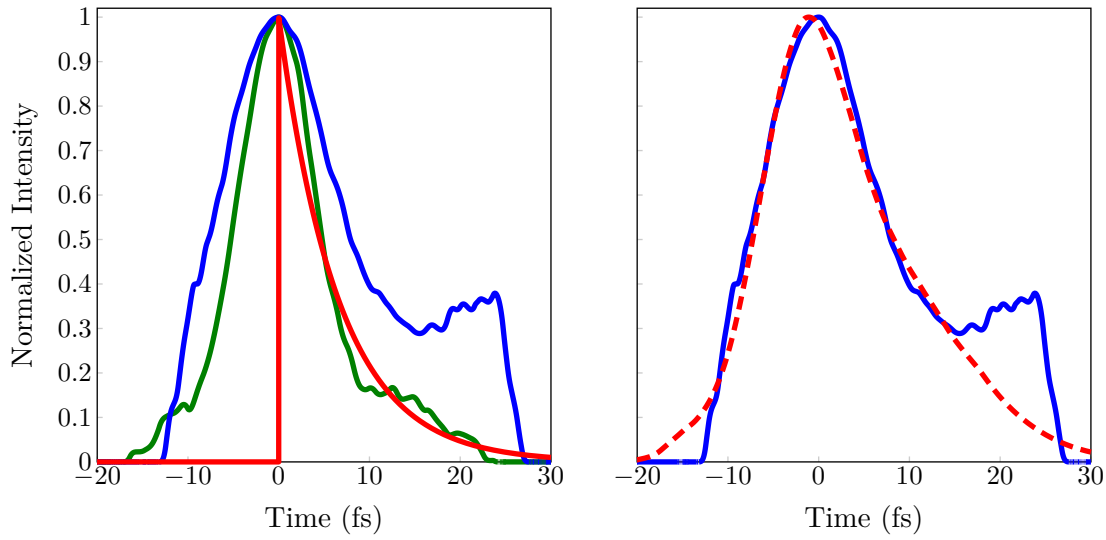
In order to estimate the lifetime of the nonlinear polarization induced by a laser pulse passing through the titania thin film, a simple deconvolution method is used to simulate material response of the highly nonlinear titania in comparison to the less reactive silica sample. By assuming that the third-order polarization of silica is a *near* instantaneous process, the reconstructed composite pulse for silica may be used as a neutral reference for the signal function $S(t)$ in the convolution

$$(S * R)(t) \equiv \int_{-\infty}^{+\infty} S(t)R(t-\tau)\,\mathrm{d}\tau = H(t)\,. \tag{6.4}$$

Here $R(t)$ is the response function representing the titania thin film's response to the presence of electric field of the pulse $S(t)$, and $H(t)$ is the pulse after interacting with titania, i.e., the reconstructed composite pulse for titania. In essence, the convolution $(S*R)(t)$ represents the smearing effect the noninstantaneous nonlinear polarization in titania imposes to the third-harmonic field it creates. Ideally, $R(t)$ could be obtained from the two composite pulses through numerical deconvolution, but this method is extremely sensitive to noise, and might give nonsensical results even in the absence of noise [91]. Instead, the nonlinear polarization is assumed to decay exponentially, so the response function may be defined as

$$R(t) = \begin{cases} e^{-t/T} & \text{if } t \geq 0 \\ 0 & \text{if } t < 0 \end{cases}, \tag{6.5}$$

where T is the time constant for the decay, i.e., the lifetime of the polarization. Replacing the signal function $S(t)$ in Equation 6.4 with the intensity of the composite silica pulse $I_{\mathrm{SiO_2}}(t)$, the convolution $(R * I_{\mathrm{SiO_2}})(t)$ is calculated and adjusted by tuning the time constant T and a time shift s for the outcome of the convolution $(H(t) \rightarrow H(t-s)\ )$ with the goal of reproducing the composite pulse for titania $I_{\mathrm{TiO_2}}(t)$ as closely as possible. The best agreement was achieved with the values $T = 6.5$ fs and $s = -4.8$ fs. The results are presented in Figure 6.6.



(a) SiO_2, TiO_2 and response function (b) TiO_2 and convolution

Figure 6.6: Estimation of the lifetime of the Kerr nonlinearity. The intensities of the composite pulses for both silica (green line) and titania (blue line) are shown alongside the response function (red line) in (a). The chosen response function is an exponential decay function with the time constant T. The convolution $(R * I_{SiO_2})(t)$ is calculated and compared to $I_{TiO_2}(t)$. The time constant T is adjusted and the resulting convolution is shifted in time to better match the retrieved pulse intensity for titania. The best agreement was obtained with the time constant $T = 6.5$ fs, the resulting convolution (dashed red line) is shown with $I_{TiO_2}(t)$ (blue line) in (b).

The convolution is in good agreement with the bulk of the titania pulse intensity, but in the range $t = 15 \ldots 25$ fs the intensity rises abruptly and the convolution is unable to account for this behaviour. This rise of intensity is due the averaged intensities of the three recovered pulses using different sub-traces for the reconstruction. As is evident from Figure 6.4e–g, all three pulses exhibited heavy oscillation of intensity in this region, which is attributed to random noise—present in regions of declining intensity such as the edges of the pulse—and is unlikely to bear any physical significance. For this reason the apparently poor agreement of the convolution and the reconstructed pulse around $t = 20$ fs is considered to be irrelevant, and the overall agreement to be excellent.

7. SUMMARY, DISCUSSION AND CONCLUSION

In this final chapter, a summary for the topics covered in this work is given. After discussing the topics, the concluding remarks are made.

7.1 Summary

A novel pulse retrieval software for the recently introduced THIFROG technique was presented, and used to successfully conduct the first ever pulse retrieval for a THIFROG trace. The retrieved pulses were harnessed to study an ultrafast nonlinear polarization process in a pure titania thin film. The astonishingly short lifetime of the $\chi^{(3)}$ nonlinearity in a dielectric was measured for the first time in history.

The fundamental theory for optics, along with a summary of nonlinear effects, and an introduction to laser pulses was given in the second chapter. After listing several technologies for creating ultrashort pulses, the discussion moved to pulse characterization in Chapter 3. Starting from the autocorrelator, several competing characterization techniques were introduced. The most attention was given to frequency-resolved optical gating and its many variants. The properties and theory of this complete characterization method were discussed in detail, while gradual progression towards the FROG variant, which gave its name for the title of this thesis, was made.

The recently introduced characterization technique for ultrashort laser pulses, third-harmonic interferometric FROG, was presented in Chapter 4. An analytical study on the structure of the THIFROG trace was carried out, and an equation explicitly stating the modulational components of the trace was derived and discussed. Measured THIFROG traces for a pair of thin film samples, one of pure silica and the second of pure titania, were presented.

A pulse retrieval software designed specifically for THIFROG was constructed, presented and discussed in detail in Chapter 5. The software, the first of its kind, was used to analyze the traces of the silica and titania thin films. Three independent retrievals were successfully executed on both of the samples, and the reconstructed

pulses were used to compile a pair of representative pulses, presented in Chapter 6. To investigate these composite pulses further, their group delay dispersion was calculated and discussed. The reconstructed pulses were subsequently used to evaluate the lifetime of the nonlinear polarization observed in titania using a deconvolution strategy. The noninstantaneous response of titania was simulated by subjecting the reconstructed pulse for silica to an exponential decay function with the goal of replicating the titania pulse as closely as possible. A time constant of 6.5 fs for the decay was measured, and found to produce excellent agreement with the data.

7.2 Discussion

The exact mechanism behind the finite lifetime of the nonlinearity in titania is unknown. The main reason for the different responses between titania and silica can, however, be attributed to the large difference in the band gaps of the two media. In order to excite valence band electrons to the conduction band, photons of less than 140 nm in wavelength are required to overcome the roughly 9 eV band gap in silica, while approximately 390 nm photons are energetic enough to be absorbed by the 3.2 eV band gap of the titania thin film. While the photons emitted by the laser are in the range of 650–950 nm (see Figure 4.3a) and alone too weak to excite electrons in either medium, three-photon absorption and even two-photon absorption can still take place in titania. Both effects have recently been observed in bulk titania for wavelengths around 800 nm [98]. Silica, on the other hand, is far from resonance and should experience negligible absorption. Once electrons are excited to higher states, recombination of carriers and emission of photons will follow. This alone can cause an observable effect as the newly created photons will reach the detector after some delay, since the recombination process is *not* instantaneous. It is possible that the measured lifetime is simply indicative of this two- and three-photon absorption induced recombination process, but without further evidence this remains but a speculative remark.

It is a well known fact that the presence of free carriers alters the nonlinear refractive index of an optical medium, as predicted by the Drude model [19, 99]. Recently, remarkably efficient generation of THz radiation has been achieved by focusing Ti:sapphire pulses at the fundamental and second-harmonic frequencies into a gaseous medium, such as ambient air [100–102]. The intense radiation forms a plasma, which interacts nonlinearly with the laser pulses to create THz radiation through the $\chi^{(3)}$ process of four-wave mixing rectification, according to Xie *et al.* [102]. It has been suggested, that the efficiency of the process is caused by an enhancement of hyperpolarizability due to the plasma [100]. This was confirmed

with the observation of greatly enhanced $\chi^{(3)}$ susceptibility in a plasma formed in air by Xie *et al.* It is very likely that the free carriers present in the titania thin film alter its optical properties in a similar manner to what has been observed in plasmas—the two are very similar, after all, as plasma consists of *almost* free electrons amidst positive ions. An increase of $\chi^{(3)}$ susceptibility due to the free carriers created by two- and three-photon absorption in the titania sample would explain the observed asymmetry in the measured THIFROG trace. This is because the front of the pulse would experience the unaltered $\chi^{(3)}$ of titania while creating free carriers in its wake, so that the rest of the pulse would experience an enhanced $\chi^{(3)}$ due to the free carriers already present. Since THG is directly proportional to $\chi^{(3)}$, the latter part of the pulse would create a more intense THG signal, so that the central part of the THIFROG trace would display higher intensities on larger delays—exactly what is observed for titania. The only discrepancy here is that the satellite structure of intensity in the titania trace is found at smaller delays, i.e., if the above explanation is true, the response is observed *before* the pulse enters the medium! This violates causality and is simply impossible.

The experimental data was provided along with screen captures of the measurement software, and the delay and wavelength axis information for the simulations was obtained from these. It was apparent from the pictures, that the THIFROG trace was upside down—the fringes were skewed in the wrong direction, and the trace center was nowhere near the third-harmonic frequency of the laser. This was eventually corrected by turning the wavelength axis around. It is impossible to say whether the same mistake was made with the delay axis just by inspecting the screen captures. If so, the asymmetric shape of the titania trace would be turned around so that the satellite structure originally observed on negative delays in the simulations would be shifted to the positive delays instead. This solves the problem of causality above. The reconstructed pulses would be turned around as well, justifying the time-reversal operation executed for all of the reconstructed titania pulses even further. As the silica trace was quite symmetric with respect to delay, the effect of reversing the delay axis would in this case be of little consequence. These facts considered, it is concluded that the delay axis was almost certainly falsely calibrated at Max Born Institute.

The pulse retrieval software was able to reconstruct the silica trace well, but the asymmetry observed in the titania trace was not reproduced. For reasons unknown, all six of the retrievals produced almost perfectly symmetric traces. The information that was obtained from the simulations was nevertheless perfectly sufficient for their ultimate purpose, that was the $\chi^{(3)}$ nonlinearity lifetime measurement. The inability of the Reconstruction program to reproduce the asymmetry of the measured

THIFROG traces constitutes further investigation into the matter. One possibility would be to measure the spectral response of the beam splitter. If the device displays an imperfect power division ratio at certain wavelengths, this would have an effect in the trace that the Reconstruction program cannot account for. If the ratio is uniformly unbalanced for the entire spectrum, the power division factor of the program can be adjusted. By measuring the power division ratio as a function of wavelength, this information could be taken into account by slightly modifying the software.

The next experimental direction for the lifetime measurements would be to fabricate new thin film samples with a variable mix of titania and silica instead of using samples consisting of only a single substance. A composite thin film would allow the effective band gap of the sample to be tuned. This can be accomplished with the rugate technique that was already used for the samples presented here. Measurement of THIFROG traces, and a subsequent pulse retrieval with the Reconstruction program for a set of samples with a transition from silica to titania in composition would provide an insight to the band gap dependency of the nonlinearity lifetime. This knowledge could be used to narrow down the list of possible mechanisms behind the finite lifetime.

Several advances were made in this work. The first pulse retrieval software for THIFROG was introduced, and the first reconstructions for THIFROG traces were executed with success. The finite lifetime of the $\chi^{(3)}$ nonlinearity in a dielectric was conducted for the first time. In conclusion, the goals laid out for this thesis were met, and several contributions to the fields of pulse characterization and nonlinear optics were made. Regardless of origin, a process with a time scale in the femtoseconds is surely one of the fastest ever measured. It is now up to theoreticians to summon an explanation for this phenomena.

7.3 Conclusion

Pulse characterization is a timely topic in ultrafast laser physics. Despite of the various methods that have already been demonstrated, there is still room for improvement, in particular when it comes to pulses in the few-cycle regime, i.e., pulses that encompass less than 10 femtoseconds in their full width at half maximum.

The main task of this thesis was the interpretation of third-harmonic interferometric FROG traces. This method constitutes a newly demonstrated collinear variant of FROG that has not been discussed much in literature. In particular, there is no method available to reconstruct the underlying pulse shape that gave rise to the

measured THIFROG trace.

To this end, this thesis first analyzed the physics behind this particular pulse characterization method and decomposed the trace into a set of four sub-traces. Moreover, a retrieval software was developed that allows retrieval of the underlying pulse shape in these measurements. The software allows for a high degree of flexibility and enables reconstructions from various sub-traces or combinations thereof.

This software was tested on a set of measured THIFROG traces from the Max Born Institute. One half of these samples was measured with SiO_2 as the medium, the other with TiO_2. Retrieving the former yielded a nearly symmetric pulse shape with 10.1 fs pulse duration whereas the latter resulted in asymmetric shape with 15.7 fs. This marked difference with identical pulses but different nonlinear media could be perfectly explained by convolving the retrieved SiO_2 pulse with a single-sided exponential of 6.5 fs time constant.

While these results clearly indicate a substantial lifetime of TiO_2 in this specific resonant excitation condition, this process is among the fastest processes ever recorded with an 800 nm Ti:sapphire laser. It also appears that this method can be pushed to significantly shorter response times down to about one half of the laser duration itself, i.e., to the single-optical cycle regime. One possible scenario for doing so is the slow detuning of the resonance condition, e.g., by gradually changing the composition of TiO_2 layers into $Ti_{1-x}Si_xO_2$ by the method of rugate layers. Another one would certainly be a variation of the center wavelength of the few-cycle excitation pulse. In either way, it appears highly interesting to explore the dynamics in the transition from non-resonant to resonant excitation.

The developed software will hopefully prove a valuable tool to further investigate this scenario. Further investigations may shed new light on the borders of our understanding of nonlinear optics. Finally, implications on the lifetime of the Kerr effect lie at hand, promising an understanding on how fast the Kerr lensing mechanism in Ti:sapphire lasers really is.

REFERENCES

[1] W. E. Lamb Jr., "Theory of an optical maser," *Phys. Rev.*, vol. 134, pp. A1429–A1450, 1964.

[2] L. E. Hargrove, R. L. Fork, and M. A. Pollack, "Locking of He-Ne laser modes induced by synchronous intracavity modulation," *Appl. Phys. Lett.*, vol. 5, 1964.

[3] F. McClung and R. Hellwarth, "Giant optical pulsations from ruby," *J. Appl. Phys.*, vol. 33, pp. 828–829, 1962.

[4] H. A. Haus, "Mode-locking of lasers," *IEEE J. Sel. Topics Quantum Electron.*, vol. 6, pp. 1173–1185, 2000.

[5] J. Degnan, "Optimization of passively Q-switched lasers," *IEEE J. Quantum Electron.*, vol. 31, pp. 1890–1901, 1995.

[6] J. A. Armstrong, "Measurement of picosecond laser pulse widths," *Appl. Phys. Lett.*, vol. 10, 1967.

[7] D. E. Spence, P. N. Kean, and W. Sibbett, "60-fsec pulse generation from a self-mode-locked Ti:sapphire laser," *Opt. Lett.*, vol. 16, pp. 42–44, 1991.

[8] D. J. Kane and R. Trebino, "Characterization of arbitrary femtosecond pulses using frequency-resolved optical gating," *IEEE J. Quantum Electron.*, vol. 29, pp. 571–579, 1993.

[9] T. Tsang, M. A. Krumbügel, K. W. DeLong, D. N. Fittinghoff, and R. Trebino, "Frequency-resolved optical-gating measurements of ultrashort pulses using surface third-harmonic generation," *Opt. Lett.*, vol. 21, pp. 1381–1383, 1996.

[10] K. W. DeLong, R. Trebino, J. Hunter, and W. E. White, "Frequency-resolved optical gating with the use of second-harmonic generation," *J. Opt. Soc. Am. B*, vol. 11, pp. 2206–2215, 1994.

[11] K. W. DeLong, R. Trebino, and D. J. Kane, "Comparison of ultrashort-pulse frequency-resolved optical-gating traces for three common beam geometries," *J. Opt. Soc. Am. B*, vol. 11, pp. 1595–1608, 1994.

[12] G. Stibenz and G. Steinmeyer, "Interferometric frequency-resolved optical gating," *Opt. Express*, vol. 13, pp. 2617–2626, 2005.

[13] A. Anderson, K. S. Deryckx, X. G. Xu, G. Steinmeyer, and M. B. Raschke, "Few-femtosecond plasmon dephasing of a single metallic nanostructure from optical response function reconstruction by interferometric frequency resolved optical gating," *Nano Lett.*, vol. 10, pp. 2519–2524, 2010.

[14] S. K. Das, C. Schwanke, A. Pfuch, W. Seeber, M. Bock, G. Steinmeyer, T. Elsaesser, and R. Grunwald, "Highly efficient THG in TiO_2 nanolayers for third-order pulse characterization," *Opt. Express*, vol. 19, pp. 16985–16995, 2011.

[15] S. K. Das, M. Bock, R. Grunwald, B. Borchers, J. Hyyti, G. Steinmeyer, D. Ristau, A. Harth, T. Vockerodt, T. Nagy, and U. Morgner, "First measurement of the non-instantaneous response time of a $\chi^{(3)}$ nonlinear optical effect," in *EPJ Web of Conferences*, vol. 41, p. 12005, 2013.

[16] J. Hyyti, L. Orsila, and G. Steinmeyer, "Analysis of non-instantaneous lifetime of Kerr-effect in interferometric FROG measurements," in *Physics Days 2012 Proceedings*, p. P1.36, 2012.

[17] J. C. Maxwell, "On physical lines of force," *Philosophical Magazine*, 1861.

[18] G. P. Agrawal, *Nonlinear Fiber Optics*. Academic Press, third ed., 2001.

[19] R. W. Boyd, *Nonlinear Optics*. Academic Press, second ed., 2003.

[20] E. Hecht, *Optics*. Addison Wesley, fourth ed., 2002.

[21] H. Fizeau, "Sur les hypothèses relatives à l'éther lumineux," *Comptes Rendus*, vol. 33, pp. 349–355, 1851.

[22] J. C. Maxwell, "A dynamical theory of the electromagnetic field," *Philosophical Transactions of the Royal Society of London*, vol. 155, pp. 459–512, 1865.

[23] Conférence G'en'erale des Poids et Mesures (CGPM), "Resolution 1 of the 17^{th} meeting of the CGPM," 1983. available at http://www.bipm.org/en/CGPM/db/17/1/.

[24] P. Franken, A. Hill, C. Peters, and G. Weinreich, "Generation of optical harmonics," *Phys. Rev. Lett.*, vol. 7, pp. 118–119, 1961.

[25] T. H. Maiman, "Stimulated optical radiation in ruby," *Nature*, vol. 187, pp. 493–494, 1960.

[26] Federal Communications Commission, "Cable landing license," 1999. Available at http://transition.fcc.gov/Bureaus/International/Orders/1999/da992042.txt.

[27] M. Bass, P. A. Franken, A. E. Hill, C. W. Peters, and G. Weinreich, "Optical mixing," *Phys. Rev. Lett.*, vol. 8, p. 18, 1962.

[28] R. L. Carman, R. Y. Chiao, and P. L. Kelley, "Observation of degenerate stimulated four-photon interaction and four-wave parametric amplification," *Phys. Rev. Lett. 17*, vol. 17, pp. 1281–1283, 1966.

[29] R. H. Stolen and A. Ashkin, "Optical Kerr effect in glass waveguide," *Appl. Phys. Lett.*, vol. 22, pp. 294–296, 1973.

[30] F. Shimizu, "Frequency broadening in liquids by a short light pulse," *Phys. Rev. Lett.*, vol. 19, pp. 1097–1100, 1967.

[31] R. Y. Chiao, E. Garmire, and C. H. Townes, "Self-trapping of optical beams," *Phys. Rev. Lett.*, vol. 13, pp. 479–482, 1964.

[32] T. Brabec, C. Spielmann, P. F. Curley, and F. Krausz, "Kerr lens mode locking," *Opt. Lett.*, vol. 17, pp. 1292–1294, 1992.

[33] C. Grebing, *Neuartige Konzepte zur Detektion und Kontrolle der Carrier-Envelope Phasendrift ultrakurzer Laserimpulse.* PhD thesis, Humboldt-Universität zu Berlin, 2010.

[34] R. Trebino, *Frequency Resolved Optical Gating: The Measurement of Ultrashort Laser Pulses.* Kluwer Academic Publishers, 2000.

[35] P. Lazaridis, G. Debarge, and P. Gallion, "Time-bandwidth product of chirped sech^2 pulses: application to phase–amplitude-coupling factor measurement," *Opt. Lett.*, vol. 20, pp. 1160–1162, 1995.

[36] R. Paschotta, "Encyclopedia of laser physics and technology." Retrieved in 2013.

[37] S. Tsuda, W. Knox, S. Cundiff, W. Jan, and J. Cunningham, "Mode-locking ultrafast solid-state lasers with saturable Bragg reflectors," *IEEE J. Sel. Topics Quantum Electron.*, vol. 2, pp. 454–464, 1996.

[38] J. D. Kafka, D. W. Hall, and T. Baer, "Mode-locked erbium-doped fiber laser with soliton pulse shaping," *Opt. Lett.*, vol. 14, pp. 1269–1271, 1989.

[39] M. Guina, N. Xiang, and O. Okhotnikov, "Stretched-pulse fiber lasers based on semiconductor saturable absorbers," *Appl. Phys. B*, vol. 74, pp. 193–200, 2002.

[40] U. Morgner, F. X. Kärtner, S. H. Cho, Y. Chen, H. A. Haus, J. G. Fujimoto, E. P. Ippen, V. Scheuer, G. Angelow, and T. Tschudi, "Sub-two-cycle pulses from a Kerr-lens mode-locked Ti:sapphire laser," *Opt. Lett.*, vol. 24, pp. 411–413, 1999.

[41] A. Baltuška, Z. Wei, M. S. Pshenichnikov, D. A. Wiersma, and R. Szipőcs, "All-solid-state cavity-dumped sub-5-fs laser," *Applied Physics B: Lasers and Optics*, vol. 65, pp. 175–188, 1997.

[42] P. M. Paul, E. S. Toma, P. Breger, G. Mullot, F. Auge, P. Balcou, H. G. Muller, and P. Agostini, "Observation of a train of attosecond pulses from high harmonic generation," *Science*, vol. 292, pp. 1689–1692, 2001.

[43] K. Zhao, Q. Zhang, M. Chini, Y. Wu, X. Wang, and Z. Chang, "Tailoring a 67 attosecond pulse through advantageous phase-mismatch," *Opt. Lett.*, vol. 37, pp. 3891–3893, 2012.

[44] C. Hernández-García, J. A. Pérez-Hernández, T. Popmintchev, M. M. Murnane, H. C. Kapteyn, A. Jaron-Becker, A. Becker, and L. Plaja1, "Zeptosecond high harmonic keV X-ray waveforms driven by midinfrared laser pulses," *Phys. Rev. Lett.*, vol. 111, p. 033002, 2013.

[45] R. Trebino, K. W. DeLong, D. N. Fittinghoff, J. N. Sweetser, M. A. Krumbügel, and B. A. Richman, "Measuring ultrashort laser pulses in the time-frequency domain using frequency-resolved optical gating," *Rev. Sci. Instrum.*, vol. 68, pp. 3277–3295, 1997.

[46] B. Kohler, V. V. Yakovlev, J. Che, J. L. Krause, M. Messina, K. R. Wilson, N. Schwentner, R. M. Whitnell, and Y. Yan, "Quantum control of wave packet evolution with tailored femtosecond pulses," *Phys. Rev. Lett.*, vol. 74, pp. 3360–3363, 1995.

[47] P. Zhou, H. Schulz, and P. Kohns, "Atomic spectroscopy with ultrashort laser pulses using frequency-resolved optical gating," *Opt. Commun.*, vol. 123, pp. 501–504, 1996.

[48] T. S. Clement, G. Rodriguez, W. M. Wood, and A. J. Taylor, "Characterization of ultrafast interactions with materials through direct measurement of the optical phase," in *Proc. SPIE 2701, Generation, Amplification, and Measurement of Ultrashort Laser Pulses III*, pp. 229–234, 1996.

[49] T. Olivier, F. Billard, and H. Akhouayri, "Nanosecond Z-scan measurements of the nonlinear refractive index of fused silica," *Opt. Express*, vol. 12, pp. 1377–1382, 2004.

[50] D. J. Maas, B. Rudin, A.-R. Bellancourt, D. Iwaniuk, S. V. Marchese, T. Südmeyer, and U. Keller, "High precision optical characterization of semiconductor saturable absorber mirrors," *Opt. Express*, vol. 16, pp. 7571–7579, 2008.

[51] G. Steinmeyer, "A review of ultrafast optics and optoelectronics," *J. Opt. A: Pure Appl. Opt.*, vol. 5, pp. R1–R15, 2003.

[52] G. J. Tearney, B. E. Bouma, and B. E. Fujimoto, "High-speed phase- and group-delay scanning with a grating-based phase control delay line," *Opt. Lett.*, vol. 22, pp. 1811–1813, 1997.

[53] D. Umstadter, E. Esarey, and J. Kim, "Nonlinear plasma waves resonantly driven by optimized laser pulse trains," *Phys. Rev. Lett.*, vol. 72, pp. 1224–1227, 1994.

[54] T. Udem, R. Holzwarth, and T. W. Hänsch, "Optical frequency metrology," *Nature*, vol. 416, pp. 233–237, 2002.

[55] C. L. Thomsen, D. Madsen, S. R. Keiding, J. Thøgersen, and O. Christiansen, "Two-photon dissociation and ionization of liquid water studied by femtosecond transient absorption spectroscopy," *J. Chem. Phys.*, vol. 110, pp. 3453–3462, 1999.

[56] A. Baltuska, M. S. Pshenichnikov, and D. A. Wiersma, "Second-harmonic generation frequency-resolved optical gating in the single-cycle regime," *IEEE J. Quantum Electron.*, vol. 35, pp. 459–478, 1999.

[57] A.-C. Tien, S. Kane, J. Squier, B. Kohler, and K. Wilson, "Geometrical distortions and correction algorithm in single-shot pulse measurements: application to frequency-resolved optical gating," *J. Opt. Soc. Am. B*, vol. 13, pp. 1160–1165, 1996.

[58] G. Stibenz and G. Steinmeyer, "Structures of interferometric frequency-resolved optical gating," *IEEE. J. Sel. Top. Quantum Electron.*, vol. 12, pp. 286–296, 2006.

[59] I. Amat-Roldán, I. Cormack, P. Loza-Alvarez, E. Gualda, and D. Artigas, "Ultrashort pulse characterisation with SHG collinear-FROG," *Opt. Express*, vol. 12, pp. 1169–1178, 2004.

[60] J.-H. Chung and A. M. Weiner, "Ambiguity of ultrashort pulse shapes retrieved from the intensity autocorrelation and the power spectrum," *IEEE J. Quantum Electron.*, vol. 7, pp. 656–666, 2001.

[61] R. W. Gerchberg and W. O. Saxton, "A practical algorithm for the determination of phase from image and diffraction plane pictures," *Optik*, vol. 35, pp. 237–246, 1972.

[62] J. Peatross and A. Rundquist, "Temporal decorrelation of short laser pulses," *J. Opt. Soc. Am. B*, vol. 15, pp. 216–222, 1998.

[63] J. W. Nicholson and W. Rudolph, "Noise sensitivity and accuracy of femtosecond pulse retrieval by phase and intensity from correlation and spectrum only (PICASO)," *J. Opt. Soc. Am. B*, vol. 19, pp. 330–339, 2002.

[64] C. Iaconis and I. A. Walmsley, "Spectral phase interferometry for direct electric-field reconstruction of ultrashort optical pulses," *Opt. Lett.*, vol. 23, pp. 792–794, 1998.

[65] R. Trebino and D. J. Kane, "Using phase retrieval to measure the intensity and phase of ultrashort pulses: frequency-resolved optical gating," *J. Opt. Soc. Am. A*, vol. 10, pp. 1101–1111, 1993.

[66] S. H. Nawab, T. F. Quatieri, and J. S. Lim, "Signal reconstruction from short-time Fourier transform magnitude," *IEEE Trans. Acoust. Speech Signal Process.*, vol. 31, pp. 986–998, 1983.

[67] D. T. Reid, "Algorithm for complete and rapid retrieval of ultrashort pulse amplitude and phase from a sonogram," *IEEE J. Quantum Electron.*, vol. 35, pp. 1584–1589, 1999.

[68] K. W. DeLong, D. N. Fittinghoff, and R. Trebino, "Pulse retrieval in frequency-resolved optical gating based on the method of generalized projections," *Opt. Lett.*, vol. 19, pp. 2152–2154, 1994.

[69] H. S. Black, *Modulation Theory.* Van Nostrad, 1953.

[70] G. Ramos-Ortiz, M. Cha, S. Thayumanavan, J. Mendez, S. R. Marder, and B. Kippelen, "Ultrafast-pulse diagnostic using third-order frequency-resolved optical gating in organic films," *Appl. Phys. Lett.*, vol. 85, pp. 3348–3350, 2004.

[71] P. Langlois and E. P. Ippen, "Measurement of pulse asymmetry by three-photon-absorption in a GaAsP photodiode," *Opt. Lett.*, vol. 24, pp. 1868–1870, 1999.

[72] T. Y. F. Tsang, "Optical third-harmonic generation at interfaces," *Phys.Rev. A*, vol. 52, pp. 4116–4125, 1995.

[73] T. Tsang, "Third- and fifth-harmonic generation at the interfaces of glass and liquids," *Phys. Rev. A*, vol. 54, pp. 5454–5457, 1996.

[74] V. N. Ginzburg, N. V. Didenko, A. V. Konyashchenko, V. V. Lozhkarev, G. A. Luchinin, A. P. Lutsenko, S. Y. Mironov, E. A. Khazanov, and I. V. Yakovlev, "Third-order correlator for measuring the time profile of petawatt laser pulses," *Quantum Electron.*, vol. 38, pp. 1027–1032, 2008.

[75] E. Palik, *Handbook of Optical Constants of Solids II*. Academic Press, 1991.

[76] S. Y. Kim, "Simultaneous determination of refractive index, extinction coefficient, and void distribution of titanium dioxide thin film by optical methods," *Appl. Opt.*, vol. 35, pp. 6703–6707, 1996.

[77] R. Barille, L. Canioni, L. Sarger, and G. Rivoire, "Nonlinearity measurements of thin films by third-harmonic-generation microscopy," *Phys. Rev. E Stat. Nonlin. Soft Matter Phys.*, vol. 66, p. 067602, 2002.

[78] B. Kulyk, Z. Essaidi, V. Kapustianyk, V. Turko, V. Rudyk, M. Partyka, M. Addou, and B. Sahraoui, "Second and third order nonlinear optical properties of nanostructured ZnO thin films deposited on α-BBO and $LiNbO_3$," *Opt. Commun.*, vol. 281, pp. 6107–6111, 2008.

[79] G. I. Petrov, V. Shcheslavskiy, V. V. Yakovlev, I. Ozerov, E. Chelnokov, and W. Marine, "Efficient third-harmonic generation in a thin nanocrystalline film of ZnO," *Appl. Phys. Lett.*, vol. 83, pp. 3993–3995, 2003.

[80] VENTEON Laser Technologies GmbH, Germany, "VENTEON | PULSE : ONE." PDF Brochure, available at venteon.com.

[81] M. Lappschies, B. Görtz, and D. Ristau, "Application of optical broadband monitoring to quasi-rugate filters by ion-beam sputtering," *Appl. Opt.*, vol. 45, pp. 1502–1506, 2006.

[82] L. Kavan, M. Grätzel, S. E. Gilbert, C. Klemenz, and H. J. Scheel, "Electrochemical and photoelectrochemical investigation of single-crystal anatase," *J. Am. Chem. Soc.*, vol. 118, pp. 6716–6723, 1996.

[83] J. M. Bennett, E. Pelletier, G. Albrand, J. P. Borgogno, B. Lazarides, C. K. Carniglia, R. A. Schmell, T. H. Allen, T. Tuttle-Hart, K. H. Guenther, and A. Saxer, "Comparison of the properties of titanium dioxide films prepared by various techniques," *Appl. Opt.*, vol. 28, pp. 3303–3317, 1989.

[84] H. Long, A. Chen, G. Yang, Y. Li, and P. Lu, "Third-order optical nonlinearities in anatase and rutile TiO_2 thin films," *Thin Solid Films*, vol. 517, pp. 5601–5604, 2009.

[85] S. Valencia, J. M. Marín, and G. Restrepo, "Study of the bandgap of synthesized titanium dioxide nanoparticules using the sol-gel method and a hydrothermal treatment," *Open Mater. Sci. J*, vol. 4, pp. 9–14, 2010.

[86] S. Miyazaki, H. Nishimura, M. Fukuda, L. Ley, and J. Ristein, "Structure and electronic states of ultrathin SiO_2 thermally grown on Si(100) and Si(111) surfaces," *Appl. Surf. Sci.*, vol. 113-114, pp. 585–589, 1997.

[87] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*. Prentice Hall, second ed., 1999.

[88] S. G. Hoggar, *Mathematics of Digital Images*. Academic Press, 2006.

[89] D. Mandic, M. Golz, A. Kuh, D. Obradovic, and T. Tanaka, eds., *Signal Processing Techniques for Knowledge Extraction and Information Fusion*. Springer, 2008.

[90] J. W. Cooley and J. W. Tukey, "An algorithm for the machine computation of the complex fourier series," *Math. Comp.*, vol. 19, pp. 297–301, 1965.

[91] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, second ed., 1992.

[92] M. J. D. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *Comput. J.*, vol. 7, pp. 155–162, 1964.

[93] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence properties of the Nelder–Mead simplex method in low dimensions," *SIAM J. Optim.*, vol. 9, pp. 112–147, 1998.

[94] J. A. Nelder and R. Mead, "A simplex method for function minimization," *Comput. J.*, vol. 7, pp. 308–313, 1965.

[95] O. Svelto, *Principles of Lasers*. Springer, fifth ed., 2010.

[96] R. Szipöcs, C. Spielmann, F. Krausz, and K. Ferencz, "Chirped multilayer coatings for broadband dispersion control in femtosecond lasers," *Opt. Lett.*, vol. 19, pp. 201–203, 1994.

[97] I. H. Malitson, "Interspecimen comparison of the refractive index of fused silica," *J. Opt. Soc. Am.*, vol. 55, pp. 1205–1209, 1965.

[98] C. C. Evans, J. D. B. Bradley, E. A. Martí-Panameño, and E. Mazur, "Mixed two- and three-photon absorption in bulk rutile (TiO_2) around 800 nm," *Opt. Express*, vol. 20, pp. 3118–3128, 2012.

[99] C. Brée, *Nonlinear Optics in the Filamentation Regime*. Springer, 2012.

[100] D. J. Cook and R. M. Hochstrasser, "Intense terahertz pulses by four-wave rectification in air," *Opt. Lett.*, vol. 25, pp. 1210–1212, 2000.

[101] T. Bartel, P. Gaal, K. Reimann, M. Woerner, and T. Elsaesser, "Generation of single-cycle THz transients with high electric-feld amplitudes," *Opt. Lett.*, vol. 30, pp. 2805–2807, 2005.

[102] X. Xie, J. Dai, and X.-C. Zhang, "Coherent control of thz wave generation in ambient air," *Phys. Rev. Lett.*, vol. 96, p. 075005, 2006.

A. SOURCE CODE FOR PULSE RETRIEVAL SOFTWARE

The MATLAB source code for the pulse retrieval software is presented in its entirety here. The software comprises 21 files, organized in sections in alphabetical order. In addition, a sample script for running the program is presented before the functions.

A.1 Sample script: DataGenerator.m

```
1  % ------------------------------------------------------------------------
2  % DataGenerator.m
3  %
4  % User operated script for the Reconstruction program, which reconstructs a
5  % pulse from THIFROG trace.
6  %
7  % Date: 2013-10-08
8  % Created by: Janne Hyyti
9  % ------------------------------------------------------------------------
10 %% Get the path for current directory.
11 % MAKE SURE YOUR CURRENT FOLDER IS THE LOCATION OF THE PROGRAM!!!
12 programPath = pwd;
13 programPath = [programPath '\'];
14
15 %% PREPARE DATA
16 % Run this for the data to prepare it for the main program. Only necessary
17 % to do once for each trace. After that you can use the files created for
18 % the actual reconstruction.
19 % run([programPath 'PrepareData.m'])
20
21 %% RECONSTRUCTION
22 % ------------------------------------------------------------------------
23 % Settings for cropping the trace to a suitable size.
24 % ------------------------------------------------------------------------
25 frequencyResolution = 32;
26 delayRange = 70e-15;
27
28 % ------------------------------------------------------------------------
29 % Settings for optimization.
30 % ------------------------------------------------------------------------
31 optimize = 1; % set to 0 to just print figures and no optimization.
32 bands = [1 0 0]; % [baseband, fundamental, second harmonic]
33 weights = [1 0 0]; % [baseband, fundamental, second harmonic]
34 optimizationPoints = 51;
35 iterationsPerCycle = 30;
36
37 % ------------------------------------------------------------------------
38 % Parameters for automated handling...
39 % ------------------------------------------------------------------------
40
```

```
% Enter the filenames of the prepared data files without the '_D' or '_P'
% suffixes.
filenames = {'FROG_traceSTHGback_1000ms0gain', ...
    'FROG_traceTiO2THGback_500ms0gainLong'};

% Enter a name for the scenarios, as many as there are filenames. These are
% included in the filenames of the computed and saved data.
scenario = {'SiO_2', 'TiO_2'};
% scenario = filenames;

% Location of the source data, prepared by the PrepareData function.
pathData = [programPath 'SourceData\'];
pathD = strcat(pathData, filenames, '_D.mat');
pathP = strcat(pathData, filenames, '_P.mat');

% Location for the saved output files.
saveFolderPath = [programPath 'SaveFolder\'];
savePath = strcat(saveFolderPath, filenames, '\');

% Loop to do reconstruction for each of the files.
for ii = 1:numel(scenario)
    % Tag to be included in every file produced by this run.
    tag = 'run1';

    % Create the save folder.
    mkdir(saveFolderPath, filenames{ii})

    % Reconstruct the pulse.
    [~, ...
        ~, ~, ~, ...
        ~, ~] = ...
        ReconstructionV13(optimize, ...
        frequencyResolution, delayRange, ...
        bands, weights, optimizationPoints, ...
        scenario{ii}, ...
        pathD{ii}, pathP{ii}, ...
        savePath{ii}, ...
        tag, iterationsPerCycle);

    close all
end
```

A.2 ComplexPulse.m

```
% Gives the complex form pulse from amplitude and phase information.
function pulse = ComplexPulse(t, omega0, amplitude,phi)
pulse = amplitude .* exp(1i* (omega0*t + phi) );
end
```

A.3 ExtractBandsFast.m

```
% Extract the desired bands.
function [baseband, harm1, harm2] = ExtractBandsFast ...
    (trace, ...
    frequency_indices, ...
    sineData2, cosineData2, ...
    sineData3, cosineData3, ...
    bands)

% Format output variables.
baseband = 0;
```

```
harm1 = 0;
harm2 = 0;

% Extract selected bands.
if bands(1)
    baseband = ExtractBasebandFast(trace, ...
    frequency_indices, ...
    sineData2{1}, cosineData2{1}, ...
    sineData3{1}, cosineData3{1});
end
if bands(2)
    harm1 = ExtractHarmonicFast(trace, ...
    frequency_indices, ...
    sineData2{2}, cosineData2{2}, ...
    sineData3{2}, cosineData3{2});
end
if bands(3)
    harm2 = ExtractHarmonicFast(trace, ...
    frequency_indices, ...
    sineData2{3}, cosineData2{3}, ...
    sineData3{3}, cosineData3{3});
end
end
```

A.4 ExtractBasebandFast.m

```
% Fast extraction of baseband from a trace.
function output = ExtractBasebandFast(trace1, ...
    frequencyIndices, ...
    sineData2, cosineData2, ...
    sineData3, cosineData3)
% Fourier transform to delay–frequency space. Only the frequencies
% corresponding to the baseband are included ---> no other filtering
% required!!!

% Calculate only positive frequencies (and zero freq). This is defined by
% the 'n' vector in cosineData creation; n = 0:n_max. We're transforming
% real valued data, so the transform should be symmetric in the Fourier
% domain, i.e. negative frequencies are equal to conjugate positive
% frequencies.
F1 = arrayfun(@(frequencyIndex) ...
    FourierDelayAxis(trace1(frequencyIndex,:), ...
    sineData2, cosineData2), ...
    frequencyIndices.', 'UniformOutput', false);
F1 = cell2mat(F1);

% Use the positive frequencies to append the transform with the negative
% frequencies.
F1 = [conj(fliplr(F1(:,2:end))) F1];

% Transform back.
IF1 = arrayfun(@(frequencyIndex) ...
    FourierDelayAxis(F1(frequencyIndex,:), ...
    sineData3, cosineData3), ...
    frequencyIndices.', 'UniformOutput', false);
IF1 = abs(cell2mat(IF1));
output = IF1/max(max(IF1));
end
```

A.5 ExtractHarmonicFast.m

```
% Fast extraction of first harmonic band from a trace.
function output = ExtractHarmonicFast(trace1, ...
    frequencyIndices, ...
    sineData2, cosineData2, ...
    sineData3, cosineData3)
% Fourier transform to delay–frequency space. Only the frequencies
% corresponding to the harmonic band are included ---> no other filtering
% required!!!

% Calculate only positive frequencies. This is defined by the 'n' vector in
% cosineData creation; n = n_min:n_max.
F1 = arrayfun(@(frequencyIndex) ...
    FourierDelayAxis(trace1(frequencyIndex,:), ...
    sineData2, cosineData2), ...
    frequencyIndices.', 'UniformOutput', false);
F1 = cell2mat(F1);

% Append the transform with the negative frequencies (only zeros).
F1 = [zeros(size(F1)) F1];

% Transform back.
IF1 = arrayfun(@(frequencyIndex) ...
    FourierDelayAxis(F1(frequencyIndex,:), ...
    sineData3, cosineData3), ...
    frequencyIndices.', 'UniformOutput', false);
IF1 = abs(cell2mat(IF1));
output = IF1/max(max(IF1));
end
```

A.6 ExtractPhase.m

```
% Get the phase of a pulse.
function phi = ExtractPhase(t, omega0, pulse)
phi = phase(pulse.*exp(-1i*omega0*t));
end
```

A.7 FourierDelayAxis.m

```
% Custom Fourier transform along the delay axis.
function output = FourierDelayAxis(data,sineData, cosineData)
output = dot(repmat(data.',1,size(sineData,2)),cosineData,1) ...
    + 1i*dot(repmat(data.',1,size(sineData,2)),sineData,1);
```

A.8 NormFunction.m

```
% Norm function for optimization.
function error = NormFunction(pulseAmpPhi, ...
    delays, pulseDelays, sparseDelays, ...
    baseOriginal, harm1Original, harm2Original, ...
    division, frequencyIndicesD, ...
    sineData2, cosineData2, ...
    sineData3, cosineData3, ...
    integerDelays, omega0, ...
    NF, frequencyIndicesF, ...
    bands, weights)
% ------------------------------------------------------------------------
```

```
% Compute the trace resulting from the pulse given as a parameter.
% ------------------------------------------------------------------------

% Interpolate the XX optimized values to match the time spacing of the
% pulse.
amplitude = interp1(sparseDelays, pulseAmpPhi(1,:), pulseDelays);
phi = interp1(sparseDelays, pulseAmpPhi(2,:), pulseDelays);

% Alternatively , use cubic spline interpolation.
% amplitude = interp1(sparseDelays, pulseAmpPhi(1,:), pulseDelays, 'spline');
% phi = interp1(sparseDelays, pulseAmpPhi(2,:), pulseDelays, 'spline');

% Convert the pulse into complex form.
pulse = ComplexPulse(pulseDelays, omega0, amplitude, phi);

% Calculate the normalized trace.
traceFit = TraceFun(pulse, delays, division, ...
    integerDelays, ...
    NF, frequencyIndicesF);

% Calculate error from different bands.
error = 0;
if bands(1)
    baseFit = ExtractBasebandFast(traceFit, ...
    frequencyIndicesD, ...
    sineData2{1}, cosineData2{1}, ...
    sineData3{1}, cosineData3{1});
    error = error + ...
        weights(1) * TraceErrorFunction(baseOriginal, baseFit);
end
if bands(2)
    harm1Fit = ExtractHarmonicFast(traceFit, ...
        frequencyIndicesD, ...
        sineData2{2}, cosineData2{2}, ...
        sineData3{2}, cosineData3{2});
    error = error + ...
        weights(2) * TraceErrorFunction(harm1Original, harm1Fit);
end
if bands(3)
    harm2Fit = ExtractHarmonicFast(traceFit, ...
        frequencyIndicesD, ...
        sineData2{3}, cosineData2{3}, ...
        sineData3{3}, cosineData3{3});
    error = error + ...
        weights(3) * TraceErrorFunction(harm2Original, harm2Fit);
end
end
```

A.9 ObtainParameters.m

```
% Prompts the user to provide the necessary parameters for the measured
% trace. Saves the parameters to a "P-file".
function saveName = ObtainParameters(fileNameData,pathNameData,pathNameSave)

c = 299792458; % Speed of light.

wavelengthMin = inputdlg('Minimum wavelength of measurement data in nm');
wavelengthMax = inputdlg('Maximum wavelength of measurement data in nm');
wavelengthMin = str2double(wavelengthMin);
wavelengthMax = str2double(wavelengthMax);

delayMin = inputdlg('Minimum time delay of measurement data in fs');
delayMax = inputdlg('Maximum time delay of measurement data in fs');
delayMin = str2double(delayMin);
```

```
delayMax = str2double(delayMax);

division = inputdlg('Power division of beam splitter','',1,{'1'});
division = str2double(division);

data = load([pathNameData fileNameData]);
data = rot90(data);

NoOfWavelengthPoints = size(data,1);
NoOfDelayPoints = size(data,2);
delays = linspace(delayMin,delayMax,NoOfDelayPoints);
wavelengths = linspace(wavelengthMin,wavelengthMax, ...
    NoOfWavelengthPoints);

h1 = figure;
s = get(0, 'ScreenSize');
set(h1,'Position', [0 0 s(3) s(4)])
contourf(delays,wavelengths,data,50,'linestyle','none');
xlabel('Delay [fs]')
ylabel('Wavelength [nm]')
title('Your data')
rise = inputdlg('How many cycles roughly is the rise time?', ...
    '',1,{'3'});
fall = inputdlg('How many cycles roughly is the fall time?', ...
    '',1,{'3'});

hDlg = helpdlg('Select a point for the LOWEST wavelength with meaningful data');
waitfor(hDlg);
[~,w1] = ginput(1);
hDlg = helpdlg('Select a point for the HIGHEST wavelength with meaningful data');
waitfor(hDlg);
[~,w2] = ginput(1);
fMin = floor(c/(w2*1e-9));
fMax = ceil(c/(w1*1e-9));


hDlg = helpdlg(['Choose a point at one crest, then another ' ...
    'one on the FIFTH crest from the first one TO THE RIGHT.'] ...
    ,'Define the duration of one cycle');
waitfor(hDlg);

hold on
[d1,w1] = ginput(1);
plot(d1,w1,'gx','markersize',40)
[d2,~] = ginput(1);
tCycle = (d2*1e-15 - d1*1e-15)/5;
close(h1);
clear d1 w1 d2 h1

rise = str2double(rise) * tCycle;
fall = str2double(fall) * tCycle;

deltaDelay = inputdlg(['Delay step in fs to be used.' ...
    'Should be around 1/10 of cycle.'],'',1,{'0.2'});
deltaDelay = str2double(deltaDelay)*1e-15;
deltaPulse = deltaDelay / 2;

saveName = [pathNameSave, '\', fileNameData(1:end-4), '_P.mat'];
save(saveName,'wavelengthMin', 'wavelengthMax', ...
    'delayMin', 'delayMax', ...
    'division', 'tCycle', 'rise', 'fall', ...
    'fMin', 'fMax', 'deltaDelay','deltaPulse')
```

A.10 PeakFunction.m

```
% Find the first local maximum position of a vector. Used in
% ExtractBaseband function.
function peakPosition = PeakFunction(index,F, bound)
[~, peakPosition] = ...
    findpeaks(abs(F(index, bound:end)), 'MINPEAKDISTANCE', ...
    15, 'NPEAKS', 1);
if isempty(peakPosition)
    peakPosition = 0;
end
peakPosition = peakPosition + bound - 1;
end
```

A.11 PrepareData.m

```
% Prepares measurement data to be used with the Reconstruction function for
% pulse retrieval. Once this function has finished, "D" and "P" files for
% all the datasets prepared here are saved on HDD. Calls the
% ObtainParameters, RemoveArtefacts and ProcessDataInitial functions.
function PrepareData

ii = 1;
more = 1;
while more
    [filenameData{ii},pathData{ii}] = uigetfile('.txt','Select a text file containing the data');
    ii = ii + 1;
    selection = questdlg('Do you want to select another dataset to prepare?', ...
        '', ...
        'Yes', 'No', 'No');
    switch selection
        case 'No'
            more = 0;
    end
end

numberOfFiles = size(filenameData,2);

% Select a folder to save edited data and setting files into
pathSave = uigetdir('','Select a folder to save edited data and setting files into.');

previous = '';

for ii = 1:numberOfFiles

    % Obtain parameters for the current dataset
    loopOK = 0;
    while ~loopOK
        selection = MFquestdlg([0.5 0.3], ['Parameters for the data in ' filenameData{ii}], ...
            '', ...
            'Enter new','Use existing file','Use previous' ,'Use previous');

        switch selection
            case 'Enter new'
                saveName = ObtainParameters(filenameData{ii},pathData{ii},pathSave);
                previous = saveName;
                load(previous);
                loopOK = 1;
            case 'Use existing file'
                [filenameSettings,pathSettings, ~] = ...
                    uigetfile('.mat','Select settings file');
                load([pathSettings, filenameSettings])
                saveName = [pathSave, '\', filenameData{ii}(1:end-4), '_P.mat'];
```

```
                save(saveName,'wavelengthMin', 'wavelengthMax', ...
                    'delayMin', 'delayMax', ...
                    'division', 'tCycle', 'rise', 'fall', ...
                    'fMin', 'fMax', 'deltaDelay','deltaPulse')
                previous = saveName;
                loopOK = 1;
            case 'Use previous'
                if ~strcmp(previous,'')
                    saveName = [pathSave, '\', filenameData{ii}(1:end-4), '_P.mat'];
                    save(saveName,'wavelengthMin', 'wavelengthMax', ...
                        'delayMin', 'delayMax', ...
                        'division', 'tCycle', 'rise', 'fall', ...
                        'fMin', 'fMax', 'deltaDelay','deltaPulse')
                    loopOK = 1;
                else
                    errordlg('There were no previous settings to be used!','Error')
                end
        end
    end

    filename = [pathData{ii} '\' filenameData{ii}];

    % Remove artefacts from the data
    traceIn = RemoveArtefacts({filename});

    [traceAfterInitialProcessing, centerFrequency, centerDelay] = ...
        ProcessDataInitial(traceIn, ...
        wavelengthMin, wavelengthMax, ...
        delayMin, delayMax);

    saveName = [saveName(1:end-5) 'D.mat'];
    save(saveName,'traceAfterInitialProcessing', 'centerFrequency', 'centerDelay')

end


end
```

A.12 ProcessData.m

```
% Processes the PREPARED measurement data for the simulation. Called by the
% Reconstruction function.
function [traceOut, ...
    delaysCentered, frequenciesExtended, ...
    NF, frequencyIndicesF] = ...
    ProcessData(traceAfterInitialProcessing, ...
    wavelengthMin, wavelengthMax, ...
    delayMin, delayMax, ...
    frequencyResolution, ...
    delayRange, ...
    fMin, fMax, ...
    deltaPulse, deltaDelay, ...
    centerDelay)

% Assume that the trace is already on correct orientation.

noOfWavelengthPoints = size(traceAfterInitialProcessing,1);
noOfDelayPoints = size(traceAfterInitialProcessing,2);

% Vectors for delays and wavelengths.
delays = linspace(delayMin,delayMax,noOfDelayPoints);
wavelengths = linspace(wavelengthMax,wavelengthMin, ...
    noOfWavelengthPoints);

```

```
% Change from fs to s.
delays = delays * 1e-15;

%-------------------------------------------------------------------------
% Convert wavelengths to frequencies.
%-------------------------------------------------------------------------
% Convert the wavelength axis to frequencies. The scale is not correct at
% this point yet, i.e., the frequency axis is not linear.
c = 299792458; % Speed of light.
frequencies = c./(wavelengths*1e-9);

%-------------------------------------------------------------------------
% Define new axes.
%-------------------------------------------------------------------------

% FREQUENCY AXIS

fRange = fMax - fMin;
xiF = fRange / frequencyResolution; % frequency step

% N in the DFT formula.
% deltaPulse = delays(2) - delays(1);
NF = ceil(1/(deltaPulse * xiF));
xiF = 1/(deltaPulse * NF); % update the frequency step

% The indices for wanted frequencies are... (f_n = n*xi)
nMax = floor(fMax / xiF);
nMin = nMax - frequencyResolution + 1;
frequencyIndicesF = nMin:nMax;

% Create axes for new resolution. An equidistant vector is created for the
% frequencies.
frequenciesEquidistant = frequencyIndicesF * xiF;

% DELAY AXIS
noOfPositiveDelays = floor((delayRange / 2) / deltaDelay);
noOfNegativeDelays = noOfPositiveDelays;

delays2 = -noOfNegativeDelays*deltaDelay : deltaDelay : ...
    noOfPositiveDelays*deltaDelay;
delays2 = delays2 + centerDelay;

% Center the delays
delaysCentered = delays2 - centerDelay;

%-------------------------------------------------------------------------
% Fit to new axes
%-------------------------------------------------------------------------
% Cubic spline interpolation to obtain values for the new axes
trace2 = interp2(delays,frequencies.',traceAfterInitialProcessing, ...
    delays2,frequenciesEquidistant.','spline');

% Normalize output.
trace2 = trace2/max(trace2(:));

% Remove very small components.
trace2(trace2 < 1e-5) = 0;

%-------------------------------------------------------------------------
% Extend the frequency range
%-------------------------------------------------------------------------
frequenciesExtention1 = 0 : xiF : frequenciesEquidistant(1);
frequenciesExtention1 = frequenciesExtention1(1:end-1);

steps_to_top = floor((2000e12 - frequenciesEquidistant(end))/ xiF);
frequenciesExtention2 = frequenciesEquidistant(end) + xiF : xiF : ...
```

```
    frequenciesEquidistant(end) + xiF * steps_to_top;

frequenciesExtended = [frequenciesExtention1 ...
    frequenciesEquidistant ...
    frequenciesExtention2];

% Fill the new frequencies with zero data.
traceOut = [zeros(numel(frequenciesExtention1),size(trace2,2)); ...
    trace2;
    zeros(numel(frequenciesExtention2),size(trace2,2));];

% Define new values for FT index n.
nMax = ceil(frequenciesExtended(end) / xiF);
nMin = 0;
frequencyIndicesF = nMin:nMax;

end
```

A.13 ProcessDataInitial.m

```
% Initial processing for measurement data. Called by PrepareData function.
% Removes background, flips the data structure to correct orientation and
% locates zero delay and center frequency.
function [traceOut, centerFrequency, centerDelay] = ...
    ProcessDataInitial(data, ...
    wavelengthMin, wavelengthMax, ...
    delayMin, delayMax)

% Flip to correct orientation...
trace1 = flipud(data);
noOfWavelengthPoints = size(trace1,1);
noOfDelayPoints = size(trace1,2);

% Vectors for delays and wavelengths.
delays = linspace(delayMin,delayMax,noOfDelayPoints);
wavelengths = linspace(wavelengthMax,wavelengthMin, ...
    noOfWavelengthPoints);

% Change from fs to s.
delays = delays * 1e-15;

%-------------------------------------------------------------------------
% Remove background (INITIAL).
%-------------------------------------------------------------------------
% Background level estimate:
backgroundLevel = median(trace1(:));

% Keep a copy of original trace1 for background level adjustment later on.
traceOrig = trace1;

% Substract backround.
trace1 = trace1 - backgroundLevel;

% Negative elements are set to zero.
trace1(trace1 < 0) = 0;

%-------------------------------------------------------------------------
% Convert wavelengths to frequencies.
%-------------------------------------------------------------------------
% Convert the wavelength axis to frequencies. The scale is not correct at
% this point yet, i.e., the frequency axis is not linear.
c = 299792458; % Speed of light.
frequencies = c./(wavelengths*1e-9);

```

```
%-------------------------------------------------------------------------
% Define the center of gravitation.
%-------------------------------------------------------------------------
% The goal here is to find the location of the baseband maximum and use it
% as the center of gravitation.

% FFT to delay-frequencies.
NFFT = size(trace1,2);
F1 = fft(trace1,NFFT,2);
h1 = figure;
s = get(0, 'ScreenSize');
set(h1,'Position', [0 0 s(3) s(4)])
imagesc(abs(F1))
set(gca,'YDir','normal')

hDlg = helpdlg(['Choose a point right from the baseband on the left. ' ...
    'Data right of this point is filtered in order to extract baseband.'] ...
    ,'Filtering for zero delay definition');
waitfor(hDlg);
[x,~] = ginput(1);
x = floor(x);
if(x<2)
    x=2;
end

% Remove all but the baseband.
F1(:,x:end-x+2) = 0;
imagesc(abs(F1))
set(gca,'YDir','normal')
pause(0.5)
close(h1)

% IFFT to obtain the baseband in delay-space.
baseband = abs(ifft(F1,NFFT,2));
h2 = figure(2);
contourf(delays,frequencies,baseband,50,'linestyle','none')
% imagesc(delays,frequencies,baseband)
% set(gca,'YDir','normal')
xlabel('Delays [s]')
ylabel('Frequencies [Hz]')
set(h2,'Position', [0 0 s(3) s(4)])

hold on
hDlg = helpdlg(['Choose a point to be used as the zero delay & center ' ...
    'frequency.'] ...
    ,'Select peak point');
waitfor(hDlg);

[centerDelay,centerFrequency] = ginput(1);

selectionOk = 0;
while ~selectionOk
    hCross = plot(centerDelay, centerFrequency, 'gx', 'Markersize', 40);
    % Dialog.
    selection = questdlg(['Are you satisfied with the selection ' ...
        '(delay = ' num2str(centerDelay*1e15) ' fs, frequency = ' ...
        num2str(centerFrequency*1e-12) ' THz)? ' ...
        'Select No to choose another point'], ...
        '', ...
        'Yes','No','More options...','No');

    switch selection
        case 'Yes'
            selectionOk = 1;
        case 'No'
            set(hCross,'visible','off')
```

```
            [centerDelay,centerFrequency] = ginput(1);
        case 'More options...'
            selectionMore = questdlg(['Are you satisfied with the selection ' ...
                '(delay = ' num2str(centerDelay*1e15) ' fs, frequency = ' ...
                num2str(centerFrequency*1e-12) ' THz)? ' ...
                'Select option'], ...
                '', ...
                'Use centroid','Use peak','Use peak');
            set(hCross,'visible','off')
            switch selectionMore
                case 'Use centroid'
                    xData = sum(baseband);
                    yData = sum(baseband,2).';
                    mass = sum(sum(baseband));

                    xCenter = 1/mass * sum(xData .* (1:numel(xData)));
                    yCenter = 1/mass * sum(yData .* (1:numel(yData)));

                    centerFrequency = interp1(1:numel(frequencies), ...
                        frequencies,yCenter);
                    centerDelay = interp1(1:numel(delays), ...
                        delays,xCenter);
                case 'Use peak'
                    % Locate baseband maximum.
                    [~,index] = max(baseband(:));
                    [freqIndexOfPeak,delayIndexOfPeak]=ind2sub(size(baseband),index);
                    centerFrequency = frequencies(freqIndexOfPeak);
                    centerDelay = delays(delayIndexOfPeak);
            end
    end
end
close(h2)
% ------------------------------------------------------------------------
% Remove background FINAL
% ------------------------------------------------------------------------
BGLevelOK = 0;
BGcnst = 1;
while ~BGLevelOK
    % Remove background.
    trace1 = traceOrig - backgroundLevel * BGcnst;
    trace1(trace1 < 0) = 0;

    % Normalize output.
    trace1 = trace1/max(trace1(:));

    % Remove very small components.
    trace1(trace1 < 1e-5) = 0;

    % Plot and let user check the level
    hBG = figure(6);
    set(hBG,'Position', [0 0 s(3) s(4)])
    imagesc(delays,frequencies,trace1);set(gca,'ydir','normal')
    xlabel('Delays [fs]')
    ylabel('Frequencies [THz]')

    selection = MFquestdlg([0.5 0.3], ['Adjust the background level. ' ...
        'Substract...'], ...
        '', ...
        'More','Less','The level is fine','The level is fine');
    switch selection
        case 'More'
            BGcnst = BGcnst + 0.005;
        case 'Less'
            BGcnst = BGcnst - 0.005;
        case 'The level is fine'
            BGLevelOK = 1;
```

```
177     end
178 end
179 close(hBG)
180 traceOut = trace1;
181 
182 end
```

A.14 PulseShapeFunction.m

```
1  % A function to generate an asymmetric pulse shape.
2  function  [amplitude, phi] = PulseShapeFunction(t, rise, fall)
3  % Hyperbolic secant.
4  amplitude = 2 ./ ( exp(t ./ ( fall) ) +  ...
5      exp(-t ./ ( rise) ) );
6  
7  % Make the amplitude decline to zero at the edges.
8  amplitude = blackman(numel(amplitude)).'.*amplitude;
9  
10 % Flat phase.
11 phi = ones(1,length(t)) * pi;
12 end
```

A.15 ReconstructionV13.m

```
1  % Main function. Reconstructs a pulse from a trace, saves data.
2  function [pulseRecovered, ...
3      pulseDelays, delays, frequencies, ...
4      traceRecovered, traceOriginal] = ...
5      ReconstructionV13(optimize, ...
6      frequencyResolution, delayRange, ...
7      bands, weights, optimizationPoints, ...
8      scenario, ...
9      pathD, pathP, ...
10     folderPath, ...
11     tag, iterationsPerCycle)
12 % Initialize output variables.
13 pulseRecovered = 0;
14 traceRecovered = 0;
15 
16 % Boolean for testing. Set to 1 if you want to use a known pulse and see
17 % how well the program retrieves it.
18 testing = 0;
19 
20 % Load the trace and the relevant parameters.
21 load(pathD);
22 load(pathP);
23 
24 % ------------------------------------------------------------------------
25 % ----------------------------------PREPARE DATA--------------------------
26 % ------------------------------------------------------------------------
27 % Extract the trace and the corresponding axes.
28 [traceOriginal, delays, frequencies, NF, frequencyIndicesF] ...
29     = ProcessData(traceAfterInitialProcessing, ...
30     wavelengthMin, wavelengthMax, ...
31     delayMin, delayMax, ...
32     frequencyResolution, ...
33     delayRange, ...
34     fMin, fMax, ...
35     deltaPulse, deltaDelay, ...
36     centerDelay);
37 
```

```
% Carrier frequency.
omega_0 = 2*pi * centerFrequency / 3;

% ------------------------------------------------------------------------
% Initial guess for pulse
% ------------------------------------------------------------------------
NoPositiveDelaysForPulse = floor(length(delays)/2);
pulseDelays = -NoPositiveDelaysForPulse*deltaPulse: deltaPulse : ...
    NoPositiveDelaysForPulse*deltaPulse;
[amplitude, phi] = PulseShapeFunction(pulseDelays, rise, fall);
pulseInitialGuess = ComplexPulse(pulseDelays, omega_0, amplitude, phi);

% Format the pulse used in testing scenario.
pulseOriginal = 0;

% ++++++++++++++++++++++++++++++TEST PULSE++++++++++++++++++++++++++++++++
if testing
    amplitude = amplitude.^2 .* (rand(1,numel(amplitude))+1);
    temp = linspace(pulseDelays(1),pulseDelays(end),20);
    amplitude = interp1(pulseDelays,amplitude,temp,'spline');
    amplitude = interp1(temp, amplitude, pulseDelays);
    amplitude = amplitude / max(amplitude);
    % phi = rand(1,numel(phi))/3 + pi;
    phi = linspace(-1,1,numel(phi)) + pi;

    pulseOriginal = ComplexPulse(pulseDelays, omega_0, amplitude, phi);
end
% ++++++++++++++++++++++++++++++/TEST PULSE++++++++++++++++++++++++++++++++

% ------------------------------------------------------------------------
% Data for the fourier transforms.
% ------------------------------------------------------------------------
% FOR FT ALONG TIME DELAY-AXIS (X)

% BASEBAND-------------------------

% Let's assume that we need L positive frequencies for the band retrival.
L = 15;

% We are only interested in baseband frequencies, so let's select them.
m = 0; % m=1 for one harmonic band and so forth.
delayFrequencyMax = (m + 0.3) * 1/tCycle;
delayFrequencyMin = 0;
delayFrequencyRange = delayFrequencyMax - delayFrequencyMin;

xiD = delayFrequencyRange / L; % frequency step

ND = ceil(1/(deltaDelay*xiD));
xiD = 1/(deltaDelay*ND); % update the frequency step

% The indices for wanted frequencies are... (fN = n*xi)
nMax = floor(delayFrequencyMax / xiD);
% Only non-negative delay frequencies wanted for the baseband
% extraction when transforming to Fourier domain. Negative frequencies are
% obtained by using the positive frequencies.
nMin = 0;
n = nMin:nMax;
frequencyIndicesD = 1:numel(frequencyIndicesF);

k = 1:length(delays);
if length(k) > ND
    disp('Problem officer: length(k) > N. Reduce delay range???')
    return
end

% These are for the transform into Fourier domain in the time delay-axis.
```

```
[sineData_2{1}, cosineData_2{1}] = SineCosineData_knN(n,k,ND);

% Both negative and positive frequencies are required when transforming
% back. Check notebook for explanation.
nMin = -nMax;
n = nMin:nMax;
[sineData_3{1}, cosineData_3{1}] = SineCosineData_knN(k,n,ND);

% FIRST & SECOND HARMONIC------------------------------
for m = 1:2 % m = 1 for first harmonic band and so forth.
    delayFrequencyMax = (m + 0.4) * 1/tCycle;
    delayFrequencyMin = (m - 0.4) * 1/tCycle;

    % The indices for wanted frequencies are... (fN = n*xi)
    nMax = floor(delayFrequencyMax / xiD);

    % Only non-negative delay frequencies wanted for the band extraction
    % when transforming to Fourier domain. Negative frequencies are
    % discarded.
    nMin = floor(delayFrequencyMin / xiD);
    n = nMin:nMax;

    % These are for the transform into Fourier domain in the time
    % delay-axis.
    [sineData_2{m + 1}, cosineData_2{m + 1}] = ...
        SineCosineData_knN(n,k,ND);

    % Both negative and positive frequencies are required when transforming
    % back.
    n = [-fliplr(n) n];
    [sineData_3{m + 1}, cosineData_3{m + 1}] = ...
        SineCosineData_knN(k,n,ND);
end

%-------------------------------------------------------------------------
% Trace for initial guess pulse.
%-------------------------------------------------------------------------
% Integer delays, used in calculation of Trace.
if mod(numel(delays),2) == 0
    d0 = find(delays==0);
    integerDelays = -(d0 - 1): numel(delays) - d0;
else
    integerDelays = -floor(numel(delays)/2) : floor(numel(delays)/2);
end

% Alakazam!
integerDelays = integerDelays*2;

if testing
    traceOriginal = TraceFun(pulseOriginal, delays, ...
        division, ...
        integerDelays, ...
    NF, frequencyIndicesF);
end

% Trace of the initial guess pulse.
traceInitialGuess = TraceFun(pulseInitialGuess, delays, ...
    division, ...
    integerDelays, ...
    NF, frequencyIndicesF);

% Extract bands.
[basebandOriginal, harm1Original, harm2Original] = ExtractBandsFast ...
    (traceOriginal, ...
    frequencyIndicesD, ...
    sineData_2, cosineData_2, ...
```

```
170     sineData_3, cosineData_3, ...
171     [1 1 1]);
172
173 [basebandInitialGuess, harm1InitialGuess, harm2InitialGuess] = ...
174     ExtractBandsFast ...
175     (traceInitialGuess, ...
176     frequencyIndicesD, ...
177     sineData_2, cosineData_2, ...
178     sineData_3, cosineData_3, ...
179     [1 1 1]);
180
181 % Errors.
182 errorTraceInitial = TraceErrorFunction(traceOriginal, ...
183     traceInitialGuess);
184 errorBasebandInitial = TraceErrorFunction(basebandOriginal, ...
185     basebandInitialGuess);
186 errorHarm1Initial = TraceErrorFunction(harm1Original, ...
187     harm1InitialGuess);
188 errorHarm2Initial = TraceErrorFunction(harm2Original, ...
189     harm2InitialGuess);
190
191 pulseInput = pulseInitialGuess;
192
193 %--------------------------------------------------------------------------
194 %-----------------------------------PLOTTING-------------------------------
195 %--------------------------------------------------------------------------
196
197 hOriginalAndGuessTraces = figure;
198
199 % Original trace.
200 subplot(2,4,1)
201 imagesc(delays*1e15,frequencies*1e-12,traceOriginal)
202 set(gca,'ydir','normal')
203 xlabel('Time delay [fs]')
204 ylabel('Frequency [THz]')
205 title([scenario ' Original trace'])
206
207 % Initial guess pulse trace.
208 subplot(2,4,5)
209 imagesc(delays*1e15,frequencies*1e-12,traceInitialGuess)
210 set(gca,'ydir','normal')
211 xlabel('Time delay [fs]')
212 ylabel('Frequency [THz]')
213 title(['Trace for Initial Guess Pulse, \delta = ' ...
214     num2str(errorTraceInitial,6)])
215
216 % Original baseband.
217 subplot(2,4,2)
218 imagesc(delays*1e15,frequencies*1e-12,basebandOriginal)
219 set(gca,'ydir','normal')
220 xlabel('Time delay [fs]')
221 ylabel('Frequency [THz]')
222 title('Original baseband')
223
224 % Initial guess pulse baseband.
225 subplot(2,4,6)
226 imagesc(delays*1e15,frequencies*1e-12,basebandInitialGuess)
227 set(gca,'ydir','normal')
228 xlabel('Time delay [fs]')
229 ylabel('Frequency [THz]')
230 title(['Baseband for IGP, \delta = ' ...
231     num2str(errorBasebandInitial,6)])
232
233 % Original first harmonic band.
234 subplot(2,4,3)
235 imagesc(delays*1e15,frequencies*1e-12,harm1Original)
```

```
236 set(gca,'ydir','normal')
237 xlabel('Time delay [fs]')
238 ylabel('Frequency [THz]')
239 title('Original FM-band')
240
241 % Initial guess pulse first harmonic band.
242 subplot(2,4,7)
243 imagesc(delays*1e15,frequencies*1e-12,harm1InitialGuess)
244 set(gca,'ydir','normal')
245 xlabel('Time delay [fs]')
246 ylabel('Frequency [THz]')
247 title(['FM-band for IGP, \delta = ' ...
248     num2str(errorHarm1Initial,6)])
249
250 % Original first harmonic band.
251 subplot(2,4,4)
252 imagesc(delays*1e15,frequencies*1e-12,harm2Original)
253 set(gca,'ydir','normal')
254 xlabel('Time delay [fs]')
255 ylabel('Frequency [THz]')
256 title('Original 2nd harm. band')
257
258 % Initial guess pulse first harmonic band.
259 subplot(2,4,8)
260 imagesc(delays*1e15,frequencies*1e-12,harm2InitialGuess)
261 set(gca,'ydir','normal')
262 xlabel('Time delay [fs]')
263 ylabel('Frequency [THz]')
264 title(['2nd harm. band for IGP, \delta = ' ...
265     num2str(errorHarm2Initial,6)])
266
267 s = get(0, 'ScreenSize');
268 set(hOriginalAndGuessTraces,'Position', [0 0 s(3) s(4)])
269
270 %-------------------------------------------------------------------------
271 %-----------------------------OPTIMIZATION--------------------------------
272 %-------------------------------------------------------------------------
273 if(optimize == 1)
274     % Optimize.
275     [pulseRecovered, traceRecovered, hRetrieval] = ...
276         RecoverPulse( ...
277         delays, pulseDelays, integerDelays, ...
278         frequencies, frequencyIndicesF, frequencyIndicesD, ...
279         NF, omega_0, division, ...
280         pulseInput, pulseOriginal, ...
281         traceOriginal, basebandOriginal, harm1Original, harm2Original, ...
282         sineData_2, cosineData_2, ...
283         sineData_3, cosineData_3, ...
284         bands, weights, optimizationPoints,iterationsPerCycle);
285
286     % Extract bands.
287     [basebandRecovered, harm1Recovered, harm2Recovered] = ...
288     ExtractBandsFast ...
289     (traceRecovered, ...
290     frequencyIndicesD, ...
291     sineData_2, cosineData_2, ...
292     sineData_3, cosineData_3, ...
293     [1 1 1]);
294
295     % Errors with the error function.
296     errorTraceRecovered = TraceErrorFunction(traceOriginal, ...
297         traceRecovered);
298     errorBasebandRecovered = TraceErrorFunction(basebandOriginal, ...
299         basebandRecovered);
300     errorHarm1Recovered = TraceErrorFunction(harm1Original, ...
301         harm1Recovered);
```

```
302     errorHarm2Recovered = TraceErrorFunction(harm2Original, ...
303         harm2Recovered);
304
305     % ———————————————————————————————————————————————————————————
306     % PLOTTING
307     % ———————————————————————————————————————————————————————————
308
309     % ERROR PLOTS OF THE TRACES
310     hErrors = figure;
311
312     % Recovered trace error.
313     subplot(2,2,1)
314     errorMatrixTrace = abs(traceRecovered – traceOriginal);
315     imagesc(delays*1e15, frequencies*1e–12, errorMatrixTrace)
316     set(gca,'ydir','normal')
317     colorbar
318     title('Trace error')
319     xlabel('Time delay [fs]')
320     ylabel('Frequency [THz]')
321
322     % Recovered baseband error.
323     subplot(2,2,2)
324     errorMatrixBaseband = abs(basebandRecovered – basebandOriginal);
325     imagesc(delays*1e15, frequencies*1e–12, errorMatrixBaseband)
326     set(gca,'ydir','normal')
327     colorbar
328     title('Baseband error')
329     xlabel('Time delay [fs]')
330     ylabel('Frequency [THz]')
331
332     % Recovered first harmonic band error.
333     subplot(2,2,3)
334     errorMatrixHarm1 = abs(harm1Recovered – harm1Original);
335     imagesc(delays*1e15, frequencies*1e–12, errorMatrixHarm1)
336     set(gca,'ydir','normal')
337     colorbar
338     title('FM–band error')
339     xlabel('Time delay [fs]')
340     ylabel('Frequency [THz]')
341
342     % Recovered second harmonic band error.
343     subplot(2,2,4)
344     errorMatrixHarm2 = abs(harm2Recovered – harm2Original);
345     imagesc(delays*1e15, frequencies*1e–12, errorMatrixHarm2)
346     set(gca,'ydir','normal')
347     colorbar
348     title('2nd harm. band error')
349     xlabel('Time delay [fs]')
350     ylabel('Frequency [THz]')
351
352     s = get(0, 'ScreenSize');
353     set(hErrors,'Position', [0 0 s(3) s(4)])
354
355     % Plot the absolute squares and phases of the pulses.
356     hPulses = figure;
357     s = get(0, 'ScreenSize');
358     set(hPulses,'Position', [0 0 s(3) s(4)])
359
360     subplot(1,2,1)
361     if testing
362         plot(pulseDelays*1e15,abs(pulseInitialGuess).^2, 'b*', ...
363             pulseDelays*1e15,abs(pulseRecovered).^2, 'r', ...
364             pulseDelays*1e15,abs(pulseOriginal).^2, 'g—')
365     else
366         plot(pulseDelays*1e15,abs(pulseInitialGuess).^2, 'b*', ...
367             pulseDelays*1e15,abs(pulseRecovered).^2, 'r')
```

```
368     end
369     title('Pulse intensity')
370     xlabel('Time delay [fs]')
371     ylabel('Normalized intensity')
372     set(gca,'ylim',[0 1.05])
373 
374     subplot(1,2,2)
375     phiInitialGuess = ...
376         ExtractPhase(pulseDelays, omega_0, pulseInitialGuess);
377     phiRecovered = ...
378         ExtractPhase(pulseDelays, omega_0, pulseRecovered);
379 
380     if testing
381         phiOriginal = ...
382             ExtractPhase(pulseDelays, omega_0, pulseOriginal);
383         plot(pulseDelays*1e15,phiInitialGuess, 'b*', ...
384             pulseDelays*1e15,phiRecovered, 'r', ...
385             pulseDelays*1e15,phiOriginal, 'g--')
386         legend('Initial guess','Recovered','Original','Location','Best')
387     else
388         plot(pulseDelays*1e15,phiInitialGuess, 'b*', ...
389             pulseDelays*1e15,phiRecovered,'r')
390         legend('Initial guess','Recovered','Location','Best')
391     end
392     title('Phase')
393     xlabel('Time delay [fs]')
394     ylabel('Phase [rad]')
395 
396 end
397 
398 %--------------------------------------------------------------------------
399 %-----------------------------SAVE DATA------------------------------------
400 %--------------------------------------------------------------------------
401 if optimize == 1;
402     % Date.
403     timeStamp = datestr(now, 'yyyy-mm-dd');
404 
405     % First part of the filename, second would be the specifying title,
406     % i.e. 'data.mat'.
407     filename = [folderPath timeStamp '_' scenario '_' tag '_'];
408 
409     % SAVE DATA ON .mat FILE
410     hWaitbar = waitbar(0,'Saving data...');
411 
412     save([filename 'Data.mat'],  ...
413         'delays', 'pulseDelays', 'frequencies', ...
414         'pulseRecovered', ...
415         'traceOriginal', 'traceRecovered', ...
416         'basebandOriginal', 'basebandRecovered', ...
417         'harm1Original', 'harm1Recovered', ...
418         'harm2Original', 'harm2Recovered', ...
419         'bands' ,'weights', 'division', ...
420         'errorTraceRecovered', 'errorBasebandRecovered', ...
421         'errorHarm1Recovered', 'errorHarm2Recovered', ...
422         'errorTraceInitial', 'errorBasebandInitial', ...
423         'errorHarm1Initial', 'errorHarm2Initial')
424 
425     % SAVE FIGURES
426     progress = 0.1;
427     waitbar(progress, hWaitbar, 'Saving figures...')
428 
429     h = hOriginalAndGuessTraces;
430     set(h,'PaperOrientation','landscape');
431     set(h,'PaperUnits','normalized');
432     set(h,'PaperPosition', [0 0 1 1]);
433     print(hOriginalAndGuessTraces, '-dpdf', ...
```

```
434         [filename 'OriginalAndGuessTraces'])
435
436     h = hRetrieval;
437     set(h,'PaperOrientation','landscape');
438     set(h,'PaperUnits','normalized');
439     set(h,'PaperPosition', [0 0 1 1]);
440     print(hRetrieval, '-dpdf', ...
441         [filename 'Retrieval'])
442
443     h = hErrors;
444     set(h,'PaperOrientation','landscape');
445     set(h,'PaperUnits','normalized');
446     set(h,'PaperPosition', [0 0 1 1]);
447     print(hErrors, '-dpdf', ...
448         [filename 'Errors'])
449
450     h = hPulses;
451     set(h,'PaperOrientation','landscape');
452     set(h,'PaperUnits','normalized');
453     set(h,'PaperPosition', [0 0 1 1]);
454     print(hPulses, '-dpdf', ...
455         [filename 'RecoveredPulse'])
456
457     % SAVE DATA ON TXT FILES
458
459     progress = progress + 0.2;
460     waitbar(progress, hWaitbar, 'Saving txt data...')
461
462     % Print pulse on a file.
463     pulseDelaysOut = pulseDelays*1e15; % fs
464     pulseOut = pulseRecovered;
465     outputForPulse = [pulseDelaysOut; real(pulseOut); imag(pulseOut)];
466
467     fileID = fopen([filename 'Pulse.txt'],'w');
468     fprintf(fileID,'%11s %11s %11s \n', ...
469         'delays [fs]', 'Real part', 'Imag part');
470     fprintf(fileID,'%6.3f\t%6.4f\t%6.4f\n', ...
471         outputForPulse );
472     fclose(fileID);
473
474     progress = progress + 0.1;
475     waitbar(progress, hWaitbar, 'Saving txt data...')
476     % Print trace delays on a file.
477     delaysOut = delays*1e15; % fs
478
479     fileID = fopen([filename 'Delays.txt'],'w');
480     fprintf(fileID,'%s \n', ...
481         'delays [fs]');
482     fprintf(fileID,'%6.3f\n', ...
483         delaysOut );
484     fclose(fileID);
485
486     progress = progress + 0.1;
487     waitbar(progress, hWaitbar, 'Saving txt data...')
488     % Print frequencies on a file.
489     frequenciesOut = frequencies*1e-12; % THz
490
491     fileID = fopen([filename 'Freqs.txt'],'w');
492     fprintf(fileID,'%s \n', ...
493         'frequencies [THz]');
494     fprintf(fileID,'%7.2f\n', ...
495         frequenciesOut );
496     fclose(fileID);
497
498     progress = progress + 0.1;
499     waitbar(progress, hWaitbar, 'Saving txt data...')
```

```
500     % Print original trace on a file.
501     dlmwrite([filename 'TraceOriginal.txt'], traceOriginal, '\t')
502 
503     progress = progress + 0.1;
504     waitbar(progress, hWaitbar, 'Saving txt data...')
505     % Print recovered trace on a file.
506     dlmwrite([filename 'TraceRecovered.txt'], traceRecovered, '\t')
507 
508     waitbar(1, hWaitbar, 'Done!')
509     pause(1)
510     close(hWaitbar)
511 
512 end
513 disp('The program has finished.')
514 msgbox('The program has finished.', 'All done!');
515 end
```

A.16 RecoverPulse.m

```
1  % AUTOMATED VERSION Recovers the original pulse through optimization.
2  function [pulseRecovered, traceRecovered, hRetrieval] = ...
3      RecoverPulse ...
4      (delays, pulseDelays, integerDelays, ...
5      frequencies, frequencyIndicesF, frequencyIndicesD, ...
6      NF, omega0, division, ...
7      pulseGuess, pulseOriginal, ...
8      traceOriginal, baseOriginal, harm1Original, harm2Original, ...
9      sineData2, cosineData2, ...
10     sineData3, cosineData3, ...
11     bands, weights, optimizationPoints,iterationsPerCycle)
12 
13 % Measure how long the optimization takes.
14 tstart = tic;
15 
16 % Estimate the envelope at XX delay points.
17 sparseDelays = linspace(pulseDelays(1),pulseDelays(end),optimizationPoints);
18 
19 % Extract phase and amplitude of the guess pulse.
20 amplitude = abs(pulseGuess);
21 phi = ExtractPhase(pulseDelays, omega0, pulseGuess);
22 
23 % Sample.
24 amplitudeInput = interp1(pulseDelays, amplitude, sparseDelays, 'spline');
25 phiInput = interp1(pulseDelays, phi, sparseDelays, 'spline');
26 
27 % Input pulse is provided in this format.
28 pulseInput = [amplitudeInput; phiInput];
29 
30 % Function to be minimized.
31 objfun = @(pulseAmpPhase) NormFunction(pulseAmpPhase, ...
32     delays, pulseDelays, sparseDelays, ...
33     baseOriginal, harm1Original, harm2Original, ...
34     division, frequencyIndicesD, ...
35     sineData2, cosineData2, ...
36     sineData3, cosineData3, integerDelays, omega0, ...
37     NF, frequencyIndicesF, ...
38     bands, weights);
39 
40 disp('Optimization follows.')
41 
42 pause(0.1)
43 
44 % Create a large figure for retrieval plots.
45 hRetrieval = figure;
```

```
46 s = get(0, 'ScreenSize');
47 set(hRetrieval,'Position', [0 0.5*s(4) s(3) 0.5*s(4)])
48
49 % Add pushbuttons to figure for user control.
50 hButtonPushStop = uicontrol('Style', 'pushbutton', 'String', 'Stop',...
51     'Position', [10 10 50 20],...
52     'Callback', @ButtonPushStop);
53
54 hButtonPushIterations = ...
55     uicontrol('Style', 'pushbutton', 'String', 'Change number of iterations',...
56     'Position', [70 10 170 20],...
57     'Callback', @ButtonPushIterations);
58
59 % Plots for original trace and its baseband.
60 subplot(2,3,1)
61 imagesc(delays*1e15, frequencies*1e-12, traceOriginal);
62 set(gca,'ydir','normal')
63 title('Original trace')
64 xlabel('Time delay [fs]')
65 ylabel('Frequency [THz]')
66
67 subplot(2,3,4)
68 imagesc(delays*1e15, frequencies*1e-12, baseOriginal)
69 set(gca,'ydir','normal')
70 title('Baseband of original trace')
71 xlabel('Time delay [fs]')
72 ylabel('Frequency [THz]')
73
74 global stop;
75 stop = 0;
76
77 drawnow
78
79 pause(0.1)
80
81 % Set maximum number of iterations per cycle. This can be increased in the
82 % UI.
83 global maxIter;
84 maxIter = iterationsPerCycle;
85
86 error = 0;
87 lastError = inf;
88
89 % Optimization loop.
90 while ~stop && lastError ~= error
91     lastError = error;
92
93     options = optimset('Display','iter', ...
94         'MaxIter', maxIter);
95     [pulseRecovered, error] = fminsearch(objfun, ...
96         pulseInput, ...
97         options);
98
99     pulseInput = pulseRecovered;
100
101     amplitude = interp1(sparseDelays, pulseRecovered(1,:), pulseDelays);
102     phi = interp1(sparseDelays, pulseRecovered(2,:), pulseDelays);
103
104     % Convert the pulse into complex form.
105     pulseRecovered = ComplexPulse(pulseDelays, omega0, amplitude, phi);
106
107     % Normalize the pulse.
108     pulseRecovered = pulseRecovered/max(abs(pulseRecovered));
109
110     % Calculate the normalized trace.
111     traceRecovered = TraceFun(pulseRecovered, delays, division, ...
```

```
112         integerDelays, ...
113         NF, frequencyIndicesF);
114 
115     % Extract the baseband.
116     baseFit = ExtractBasebandFast(traceRecovered, frequencyIndicesD, ...
117         sineData2{1}, cosineData2{1}, ...
118         sineData3{1}, cosineData3{1});
119 
120     figure(hRetrieval)
121 
122     subplot(2,3,2)
123     imagesc(delays*1e15, frequencies*1e-12, traceRecovered)
124     set(gca,'ydir','normal')
125     title(['Trace fit, weighted error = ' num2str(error,6)])
126     xlabel('Time delay [fs]')
127     ylabel('Frequency [THz]')
128 
129     subplot(2,3,5)
130     imagesc(delays*1e15, frequencies*1e-12, baseFit)
131     set(gca,'ydir','normal')
132     title('Baseband fit')
133     xlabel('Time delay [fs]')
134     ylabel('Frequency [THz]')
135 
136     PlotPulses(pulseDelays, omega0, ...
137         pulseGuess, pulseRecovered, pulseOriginal)
138 
139     drawnow
140 
141     % Round error.
142     error = fix(error*10^8)/10^8;
143 end
144 
145 set(hButtonPushStop, 'Enable', 'Off')
146 set(hButtonPushIterations, 'Enable', 'Off')
147 
148 disp('Optimization done.')
149 
150 % Display how long the optimization took.
151 telapsed = toc(tstart);
152 disp(['Optimization took '  num2str(telapsed) ' seconds.'])
153 end
154 
155 function ButtonPushStop(hObj,event) %#ok<INUSD>
156 global stop;
157 stop = 1;
158 end
159 
160 function ButtonPushIterations(hObj,event) %#ok<INUSD>
161 global maxIter;
162 ok = 0;
163 while ~ok
164     prompt = {'Enter number of iterations before figures are drawn.'};
165     dlgTitle = 'Maximum iterations';
166     numLines = 1;
167     def = {num2str(maxIter)};
168     maxIter = inputdlg(prompt,dlgTitle,numLines,def);
169     maxIter = str2num(maxIter{1});
170 
171     % Error check.
172     if length(maxIter)~=1 || mod(maxIter,1)~=0
173         errordlg('Please provide an integer number.')
174     else
175         ok = 1;
176     end
177 end
```

```
178 end
179
180 function PlotPulses(pulseDelays, omega0, ...
181     pulseGuess, pulseFit, pulseOriginal)
182
183 subplot(2,3,3)
184
185 if length(pulseOriginal) == 1 % No original pulse involved (not testing)
186     plot(pulseDelays*1e15,abs(pulseGuess).^2, 'b*', ...
187         pulseDelays*1e15,abs(pulseFit).^2, 'r')
188 else
189     plot(pulseDelays*1e15,abs(pulseGuess).^2, 'b*', ...
190         pulseDelays*1e15,abs(pulseFit).^2, 'r',...
191         pulseDelays*1e15,abs(pulseOriginal).^2, 'g--')
192 end
193 title('Intensity')
194 xlabel('Time delay [fs]')
195 ylabel('Normalized intensity')
196 set(gca,'ylim',[0 1.05])
197
198 subplot(2,3,6)
199
200 phiGuess = ...
201     ExtractPhase(pulseDelays, omega0, pulseGuess);
202 phiFit = ...
203     ExtractPhase(pulseDelays, omega0, pulseFit);
204 if length(pulseOriginal) ~= 1 % Original pulse involved (testing)
205     phiOriginal = ...
206         ExtractPhase(pulseDelays, omega0, pulseOriginal);
207     plot(pulseDelays*1e15,phiGuess, 'b*', ...
208         pulseDelays*1e15,phiFit, 'r', ...
209         pulseDelays*1e15,phiOriginal, 'g--')
210     legend('Initial guess','Recovered','Original','Location','Best')
211 else
212     plot(pulseDelays*1e15,phiGuess, 'b*', ...
213         pulseDelays*1e15,phiFit, 'r')
214         legend('Initial guess','Recovered','Location','Best')
215 end
216 title('Phase')
217 xlabel('Time delay [fs]')
218 ylabel('Phase [rad]')
219 end
```

A.17 RemoveArtefacts.fig

```
1  % REMOVEARTEFACTS MATLAB code for RemoveArtefacts.fig
2  function varargout = RemoveArtefacts(varargin)
3  % Begin initialization code - DO NOT EDIT
4  gui_Singleton = 1;
5  gui_State = struct('gui_Name',       mfilename, ...
6                     'gui_Singleton',  gui_Singleton, ...
7                     'gui_OpeningFcn', @RemoveArtefacts_OpeningFcn, ...
8                     'gui_OutputFcn',  @RemoveArtefacts_OutputFcn, ...
9                     'gui_LayoutFcn',  [] , ...
10                    'gui_Callback',   []);
11 if nargin && ischar(varargin{1})
12     gui_State.gui_Callback = str2func(varargin{1});
13 end
14
15 if nargout
16     [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
17 else
18     gui_mainfcn(gui_State, varargin{:});
19 end
```

```
20 % End initialization code - DO NOT EDIT
21
22 % --- Executes just before RemoveArtefacts is made visible.
23 function RemoveArtefacts_OpeningFcn(hObject, eventdata, handles, varargin)
24 % This function has no output args, see OutputFcn.
25 % hObject    handle to figure
26 % eventdata  reserved - to be defined in a future version of MATLAB
27 % handles    structure with handles and user data (see GUIDATA)
28 % varargin   command line arguments to RemoveArtefacts (see VARARGIN)
29
30 % Choose default command line output for RemoveArtefacts
31 handles.output = hObject;
32
33 % Initialize variables
34 handles.data{1} = 0;
35 handles.currentDataIndex = 1;
36 set(handles.eBrushSizeX,'value',1)
37 set(handles.eBrushSizeY,'value',1)
38 set(handles.eBrushSizeX,'string',num2str(get(handles.eBrushSizeX,'value')))
39 set(handles.eBrushSizeY,'string',num2str(get(handles.eBrushSizeX,'value')))
40
41 % Update handles structure
42 guidata(hObject, handles);
43
44 % Handle no structure being passed to GUI
45 if nargin<4
46  handles.fileNameOriginal = 'Default String';
47 else
48     fileNameOriginal = varargin{1}{1};
49     handles.fileNameOriginal = varargin{1}{1};
50 end
51
52 % Load data.
53 data = load(fileNameOriginal);
54 % handles.data{1} = data.';
55 handles.data{1} = rot90(data);
56 imagesc(handles.data{1})
57 set(gca,'YDir','normal')
58
59 % Update handles structure
60 guidata(hObject, handles);
61
62 % UIWAIT makes RemoveArtefacts wait for user response (see UIRESUME)
63 uiwait(handles.figure1);
64
65 % --- Outputs from this function are returned to the command line.
66 function varargout = RemoveArtefacts_OutputFcn(hObject, eventdata, handles)
67 % Output data (the trace with artefacts removed)
68 varargout{1} = handles.output;
69
70 % Clean up.
71 delete(handles.figure1)
72
73 % --- Executes on button press in pbUndo.
74 function pbUndo_Callback(hObject, eventdata, handles)
75 % hObject    handle to pbUndo (see GCBO)
76 % eventdata  reserved - to be defined in a future version of MATLAB
77 % handles    structure with handles and user data (see GUIDATA)
78 if handles.currentDataIndex ~= 1
79     handles.currentDataIndex = handles.currentDataIndex - 1;
80     guidata(hObject, handles);
81
82     xLim = get(gca,'xlim');
83     yLim = get(gca,'ylim');
84     hold off
85     imagesc(handles.data{handles.currentDataIndex});set(gca,'YDir','normal')
```

```
86      set(gca,'xlim',xLim)
87      set(gca,'ylim',yLim)
88      hold on
89      plot(handles.levelX,handles.levelY,'gx','markersize',40)
90  end
91
92  % ——— Executes on button press in pbSelectLevel.
93  function pbSelectLevel_Callback(hObject, eventdata, handles)
94  [x,y] = ginput(1);
95  levelX = floor(x);
96  levelY = floor(y);
97  handles.level = handles.data{1}(levelY,levelX);
98  handles.levelX = levelX;
99  handles.levelY = levelY;
100
101 xLim = get(gca,'xlim');
102 yLim = get(gca,'ylim');
103 hold off
104 imagesc(handles.data{handles.currentDataIndex});set(gca,'YDir','normal')
105 set(gca,'xlim',xLim)
106 set(gca,'ylim',yLim)
107 hold on
108 handles.hLevelCross = plot(levelX,levelY,'gx','markersize',40);
109 guidata(hObject, handles);
110
111 % ——— Executes on button press in pbRedo.
112 function pbRedo_Callback(hObject, eventdata, handles)
113 if handles.currentDataIndex ~= size(handles.data,2)
114     handles.currentDataIndex = handles.currentDataIndex + 1;
115     guidata(hObject, handles);
116
117     xLim = get(gca,'xlim');
118     yLim = get(gca,'ylim');
119     hold off
120     imagesc(handles.data{handles.currentDataIndex});set(gca,'YDir','normal')
121     set(gca,'xlim',xLim)
122     set(gca,'ylim',yLim)
123     hold on
124     plot(handles.levelX,handles.levelY,'gx','markersize',40)
125 end
126
127 function eBrushSizeX_Callback(hObject, eventdata, handles)
128 set(hObject,'Value',str2num(get(hObject,'string')));
129
130 % ——— Executes during object creation, after setting all properties.
131 function eBrushSizeX_CreateFcn(hObject, eventdata, handles)
132 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
133     set(hObject,'BackgroundColor','white');
134 end
135
136 function eBrushSizeY_Callback(hObject, eventdata, handles)
137 set(hObject,'Value',str2num(get(hObject,'string')));
138
139 % ——— Executes during object creation, after setting all properties.
140 function eBrushSizeY_CreateFcn(hObject, eventdata, handles)
141 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
142     set(hObject,'BackgroundColor','white');
143 end
144
145 % ——— Executes on button press in pbBrush.
146 function pbBrush_Callback(hObject, eventdata, handles)
147 [x,y] = ginput(1);
148 x = round(x);
149 y = round(y);
150
151 brushSizeX = get(handles.eBrushSizeX,'Value');
```

```
152 brushSizeY = get(handles.eBrushSizeY,'Value');
153
154 editRangeX = x - floor(brushSizeX/2) : x + floor(brushSizeX/2);
155 editRangeY = y - floor(brushSizeY/2) : y + floor(brushSizeY/2);
156
157 dataEdited = handles.data{handles.currentDataIndex};
158
159 % Make sure that indices are applicable
160 editRangeX = editRangeX(editRangeX>0);
161 editRangeY = editRangeY(editRangeY>0);
162 editRangeX = editRangeX(editRangeX<=size(dataEdited,2));
163 editRangeY = editRangeY(editRangeY<=size(dataEdited,1));
164
165 dataEdited(editRangeY,editRangeX) = handles.level;
166
167 handles.data{handles.currentDataIndex + 1} = dataEdited;
168 handles.currentDataIndex = handles.currentDataIndex + 1;
169 handles.data = handles.data(1:handles.currentDataIndex);
170 guidata(hObject, handles);
171
172 xLim = get(gca,'xlim');
173 yLim = get(gca,'ylim');
174 hold off
175 imagesc(dataEdited);set(gca,'YDir','normal')
176 set(gca,'xlim',xLim)
177 set(gca,'ylim',yLim)
178 hold on
179 plot(handles.levelX,handles.levelY,'gx','markersize',40)
180
181 % --- Executes on button press in cbZoom.
182 function cbZoom_Callback(hObject, eventdata, handles)
183 if get(hObject,'Value')
184     zoom on
185 else
186     zoom off
187 end
188
189 % --- Executes when user attempts to close figure1.
190 function figure1_CloseRequestFcn(hObject, eventdata, handles)
191 handles.output = handles.data{handles.currentDataIndex};
192 guidata(hObject, handles);
193
194 % The GUI is still in UIWAIT, us UIRESUME
195 uiresume(handles.figure1);
```

A.18 RemoveArtefacts.m

```
1  % REMOVEARTEFACTS MATLAB code for RemoveArtefacts.fig
2  function varargout = RemoveArtefacts(varargin)
3  % Begin initialization code - DO NOT EDIT
4  gui_Singleton = 1;
5  gui_State = struct('gui_Name',       mfilename, ...
6                     'gui_Singleton',  gui_Singleton, ...
7                     'gui_OpeningFcn', @RemoveArtefacts_OpeningFcn, ...
8                     'gui_OutputFcn',  @RemoveArtefacts_OutputFcn, ...
9                     'gui_LayoutFcn',  [] , ...
10                    'gui_Callback',   []);
11 if nargin && ischar(varargin{1})
12     gui_State.gui_Callback = str2func(varargin{1});
13 end
14
15 if nargout
16     [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
17 else
```

```
18      gui_mainfcn(gui_State, varargin{:});
19  end
20  % End initialization code - DO NOT EDIT
21
22  % --- Executes just before RemoveArtefacts is made visible.
23  function RemoveArtefacts_OpeningFcn(hObject, eventdata, handles, varargin)
24  % This function has no output args, see OutputFcn.
25  % hObject    handle to figure
26  % eventdata  reserved - to be defined in a future version of MATLAB
27  % handles    structure with handles and user data (see GUIDATA)
28  % varargin   command line arguments to RemoveArtefacts (see VARARGIN)
29
30  % Choose default command line output for RemoveArtefacts
31  handles.output = hObject;
32
33  % Initialize variables
34  handles.data{1} = 0;
35  handles.currentDataIndex = 1;
36  set(handles.eBrushSizeX,'value',1)
37  set(handles.eBrushSizeY,'value',1)
38  set(handles.eBrushSizeX,'string',num2str(get(handles.eBrushSizeX,'value')))
39  set(handles.eBrushSizeY,'string',num2str(get(handles.eBrushSizeX,'value')))
40
41  % Update handles structure
42  guidata(hObject, handles);
43
44  % Handle no structure being passed to GUI
45  if nargin<4
46   handles.fileNameOriginal = 'Default String';
47  else
48      fileNameOriginal = varargin{1}{1};
49      handles.fileNameOriginal = varargin{1}{1};
50  end
51
52  % Load data.
53  data = load(fileNameOriginal);
54  % handles.data{1} = data.';
55  handles.data{1} = rot90(data);
56  imagesc(handles.data{1})
57  set(gca,'YDir','normal')
58
59  % Update handles structure
60  guidata(hObject, handles);
61
62  % UIWAIT makes RemoveArtefacts wait for user response (see UIRESUME)
63  uiwait(handles.figure1);
64
65  % --- Outputs from this function are returned to the command line.
66  function varargout = RemoveArtefacts_OutputFcn(hObject, eventdata, handles)
67  % Output data (the trace with artefacts removed)
68  varargout{1} = handles.output;
69
70  % Clean up.
71  delete(handles.figure1)
72
73  % --- Executes on button press in pbUndo.
74  function pbUndo_Callback(hObject, eventdata, handles)
75  % hObject    handle to pbUndo (see GCBO)
76  % eventdata  reserved - to be defined in a future version of MATLAB
77  % handles    structure with handles and user data (see GUIDATA)
78  if handles.currentDataIndex ~= 1
79      handles.currentDataIndex = handles.currentDataIndex - 1;
80      guidata(hObject, handles);
81
82      xLim = get(gca,'xlim');
83      yLim = get(gca,'ylim');
```

```
84      hold off
85      imagesc(handles.data{handles.currentDataIndex});set(gca,'YDir','normal')
86      set(gca,'xlim',xLim)
87      set(gca,'ylim',yLim)
88      hold on
89      plot(handles.levelX,handles.levelY,'gx','markersize',40)
90  end
91
92  % ——— Executes on button press in pbSelectLevel.
93  function pbSelectLevel_Callback(hObject, eventdata, handles)
94  [x,y] = ginput(1);
95  levelX = floor(x);
96  levelY = floor(y);
97  handles.level = handles.data{1}(levelY,levelX);
98  handles.levelX = levelX;
99  handles.levelY = levelY;
100
101 xLim = get(gca,'xlim');
102 yLim = get(gca,'ylim');
103 hold off
104 imagesc(handles.data{handles.currentDataIndex});set(gca,'YDir','normal')
105 set(gca,'xlim',xLim)
106 set(gca,'ylim',yLim)
107 hold on
108 handles.hLevelCross = plot(levelX,levelY,'gx','markersize',40);
109 guidata(hObject, handles);
110
111 % ——— Executes on button press in pbRedo.
112 function pbRedo_Callback(hObject, eventdata, handles)
113 if handles.currentDataIndex ~= size(handles.data,2)
114     handles.currentDataIndex = handles.currentDataIndex + 1;
115     guidata(hObject, handles);
116
117     xLim = get(gca,'xlim');
118     yLim = get(gca,'ylim');
119     hold off
120     imagesc(handles.data{handles.currentDataIndex});set(gca,'YDir','normal')
121     set(gca,'xlim',xLim)
122     set(gca,'ylim',yLim)
123     hold on
124     plot(handles.levelX,handles.levelY,'gx','markersize',40)
125 end
126
127 function eBrushSizeX_Callback(hObject, eventdata, handles)
128 set(hObject,'Value',str2num(get(hObject,'string')));
129
130 % ——— Executes during object creation, after setting all properties.
131 function eBrushSizeX_CreateFcn(hObject, eventdata, handles)
132 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
133     set(hObject,'BackgroundColor','white');
134 end
135
136 function eBrushSizeY_Callback(hObject, eventdata, handles)
137 set(hObject,'Value',str2num(get(hObject,'string')));
138
139 % ——— Executes during object creation, after setting all properties.
140 function eBrushSizeY_CreateFcn(hObject, eventdata, handles)
141 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
142     set(hObject,'BackgroundColor','white');
143 end
144
145 % ——— Executes on button press in pbBrush.
146 function pbBrush_Callback(hObject, eventdata, handles)
147 [x,y] = ginput(1);
148 x = round(x);
149 y = round(y);
```

```
150
151 brushSizeX = get(handles.eBrushSizeX,'Value');
152 brushSizeY = get(handles.eBrushSizeY,'Value');
153
154 editRangeX = x - floor(brushSizeX/2) : x + floor(brushSizeX/2);
155 editRangeY = y - floor(brushSizeY/2) : y + floor(brushSizeY/2);
156
157 dataEdited = handles.data{handles.currentDataIndex};
158
159 % Make sure that indices are applicable
160 editRangeX = editRangeX(editRangeX>0);
161 editRangeY = editRangeY(editRangeY>0);
162 editRangeX = editRangeX(editRangeX<=size(dataEdited,2));
163 editRangeY = editRangeY(editRangeY<=size(dataEdited,1));
164
165 dataEdited(editRangeY,editRangeX) = handles.level;
166
167 handles.data{handles.currentDataIndex + 1} = dataEdited;
168 handles.currentDataIndex = handles.currentDataIndex + 1;
169 handles.data = handles.data(1:handles.currentDataIndex);
170 guidata(hObject, handles);
171
172 xLim = get(gca,'xlim');
173 yLim = get(gca,'ylim');
174 hold off
175 imagesc(dataEdited);set(gca,'YDir','normal')
176 set(gca,'xlim',xLim)
177 set(gca,'ylim',yLim)
178 hold on
179 plot(handles.levelX,handles.levelY,'gx','markersize',40)
180
181 % --- Executes on button press in cbZoom.
182 function cbZoom_Callback(hObject, eventdata, handles)
183 if get(hObject,'Value')
184     zoom on
185 else
186     zoom off
187 end
188
189 % --- Executes when user attempts to close figure1.
190 function figure1_CloseRequestFcn(hObject, eventdata, handles)
191 handles.output = handles.data{handles.currentDataIndex};
192 guidata(hObject, handles);
193
194 % The GUI is still in UIWAIT, us UIRESUME
195 uiresume(handles.figure1);
```

A.19 ShiftedPulse.m

```
1 % Creates a delayed pulse of original pulse length, padded with zeros. Used
2 % to construct the second matrix in TraceFun function.
3 function output = ShiftedPulse(pulse, delayInteger)
4 output = zeros(1, 3*length(pulse) - 1);
5 start = length(pulse) + delayInteger;
6 output(start:start + length(pulse) - 1) = pulse;
7 end
```

A.20 SineCosineData_knN.m

```
% Sine and cosine data for custom Fourier transform functions.
function [sineData, cosineData] = SineCosineData_knN(k,n,N)
temp = (n.' * k / N) * (2*pi);
sineData = sin(temp);
cosineData = cos(temp);
```

A.21 TraceErrorFunction.m

```
% Frog error. Measure for the goodnes of a fit.
function error = TraceErrorFunction(traceOriginal, traceFit)
% Calculate the normalization constant mu.
mu = sum(sum(traceOriginal .* traceFit)) / sum(sum(traceFit.^2));

% Number of points.
N = size(traceOriginal,1) * size(traceOriginal,2);

% Trebino p.160 formula.
error = sqrt( ...
    1/N * sum(sum( ...
    abs(traceOriginal - mu * traceFit).^2 ...
    )));
end
```

A.22 TraceFun.m

```
% Calculates the trace of a pulse.
function output = TraceFun(pulse, delays, division, ...
    integerDelays, ...
    NF, frequencyIndices)
% ------------------------------------------------------------------------
% Pulse matrices
% ------------------------------------------------------------------------
% Matrix for the original pulse. Including zero padding to increase data
% width to 3*pulseLength - 1
pulseLength = length(pulse);
zeroPaddedPulse = [zeros(1, pulseLength - 1), pulse, zeros(1, pulseLength)];
pulseMatrix1 = repmat(zeroPaddedPulse,numel(delays),1);

% Matrix for delayed pulse values.
pulseMatrix2 = arrayfun(@(delayInteger) ...
    ShiftedPulse(pulse, delayInteger), ...
    integerDelays.', 'UniformOutput', false);
pulseMatrix2 = cell2mat(pulseMatrix2);

% ------------------------------------------------------------------------
% Calculate the trace.
% ------------------------------------------------------------------------
data = (pulseMatrix1 + division * pulseMatrix2).^3;

% This is a workaround to avoid data truncation.
M = 2;
NFFT = M*NF;
while NFFT < size(data,2)
    M = M + 1;
    NFFT = M*NF;
end

trace1 = fft(data, NFFT, 2);
```

```
34
35 % Take only the relevant frequencies.
36 trace1 = trace1(:,M*(frequencyIndices + 1));
37
38 % Flip to correct orientation.
39 trace1 = trace1.';
40
41 % Intensity.
42 trace1 = abs(trace1).^2;
43
44 % Normalize.
45 output = trace1/max(max(abs(trace1)));
46 end
```