**University of Warsaw**
Faculty of Physics

**Filip Sośnicki**
Student's book no.: 366112

# Electrooptic temporal telescope for nanosecond pulses

**Master's thesis**
**field of study: PHYSICS**
**speciality: Theoretical physics**

The thesis written under the supervision of
**Dr Michał Karpiński**
Institute of Experimental Physics

Warsaw, September 2016

## Oświadczenie kierującego pracą

Potwierdzam, że niniejsza praca została przygotowana pod moim kierunkiem i stwierdzam, że spełnia ona warunki do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Data Podpis kierującego pracą

## Statement of the Supervisor on Submission of the Thesis

I hereby certify that the thesis submitted has been prepared under my supervision and I declare that it satisfies the requirements of submission in the proceedings for the award of a degree.

Date Signature of the Supervisor

## Oświadczenie autora pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Data Podpis autora pracy

## Statement of the Author on Submission of the Thesis

Aware of legal liability I certify that the thesis submitted has been prepared by myself and does not include information gathered contrary to the law.

I also declare that the thesis submitted has not been the subject of proceedings resulting in the award of a university degree.

Furthermore I certify that the submitted version of the thesis is identical with its attached electronic version.

Date Signature of the Author of the Thesis

## Abstract

Obiecującym podejściem do implementacji protokołów kwantowych jest wielofunkcyjny, fotoniczny system informacji kwantowej. Można go uzyskać za pomocą hybrydowych sieci kwantowych, które składają się z różnych elementów takich jak: kropki kwantowe, pamięci kwantowe czy bramki kwantowe, które są ze sobą optycznie połączone. Jednakże te podzespoły wykazują zasadniczo odmienne własności spektralne jak i czasowe. Z tego powodu ważnym wyzwaniem staje się stworzenie urządzenia, które będzie w stanie zmienić szerokość widmową kwantowych impulsów światła z rzędu GHz do rzędu MHz. W tej pracy dokonano analizy dwóch możliwości implementacji takiego urządzenia, które połączy odmienne części sieci kwantowej. Pierwsza z nich opiera się na idei konwersji z dziedziny czasu do dziedziny częstości, jednakże zysk wynikający z zastosowania realistycznej implementacji takiego urządzenia jest niwelowany poprzez straty spowodowane przez wymagane elementy dyspersyjne. W związku z tym wprowadzono ideę teleskopu czasowo-spektralnego, opartego na zasadzie dualizmu przestrzenno-czasowego. Wykonano analizę numeryczną wydajności teleskopu zaimplementowanego poprzez elektrooptyczną modulację fazy czasowej. Użycie architektury CUDA (Compute Unfied Device Architecure) pozowoliło na zaprogramowanie efektywnych symulacji takiego urządzenia. Stwierdzono, iż realizacja teleskopu elektrooptycznego stanie się możliwa w nieodległej przyszłości, wraz z dalszym rozwojem techniki szybkiej, elektrooptycznej modulacji fazy. Uzyskane wyniki pokazują, że jest to obiecujący sposób widmowej kompresji pojedynczych fotonów w przyszłych sieciach kwantowych.

## Key words

kwantowa informacja, elektrooptyczna modulacja fazy, kształtowanie krótkich impulsów światła

## Area of study (codes according to Erasmus Subject Area Codes List)

13.2 Physics

## The title of the thesis in Polish

Elektrooptyczny teleskop czasowy dla impulsów nanosekundowych

*I dedicate this work to my Parents for their love, support and for being a constant source of knowledge and inspiration.*

*I would like to thank my Supervisor, Dr Michał Karpiński, for the patient guidance, encouragement and advice.*

# Contents

# Table Of Abbreviations

| **Abbrevation** | **Meaning** |
|---|---|
| AWG | Arbitrary Waveform Generator |
| CFBG | Chirped Fiber Bragg Grating |
| CPU | Central Processing Unit |
| CUDA | Compute Unified Device Architecture |
| DAC | Digital to Analog Converter |
| DCF | Dispersion Compensating Fiber |
| (D)WDM | (Dense) Wavelength Division Multiplexing |
| EOM | Electrooptic Modulator |
| FFT | Fast Fourier Transform |
| FWHM | Full Width at Half Maximum |
| FWM | Four Wave Mixing |
| GDD | Group Delay Dispersion, also Group Velocity Dispersion propagator |
| GPU | Graphics Processing Unit |
| GV | Group Velocity propagator |
| IFFT | Inverse Fast Fourier Transform |
| SFG | Sum Frequency Generation |
| SPDC | Spontaneous Parametric Down Conversion |
| SS | Spectral Shift propagator |
| SVEA | Slowly Varying Envelope Approximation |
| TL | Time Lens |
| TPB | Time Bandwidth Product |
| VP | Phase Velocity propagator |

Functions in the time domain are denoted by small letters, e.g.:

$$e(t) = \frac{1}{\pi^{1/4}\sqrt{\sigma_t}}\,\mathrm{e}^{-\frac{t^2}{2\sigma_t^2}}$$

Functions in the frequency domain are denoted by capital letters, e.g.:

$$E(\omega) = \frac{1}{\pi^{1/4}\sqrt{\sigma_\omega}}\,\mathrm{e}^{-\frac{\omega^2}{2\sigma_\omega^2}}$$

# Chapter 1

# Introduction

A multifunctional photonic quantum information processing system – 'the quantum internet' [1] – is one of the greatest interest of quantum information research community. One approach of achieving it is to use a hybrid quantum network that combine different matter-based quantum information processing systems ('nodes') with photonic links, see Fig. 1.1. Different nodes can be responsible for different operations, some nodes can be responsible for emitting single photons pulses, others for interfering them. This interference can generate the entanglement, which can be used e.g. for secure communication or quantum computation. However, the efficiency of the entanglement generation is dependent on a visibility of the interference. Mismatch of the spectro-temporal modes, which contains interfering single photons causes a significant decrease of interference visibility. Therefore it dumps the efficiency of the entanglement generation as well, annihilating the performance of the quantum network. An important challenge is to create an "interface" device, which will coherently convert spectro-temporal modes of single photons.

Here I will briefly discuss several possible components of such a network. In point of

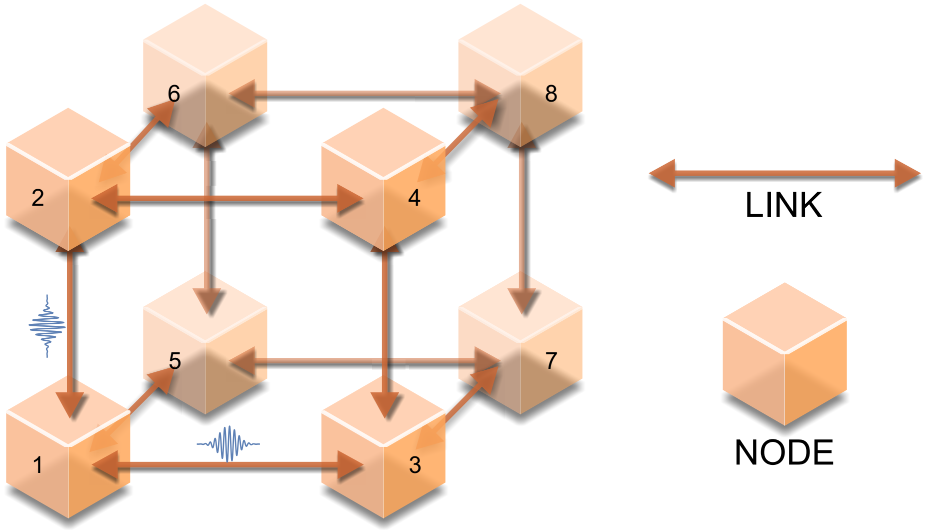

Figure 1.1: Schematic view of quantum network consisting of links and nodes. Entanglement can be spread through whole network by exchanging entangled photons, e.g. nodes 2 and 3 become entangled when they share entangled photons from node 1. A quantum network can be easily scalable just by adding more nodes and links.

view of spectrotemporal properties one important part of this system is the generation of single photons. Usually it is done by heralding single photons from photon pairs generated in spontaneus parametric down conversion (SPDC), where spectral width of photons equals few hundreds GHz, however it is a probabilistic source. In order to obtain a deterministic ("on demand") single photons sources one can use e.g. quantum dots [2–8], NV centers [9–12], alkali atoms [13] or single molecules [14], even organic ones [15]. Each of these sources has different spectrotemporal properties. A bandwidth of single photons can be from few MHz to hundreds (or even thousands) of GHz.

In order to obtain a fast long distance quantum communication it would be very desirable to use industry established standards such as the (dense) wavelength division multiplexing. It consists of dividing a single telecom window into multiple channels. Each of these channels has the spectral width from 12.5 GHz to 100 GHz. It allows to simultaneously send many single photons through an single optical fiber, resulting in increase of a communication bandwidth. However, these photons should be spectrally matched to the channel widths, because the spectraly narrower pulses have longer duration, which results in a reduction of communication bandwidth.

The other part of the photonic quantum information processing system is storing the quantum information, e.g. by quantum memories [16] or quantum repeaters [17]. It is done mainly by interaction between light and single atoms. Because of light-atom interaction, single photons are expected to have similar spectral properties as spectral lines of atoms, which width is in order of single MHz. Therefore there is at least 3 orders of magnitude of difference between spectral widths of this two parts of the quantum network. It can appear *inter alia* by highly decreased repetition rates of recording a quantum information from single photons to quantum memories or by even in inability to record or control the whole writing process. This problem can be reduced by putting between those two parts an interface device [18], which will modify the spectral bandwidth of a single photons by orders of magnitude in coherent way.

The already existing protocols [19–22] allow to compress single photons to required spectral widths, but they work only if spectral width of incoming photons is in order of hundreds GHz and the target bandwidth is larger than single GHz as well. However there is still lack of protocol, which allows to compress a single GHz width pulse to a single MHz. Also there is no protocol that will compress single photon bandwidths from 100 GHz range to few MHz range.

In this thesis I will explore a promising way to do such compression. The basic idea of spectral compression consists of chirping an input pulse in order to lenghten its pulse duration. It consists also of temporal phase modulation resulting in spectral compression. It can be realized be means of fast electrooptic phase modulation, which allows to have good control of this process. However, for spectrally narrow input pulses the needed amount of chirp is absurdly large introducing extremely large losses, causing the idea to become completely impractical. But amount of needed chirp can be significantly decresed by spectraly stretching the input pulse at the begining. This idea corresponds to the working principle of a time telescope, which is a temporal counterpart of a spatial telescope, which can be obtained by using the space-time duality principle [23]. A time telescope consists of two lenses and one dispersive element between them. The first lens spectrally broadens the input pulse, which can gain more temporal broadening per given amount of dispersion. This enables sufficient temporal stretching of the pulse with manageable levels of loss. In this element a pulse obtains an output temporal width. Then the second lens with a large aperture spectrally compresses the pulse to the desired target bandwidth, which is a few orders of magnitude narrower than the input pulse. In this thesis I will derive equations describing a bandwidth converter, which is the same as the temporal telescope, but without the first lens, and I will attempt to derive equations describing the time telescope. Then I will show results of numerical simulations focusing on efficiency of such devices, using both realistic and future parameters, such as speed of electronics and amount of available amount of dispersion with realistic levels of loss.

## 1.1. Photonic networks

In the first part of quantum network one can include single photons sources [24] such as quantum dots [2–8], NV centers [9–12] and alkali atoms [13]. It also includes a long distance communication. This one contains the use of currently existing telecom infrastructure including the (dense) wavelength division multiplexing technique – (D)WDM, which allows to increase rates of information sent through a single optical fiber. Here I want to briefly describe spectrotemporal properties of single photons produced by mentioned quantum sources and also I will describe the (D)WDM technique.

### 1.1.1. Quantum dots

Quantum dots are usually atom-like systems based on semiconductor properties. They consist of a semiconducting structure spatially limited in all three directions (hence the name 'dot') grown on another semiconductor with different energy level system. Therefore this kind of structure can trap a single electron. Because their small size results in quantization of motion of electrons in all three direction, they are characterized by a discrete energy-level spectrum, therefore they are often referred to as 'artificial atoms'. An exciton (electron-hole pair) can be created on demand in optical [3,4] or electrical [5,6] way and its radiative recombination results in emission of single photons. Because in single quantum dot there can be only one exciton, the probability of creating two photons at once is extremely low. It results also in relatively low degree of second-order coherence $g^{(2)}$ [2, 3]. Due to noise present in semiconductor sturctures the achieved $g^{(2)}$ values are not as low as for NV centers and single atoms [4, 6].

Quantum dots can be engineered in order to achieve a required properties such as wavelength of emitted photons or direction of emission, e.g. by using a specific semiconducting material with specific energy levels or by growing a distributed-Bragg-reflectior on them, which directs emitted photons. Quantum dots can be also integrated into microcavities. When the energy and polarization of the emission field is coupled to the cavity one can obtain significantly higher rate of spontaneous emission from quantum dots, due to the Purcell effect. Quantum dots as single photons emitters are operating at cryogenic temperatures. A possibility to get single photons "on demand" with desired properties results in a very promising device to be used as single photon emitter.

Single photons that are emitted from quantum dots are typically a few nanosecond or less monoexponentialy decaying pulses. Spectral width of single photons produced by typical quantum dots is of the order of single GHz.

### 1.1.2. NV centers

A single nitrogen-vacancy centers in diamond are one of numerous well studied luminescent point defects in diamond [9–12]. This kind of defect is consisting of one substitutional nitrogen – vacancy pair. It can be formed e.g. by radiation damage and annealing or as a product of the chemical vapor deposition diamond synthesis process.

The NV centers posses an atom-like energy level structure. In order to excite NV centers one has to use a photoluminescence excitation. Energy levels can by slightly tuned by external electric field through the dc Stark effect. The NV centers optical emission spectrum consists of two parts, a zero phonon line (ZPL) that contributes few percent of to the total emission and a phonon side band (PSB). To obtain a single photon emitter one has to collect only these photons coming from ZPL, which is done by narrowband filters and by using specific oriented NV centers and polarizers. There have been reported NV centers with $g^{(2)}$ less than 0.1. The spectral width of single photons produced in NV centers is of order of several hundreds of MHz. Other defects, such as SiV are also explored as promising single photon sources.

### 1.1.3. Single alkali atoms

Single atoms strongly coupled to a high-finesse optical cavity can be used as a quantum emitter, however this kind of quantum emitters need significant experimental efforts to achieve [13]. Alkali atoms such as Cs and Rb are at first captured into magneto-optical trap (MOT) and cooled down. Then the MOT is turned off causing a free fall of atoms under the pull of gravity. When atoms pass through an optical cavity, an optical trap is turned on. In order to obtain a single photon emitter it is important to trap only a single atom. One usually uses a $\Lambda$ energy levels system with three states, two ground $|g\rangle$ and $|u\rangle$ and one excited $|e\rangle$. The resonance of the cavity is tuned close to transition of $|g\rangle \rightarrow |e\rangle$ and the pump laser pulses are on-resonance with transition $|u\rangle \rightarrow |e\rangle$. With appropriate control of pump pulses and coupling of atom with cavity one can obtain a transition $|u\rangle$ to $|g\rangle$ through $|e\rangle$ with emitting a single photon. The spectrum of emitted photons matches an atomic line bandwidth of those states, therefore it is about several MHz. The duration of pulse containing single photon is about hundred of nanoseconds.

### 1.1.4. (Dense) Wavelength Division Multiplexing – (D)WDM

The (Dense) Wavelength Division Multiplexing is a technique allowing to achieve a significant increase of data rates in every single optical fiber. The working principle of (D)WDM is to divide a single optical window into smaller channels with width about 12.5 GHz (or 25 GHz, 50 GHz, etc. depending on implementation), therefore with central wavelength spacing with 12.5 GHz. This value is optimal at this time, because on one hand wider channels means less parallelism of data flow, which is the main idea of (D)WDM and also spectrally wider photons causes a faster broadening of its time envelope due to group velocity dispersion. Therefore it causes a decrease of number of transmitted photons per second. On the other hand narrower channels are problematic due to non existing device which will be able to demultiplex channels at the output of an optical fiber.

## 1.2. Quantum memories and quantum repeaters

The main motivation for obtaining quatum memories (QM) is synchronization of different nodes of the quantum network. Quantum memory used in this manner can be named as quantum repeater. It is needed due to probabilistic behavior of many elements of the quantum network. QMs are based on light-atom interaction [16]. It is usually realizaed in vapors of atoms in ground state $|g\rangle$. Then, for the simplest protocol, the incoming photon one wants to store in memory is exciting an atom to excited state $|e\rangle$ and at the same time a strong control laser field is transiting this excitation into lower 'dark' state $|s\rangle$. As it can be seen, the whole process is based on using absorption lines of atoms and those are about single MHz width. If spectral width of writing photons is larger it means that a lot of those do not excite an atom in a desirable way, therefore these photons will not be stored in quantum memory. Quantum memories with larger bandwidth have been demonstrated [16], however their storage time is much smaller than that of the spectrally narrower ones.

### 1.2.1. Single photon nonlinearity

The main idea of the quantum network is to produce single photons, create an entanglement between them and than use it for e.g. quantum computation or secure communication. The entanglement in such system arises from photon-photon interactions, however these in vacuum are extremely weak. The strong interaction between light pulses can be obtained by nonlinear processes, such as sum frequency generation, four wave mixing, etc. Due to weakness of the light-light interaction, the effects abundance achieved by nonlinear processes can be typically reached only by the strong light pulses containing millions of photons. However latest achievements in the use of the electromagnetically induced transparency (EIT) or the Rydberg blockade in the Rydberg atoms shows a possibility to achieve nonlinear processes in the single photon regime [25, 26]. Especially combining these two techniques results in the strong interaction between single photons. It was demonstrated e.g. by Tiarks *et al.* [26] by constructing a c-phase quantum gate (phase gate with control qubit), which can achieve a $\pi$ phase shift. This kind of two-photon quantum gates enable construction of a quantum computer. They also enable creation of highly non-trivial photonic quantum states that may be useful for e.g. quantum metrology or quantum simulations. Both approaches, EIT and Rydberg atoms, are compatible with photons whose bandwidth is on the order of 1-10 MHz.

## 1.3. Summary

As can be seen from the examples listed above, the possible components of a hybrid quantum network vastly differ in their characteristic bandwidths: from tens of GHz for (D)WDM channels, through single GHz for quantum dot single photons sources to single MHZ for media exhibiting single-photon nonlinearity and for most quantum memories.

## 1.4. Optical space-time duality

Optical space-time duality is a useful tool, which allows us to apply well-known concepts of free-space spatial optics to the time domain [23, 27, 28]. It is based on mathematical equivalence of wave equations describing paraxial diffraction and narrowband pulse dispersion. Space-time duality provides us a new kind of temporal optical elements, such as time lenses or time prisms. Using it one can easily translate a spatial optical system into its temporal counterpart, e.g. a telescope, which compresses a spatial waveform transverse to propagation axis, into a time-telescope, which changes a spectrotemporal properties of light pulses.

In order to show this duality, as in almost every optics tasks, one has to start with Maxwell's equations. I will use the form of Maxwell's equations, where no charges and no currents are present, however I will not introduce now any assumptions on material properties:

$$\nabla \cdot \vec{e} = 0, \tag{1.1}$$

$$\nabla \cdot \vec{b} = 0, \tag{1.2}$$

$$\nabla \times \vec{e} = -\frac{\partial \vec{b}}{\partial t}, \tag{1.3}$$

$$\nabla \times \vec{b} = \mu\mu_0\varepsilon\varepsilon_0\frac{\partial \vec{e}}{\partial t}, \tag{1.4}$$

where $\vec{e}$ and $\vec{b}$ are electric and magnetic fields respectively and $\varepsilon\varepsilon_0$ is permittivity and $\mu\mu_0$ is permeability of the material. Assuming propagation in vacuum, where $\varepsilon = 1$ and $\mu = 1$, one can use rotational equations (1.3) and (1.4) to derive the wave equation:

$$\left(\nabla^2 - \frac{1}{c^2}\frac{\partial^2}{\partial t^2}\right)\vec{e}(x,y,z,t) = 0, \tag{1.5}$$

where $c = \frac{1}{\sqrt{\varepsilon_0\mu_0}}$ is the speed of light in vacuum.

The most general solution to the wave equation will describe propagation of light both in space and time of any arbitrarily shaped wave. However, such a solution is hard to obtain, so one can focus on spatial or temporal propagation by making appropriate assumptions.

### 1.4.1. Paraxial diffraction in vacuum

In order to study the spatial problem firstly one assumes monochromacity of the propagating wave, so that it has only one spectral component $\omega_0$. Also one can take, without loss of generality, the field to be a scalar function of position. In such case it is easier to make calculations in the frequency domain, rather than in the time domain, where the propagating wave reads:

$$E(x,y,z,\omega) = \mathcal{E}(x,y,z)\,\delta\left(\omega - \omega_0\right). \tag{1.6}$$

After introducing this assumption, the wave equation (1.5) reduces to the Helmholtz equation:

$$\left(\nabla^2 + \frac{\omega_0^2}{c^2}\right)\mathcal{E}(x,y,z) = 0, \tag{1.7}$$

where $\frac{\omega_0}{c} = k = \frac{2\pi}{\lambda}$ can be identified as the wavenumber.

In the paraxial approximation of light propagation our interest is to study propagation of rays confined mostly along a given axis, for simplicity let us assume it is the $z$-axis. Then the most rapid changes in phase will occur along this $z$-axis, which allows us to write:

$$\mathcal{E}(x,y,z) = \tilde{\mathcal{E}}(x,y,z)\,\mathrm{e}^{-\mathrm{i}kz}, \tag{1.8}$$

where $\tilde{\mathcal{E}}(x,y,z)$ varies slowly with change in position, in comparison to the wavelength $\frac{2\pi}{k}$. This assumption is called the slowly varying envelope approximation (SVEA). Inserting this result to the reduced wave equation (1.7) will give us:

$$\frac{\partial^2\tilde{\mathcal{E}}}{\partial x^2} + \frac{\partial^2\tilde{\mathcal{E}}}{\partial y^2} + \frac{\partial^2\tilde{\mathcal{E}}}{\partial z^2} - 2\mathrm{i}k\frac{\partial\tilde{\mathcal{E}}}{\partial z} = 0. \tag{1.9}$$

The next step is to invoke the paraxial approximation:

$$\left|\frac{\partial^2\tilde{\mathcal{E}}}{\partial z^2}\right| \ll \begin{cases} \left|\dfrac{\partial^2\tilde{\mathcal{E}}}{\partial x^2}\right| \\[2ex] \left|\dfrac{\partial^2\tilde{\mathcal{E}}}{\partial y^2}\right| \\[2ex] 2\mathrm{i}k\left|\dfrac{\partial\tilde{\mathcal{E}}}{\partial z}\right| \end{cases}. \tag{1.10}$$

In other words the paraxial approximation tells us that the curvature of field's envelope function $\tilde{\mathcal{E}}$ is much smaller along the propagation axis than along the transverse profile. It results in simplified, paraxial wave equation:

$$\frac{\partial \tilde{\mathcal{E}}}{\partial z} = -\frac{\mathrm{i}}{2k}\left(\frac{\partial^2 \tilde{\mathcal{E}}}{\partial x^2} + \frac{\partial^2 \tilde{\mathcal{E}}}{\partial y^2}\right), \tag{1.11}$$

which can be also presented as:

$$\frac{\partial}{\partial z}\tilde{\mathcal{E}}(x,y) = -\frac{\mathrm{i}}{2k}\nabla_T^2 \tilde{\mathcal{E}}(x,y), \tag{1.12}$$

where $\nabla_T^2$ is the Laplacian operator in the plane transverse to the $z$-axis.

### 1.4.2. Narrowband dispersion

In this section I will derive a differential equation governing propagation of a narrowband pulse in a dispersive medium along $z$-axis with a propagation constant $\beta(\omega)$. Again, without loss of generality, the field is taken to be a scalar function of position and time. Next by using the SVEA one can write a function describing a wave packet traveling in the $z$ direction:

$$e(x,y,z,t) = a(z,t)\, \mathrm{e}^{\mathrm{i}(\omega_0 t - \beta(\omega_0) z)}. \tag{1.13}$$

In contrast to the paraxial diffraction, propagation takes place in a dispersive medium, where $\varepsilon$ is a function of frequency $\varepsilon(\omega)$ and, to a good approximation, $\mu$ can be taken constant. Now the propagation constant $\beta$ (also called $k$ in previous section) becomes a function of frequency, therefore the next step is to expand it in frequency to second order around $\omega_0$:

$$\begin{aligned}\beta(\omega) = \beta(\omega_0) + \left.\frac{\mathrm{d}\beta}{\mathrm{d}\omega}\right|_{\omega=\omega_0}(\omega-\omega_0) + \left.\frac{1}{2}\frac{\mathrm{d}^2\beta}{\mathrm{d}\omega^2}\right|_{\omega=\omega_0}(\omega-\omega_0)^2 + \mathcal{O}\left(\omega^3\right) = \\ = \beta_0 + \beta_1(\omega-\omega_0) + \frac{\beta_2}{2}(\omega-\omega_0)^2 + \mathcal{O}\left(\omega^3\right).\end{aligned} \tag{1.14}$$

Here, for simplicity, we are interested only in the second order component, while the rest will be closely investigated in another section (see section 2.2). The $\beta_2$ is corresponding to a group velocity dispersion, which causes plane waves of different frequency to travel with different velocities. Now one can make a spectral decomposition of envelope function $a(z,t)$:

$$a(z,t) = \int \mathrm{d}\omega\; A(\omega-\omega_0)\; \mathrm{e}^{\mathrm{i}(\omega_0 t - \beta(\omega) z)}. \tag{1.15}$$

We are looking for information on how the envelope function changes along $z$-axis, which can be obtained by first derivative in respect to the distance $z$:

$$\begin{aligned}\frac{\partial}{\partial z}A(z,t) = \int \mathrm{d}\omega \left(-\mathrm{i}\frac{\beta_2}{2}(\omega-\omega_0)^2\right) A(\omega-\omega_0)\,\mathrm{e}^{\mathrm{i}(\omega_0 t - \beta(\omega) z)} = \\ = \frac{\mathrm{i}\beta_2}{2}\frac{\partial^2}{\partial t^2}\int \mathrm{d}\omega\; A(\omega-\omega_0)\,\mathrm{e}^{\mathrm{i}(\omega_0 t - \beta(\omega) z)} = \frac{\mathrm{i}\beta_2}{2}\frac{\partial^2}{\partial t^2}\,A(z,t).\end{aligned} \tag{1.16}$$

In this manner we have arrived at a differential equation describing the propagation of a narrowband pulse in a dispersive medium:

$$\frac{\partial}{\partial z}A(z,t) = \frac{\mathrm{i}\beta_2}{2}\frac{\partial^2}{\partial t^2}\,A(z,t), \tag{1.17}$$

which can be rewritten in the same form as (1.12):

$$\frac{\partial}{\partial z}A(z,t) = \frac{\mathrm{i}\beta_2}{2}\nabla_t^2 A(z,t), \tag{1.18}$$

where $\nabla_t^2$ is the Laplacian operator in "transverse" direction to $z$-axis – time.

### 1.4.3. Temporal counterparts of spatial optical elements

The exact equivalence of equations describing paraxial diffraction (1.12) and narrowband pulse dispersion (1.18) allows us to find temporal counterparts of spatial optical elements, such as free (but paraxial) propagation, lenses, prisms or even whole systems based on the mentioned optical components.

#### Diffraction and dispersion

The paraxial diffraction, known also as Fresnel diffraction, results in adding a quadratic phase to the signal in the domain of spatial frequencies. On the other hand the narrowband dispersion causes an additional quadratic phase in frequency domain. Thus, if we notice, that:

- transverse dimension $x$ or $y$ in paraxial diffraction is corresponding to retarded time $t$ (measured from center of light pulse) in a narrowband pulse,
- wavenumber $k$ in paraxial diffraction is corresponding to minus inverse of second order of propagation constant $-\beta_2^{-1}$,
- diffraction distance $L$ corresponds to amount of introduced dispersion $\Phi$ on the same length $L$, where:

  $$\Phi = -\beta_2 L,$$

  where the minus sign arises from fact, that normal dispersion is negative: $\beta_2 < 0$,
- spatial envelope function corresponds to temporal envelope function,

one can conclude, that Fresnel diffraction and narrowband pulse propagation are mathematically the same, but in different domains.

Dispersion can be introduced with different techniques, e.g. using optical fibers, dispresive gratings or chirped fiber Bragg gratings (CFBG) (see section 1.6.3).

#### Lens

The working principle of an ideal, thin spatial lens is to add a quadratic phase to a spatial signal in a plane transverse to the propagation axis. In order to describe it I will use a propagator notation which allows to describe transformations in such a way that one can easy calculate the output signal for any input signal just by integrating it over input system. I introduce the propagator language in section 2.2.1.

For a spatial lens the propagator reads:

$$L(x', x; f) = \mathrm{e}^{\mathrm{i}\frac{x^2}{2f}}\,\delta(x'-x). \tag{1.19}$$

Knowing that transverse dimensions in spatial optics correspond to time in temporal optics, one can conclude that if one adds a quadratic phase to a wavepacket in time, then it can be named as a time lens, and its time propagator is:

$$TL\left(t', t, f_T\right) = \mathrm{e}^{\mathrm{i}\frac{\omega_0 t^2}{2f_T}}\,\delta(t'-t) \qquad \text{or} \qquad TL\left(t', t, K\right) = \mathrm{e}^{\mathrm{i}\frac{Kt^2}{2}}\,\delta(t'-t), \tag{1.20}$$

depending on the used formalism. In this work I will be using mainly the second formalism, where the parameter describing the modulation depth (focusing) of lens is $K$. The spatial lens modifies the spatial spectrum, therefore the time lens does the same in the frequency spectrum.

Its implementation requires introducing a quadratic time-varying phase. It can be done in different ways, the main being: electrooptic phase modulators (EOM) using Pockels effect [19] or optical Kerr effect; using nonlinear effects such as sum frequency generation, where input pulse meets a chirped classical pulse [20] or four-wave-mixing [29] or even cross-phase modulation of the original pulse with an intense pump pulse in a nonlinear fiber [21].

Here we focus on an electrooptic phase modulation due to directness of this method. It does not need any additional laser beams, which can introduce noise to the modulated single photons, therefore in principle it should preserve their quantum state. Because there is no other laser it does not need to satisfy the stringent phase-matching conditions required by nonlinear optical methods, therefore it is very versatile. Additionally, its operation is always deterministic (contrary to nonlinear methods).

### Imaging system

One of the simplest imaging system consists of a thin lens with focal length $f$ and two diffraction lengths $L_1$ and $L_2$ (Fig.1.2), which satisfy the lens formula:

$$\frac{1}{f} = \frac{1}{L_1} + \frac{1}{L_2}. \tag{1.21}$$

Such arrangement results in creation of a magnified image with magnification proportional to the ratio of the distances $p = {}^{L_2}/_{L_1}$.

Using the space-time duality – temporal counterparts of spatial optical elements mentioned above – one can translate such a spatial imaging system into a temporal imaging system, that instead of transforming a spatial image, transforms the temporal profile of a wave-packet with the same properties as its spatial protoplast – magnification and inversion. Now in place of diffraction one has to put a dispersive element, e.g. an optical fiber, and in place of a spatial lens one has to put a time lens, e.g. by using EOM. Then the focal length $f$ translates into the temporal focal length $f_T$ or time lens parameter $K = \frac{\omega_0}{f_T}$ and diffraction distances translate into amounts of dispersion $\Phi_1$ and $\Phi_2$, such that the time lens formula is satisfied:

$$\frac{\omega_0}{f_T} = \frac{1}{\Phi_1} + \frac{1}{\Phi_2} \qquad \text{or} \qquad K = \frac{1}{\Phi_1} + \frac{1}{\Phi_2}. \tag{1.22}$$

## 1.5. Idea of a bandwidth converter and a temporal telescope

The main purpose of this work is to examine two electrooptical systems that are intended to spectrally compress light pulses, in particular single photons. The first one is based on a single time lens and one dispersive element, I will call it here a bandwidth converter. It is a temporal counterpart of an optical Fourier transform in the spatial domain, that consists of a diffraction along the length $L$ and a lens with focal length matching the diffraction length:

$$f = L. \tag{1.23}$$

For Gaussian beam it results in a translation the initial beam waist into scaled output curvature of wavefront, or if one reverse a beam, then it results in translation the initial curvature

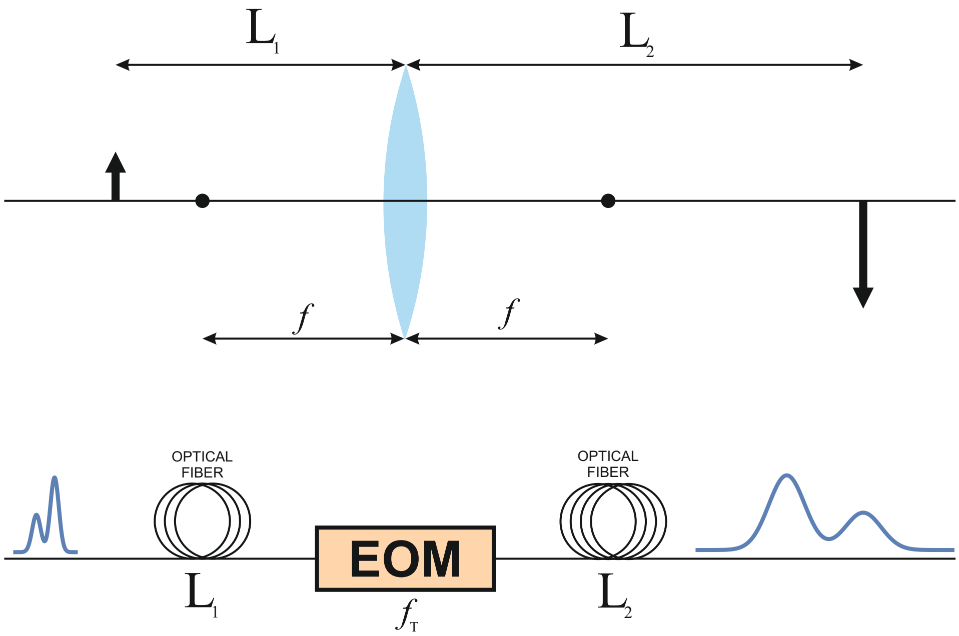

Figure 1.2: Schematic view of spatial and temporal imaging system.

of wavefront into scaled output beam waist, where the scaling factor is given by $f$. Therefore, when one uses space-time duality to move this system into temporal-spectrum domain, one can use it to change properties of a Gaussian pulse. Now the curvature of beam wavefront translates into spectral width of Gaussian pulse spectrum and beam waist translates into a duration of Gaussian pulse. Therefore, at the output of a system consisting of a dispersion element and a time lens, when the input is a Gaussian pulse, one gets again a Gaussian pulse, since the Fourier transform of Gaussian function is Gaussian function, however with different temporal and spectral widths, scaled by a factor given by the lens parameter $K$ or by the dispersion parameter $\Phi$, which satisfy the condition:

$$\Phi = \frac{1}{K}. \tag{1.24}$$

In other words, using the system shown in Fig. 1.3, first a dispersive element broadens a pulse in time and then a time lens compresses its spectrum. It allows us to efficiently change the spectral width of single photons from hundreds GHz to hundreds or tens of MHz. However, the use of this device is limited due to imperfection of implementation (see section 1.6), especially those resulting from finite transmittance of dispersive elements. This is because, the narrower spectrum of the input pulse is, the more dispersion one needs to obtain compression to the same output width. In particular for compressing 1 THz width pulse into 10 MHz one needs 2500 ps$^2$, which is practically achievable, however for the compression from 1 GHz width input pulse into 10 MHz one needs $2.5 \times 10^6$ ps$^2$, which is to our best knowledge impossible to realize using currently developed technologies.

Therefore instead of using just one time lens and one dispersive element, one can use two time lenses and a dispresive element. This kind of system is called a telescope and its working principle is well developed in spatial domain, especially for Gaussian beams. Using again space-time duality one can translate it into temporal domain creating a temporal telescope, which in principle should allow us to change both temporal and spectral width of single photons in a coherent way for several orders of magnitude, even for spectrally narrow input

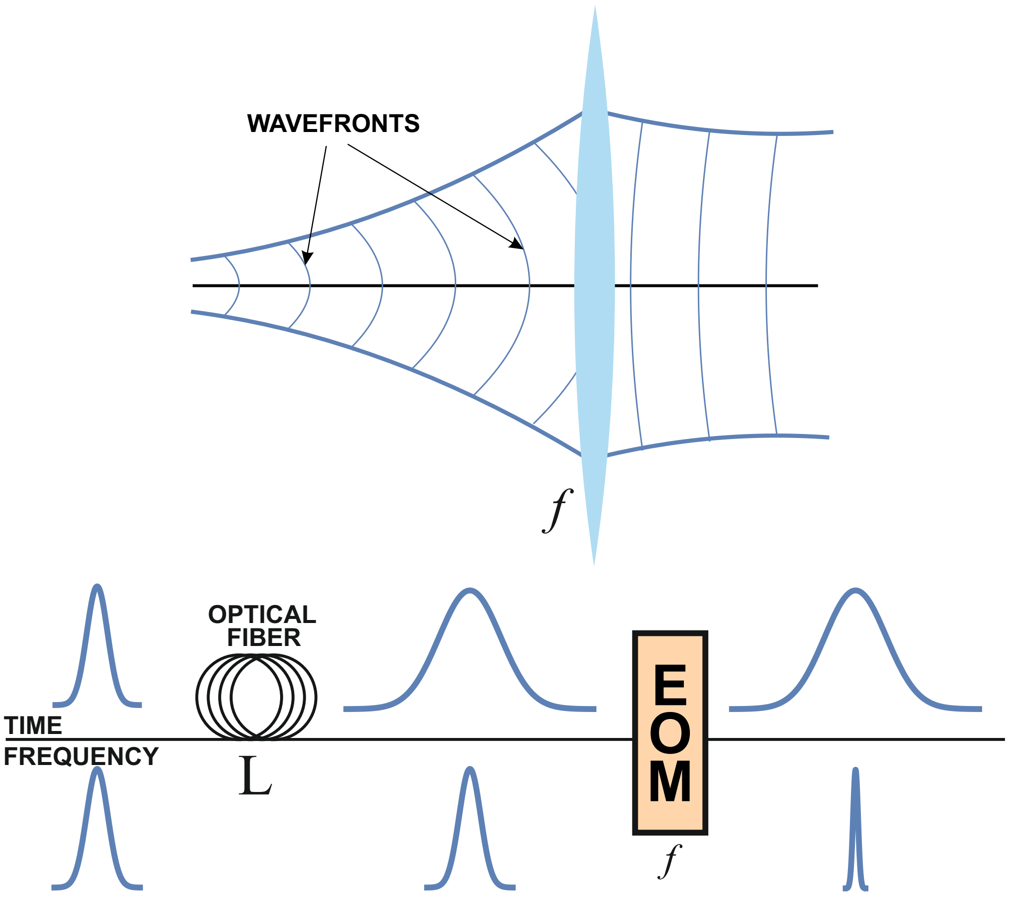

Figure 1.3: An idea of the bandwidth converter for transform limited pulses is to translate Gaussian beam waist into pulse duration and curvature of wavefronts into spectral width

pulses, e.g. from 5 GHz to 5 MHz. In the spatial domain the telescope consists of two lenses separated by a single diffraction length, one converging and one diverging or two converging. Each of these elements can be easy translated into temporal domain using space-time duality. Efficient implementation of a telescope requires using the lowest possible amount of dispersion (spatial diffraction length), therefore we choose a setup of one diverging and one converging, because hypothetically it ensures the best performance.
The spatial telescope equation reads:

$$L = f_1 + f_2, \tag{1.25}$$

where $L$ is the length of propagation (amount of diffraction), $f_1 < 0$ is the focal length of the first, diverging lens, and $f_2 > 0$ is the focal length of the second, converging lens. This setup provides a magnification of spatial pattern:

$$P = \frac{f_2}{f_1}. \tag{1.26}$$

After translation into the time domain, the temporal telescope equation is as follows:

$$\Phi = \frac{1}{K_1} + \frac{1}{K_2}, \tag{1.27}$$

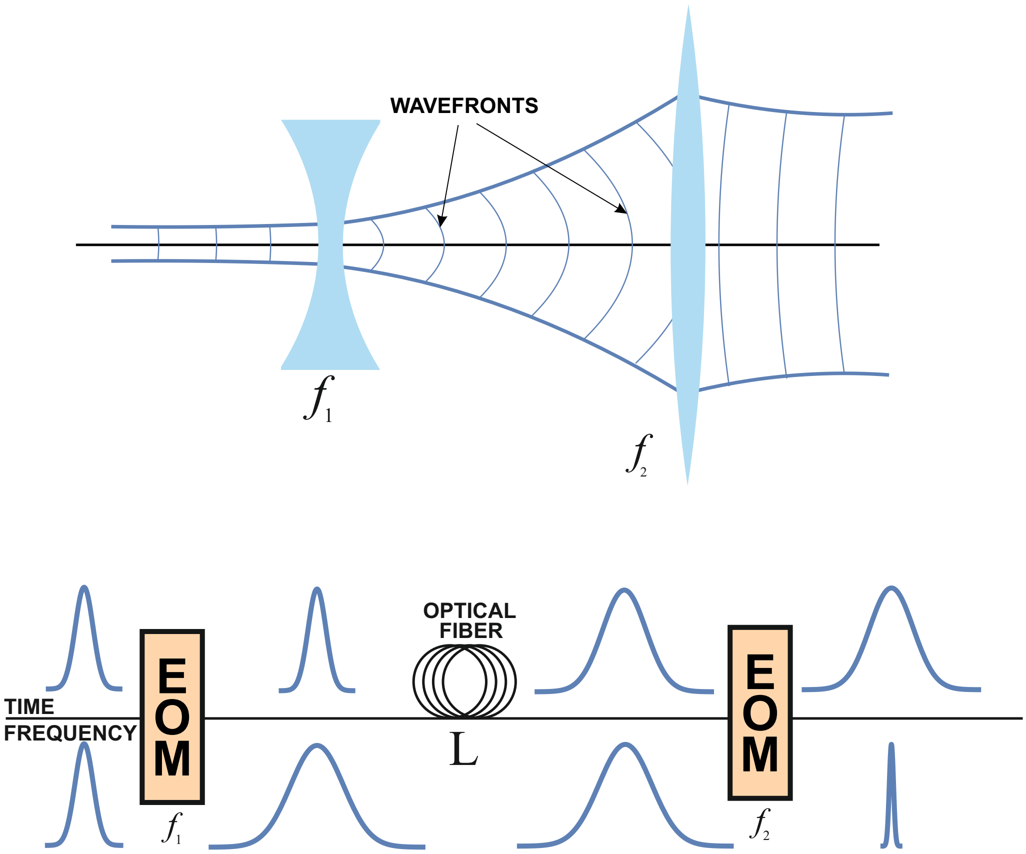

Figure 1.4: The idea of a temporal telescope for transform limited the Gaussian pulses is to translate Gaussian beam waist into pulse duration and the curvature of wavefronts into spectral width.

where $\Phi$ is total introduced dispersion, $K_1 < 0$ describes the first, diverging lens and $K_2 > 0$ describes the second, converging lens. It is now clear, that the setup of diverging-converging lenses is optimal for minimalization of the required amount of dispersion. The temporal telescope should provide a magnification of a time signal:

$$P_t = \frac{K_1}{K_2} \tag{1.28}$$

and a magnification of a spectral signal:

$$P_\omega = \frac{K_2}{K_1} = P_t^{-1}. \tag{1.29}$$

In case of Gaussian beam in the spatial regime one can translate it into time propagation of transform limited Gaussian pulses. Then the beam waist in spatial regime becomes the pulse duration of a Fourier-limited a Gaussian pulse and the curvature of beam wavefronts translates into the spectral width of Gaussian pulse in time, as shown in Fig.1.4

## 1.6. Implementation and technical problems

Typical implementation of the bandwidth converter and the telescope is to use an electrooptic phase modulator driven with an RF signal from an arbitrary waveform generator (AWG) or alternatively a sinusoidal waveform generator and a dispersive element, e.g. optical fiber with high dispersion or chirped fiber Bragg gratings (CFBGs). In this section I will describe the working principle of each element used in implementation and list problems related to the use of the specific element.

### 1.6.1. Electro-optic phase modulator

An electro-optic phase modulator (EOM) is a device, which uses the Pockels effect, where an electric field, which is applied to an electro-optic crystal with external electronics (e.g. AWG), causes a change of refractive index of the crystal, which modifies the phase acquired by a beam sent through a crystal. The modulation depth is proportional to the applied RF signal, however most of EOMs are designed to get modulation depth of $\pi$. It is possible to get deeper modulation by increasing the driving voltage, but there exist a modulation depth limit, due to maximal voltage for which one can simply burn an EOM, therefore maximal modulation depth is about several $\pi$. There exist free-space electro-optic phase modulators, for which the half-wave voltage – the voltage required for inducing a phase change of $\pi$ – is in order of single $kV$ and waveguide electro-optic phase modulators, for which the half-wave voltage is in order of few $V$.

### 1.6.2. Arbitrary waveform generation

An arbitrary waveform generator is a device, that generates an electronic output signal, given in digital form, e.g. from a computer. It consists mainly of digital to analog converter (DAC). It allows to generate an arbitrary RF signal in time, however it has several limitations.

The first limitation is the frequency response of the analog part of AWG. It is well known that in electronic devices resistance increases with higher frequencies due to e.g. electrical reactance (or more generally impedance). The best produced AWG at this time, Keysight M8196A, provides a frequency response shown in Fig. 1.5. After mathematical compensation done by increasing amplitudes of high frequencies in digital signal (before it is put to DAC), one can obtain an analog bandwidth of 32 GHz (where frequency response is crossing 3 dB line).

The second limitation is due to discretization of the input signal in the digital part of the AWG. It consists of discretization in time and discretization of amplitude. The first is resulting in limitation in amounts of samples per second and the best AWG at this time, the mentioned above Keysight M8196A, is able to achieve 92 Gsamples/s, which means that time resolution is around 10 ps between change of amplitude in time. The second type of discretization results from the fact, that each amplitude is recorded as a 8-bit word, which means that one is able to code only $2^8 = 256$ different values of amplitude, which can be a problem when one needs a very detailed waveform on different scales. However the effect of disctretization is quite strongly damped by the frequency response, because the disctertization causes generation of steps in output waveform, which consists mainly of high frequencies. Therefore frequency response results, in some sense, in averaging of discretization steps creating continuous signal, which is consistent with the Nyquist limit.

The third problem is that additional electronic noise is added on the output signal from AWG, therefore each repetition of the signal is not exactly the same. This effect however is not included in this thesis and will be closely investigated in future work.

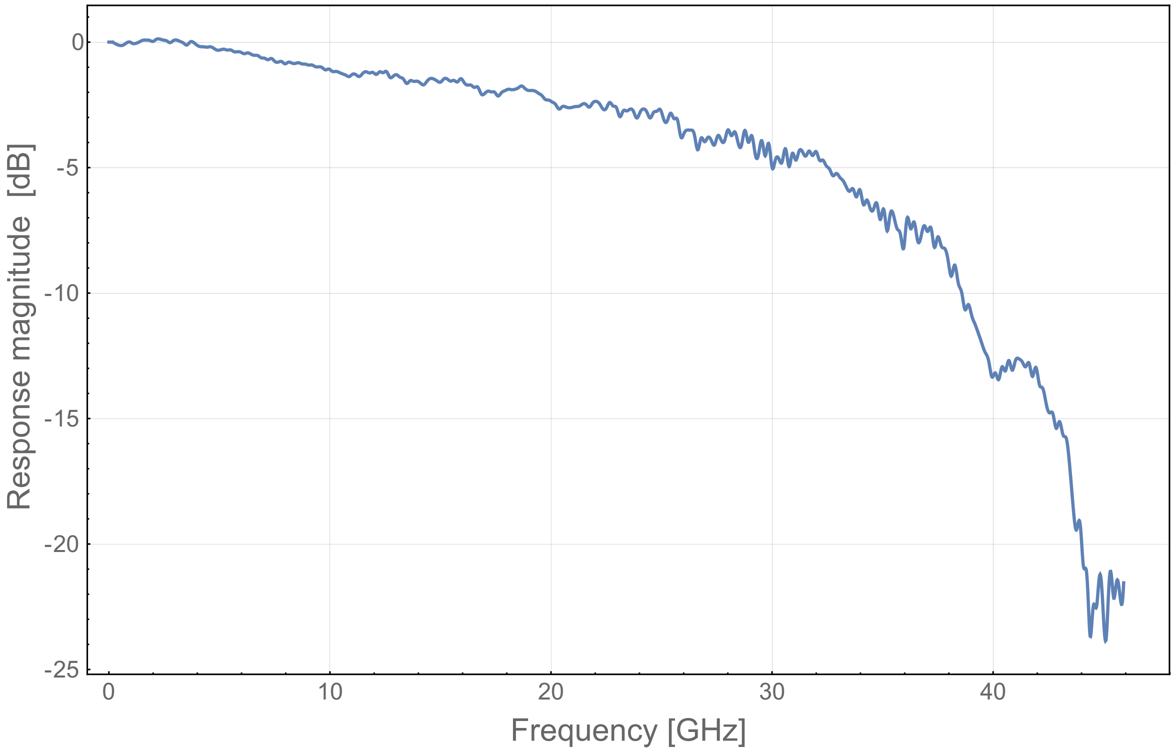

Figure 1.5: Frequency response for an arbitrary waveform generated by Keysight M8196A – it does not include mathematical compensation.

### Sine signal generation

One of the implementations of the time lens is approximating quadratic temporal phase with the temporal phase profile around the maximum/minimum of the sine signal. To this task one can employ either an AWG or a sine signal generator. Because it has only one frequency, one does not has to worry about frequency response and its compensation. However using sine signals in implementing time lenses has its drawback, such as a limited temporal $f$-number, which is the ratio between the focal length and aperture of a lens. It will be investigated more closely later (see section 4.2).

## 1.6.3. Dispersion

Dispersion can be introduced by different dispersive elements and its effects will be investigated in another section (see section 2.2). However in the most simple words group velocity dispersion causes that different spectral components arrive at different time, where arrival time is linear with spectral distance from some central frequency. In other words a quadratic spectral phase is applied to the spectrum of a signal. Therefore dispersion can be obtained from material properties (e.g. in optical fibers) or specially designed devices as chirped fiber Bragg grating (CFBG), which introduce delays for different spectral components.

### Optical fibers and dispersion compensating optical fibers

An arbitrary optical fiber introduces dispersion to a signal propagating in it due to material (dielectric) properties and propagating conditions. They can be optimized in order to get zero-dispersion fibers (for certain wavelength, e.g. telecomunication 1550 nm) or high-dispersion fibers, used mainly to compensate dispersion coming from dispersion in another optical fiber, hence the name – dispersion compensating fibers (DCF). The total amount of dispersion (chirp) can be easily chosen, because dispersion in optical fibers is given per length unit. Therefore in order to get certain amount of chirp one simply takes a certain length of fiber. However, propagation in fibers causes not only chromatic dispersion, but also losses, which are given per length unit as well. So when big amount of dispersion is needed, losses are also high. In particular the DCF from Thorlabs reaches 48.5 $\frac{\mathrm{ps}^2}{\mathrm{km}}$ with maximal losses of 0.265 $\frac{\mathrm{dB}}{\mathrm{km}}$. It means that length of such optical fiber providing dispersion of 11500 $\mathrm{ps}^2$ is around 237 km causing losses of around 62 dB, therefore only 0.08% of photons are transmitted through it.

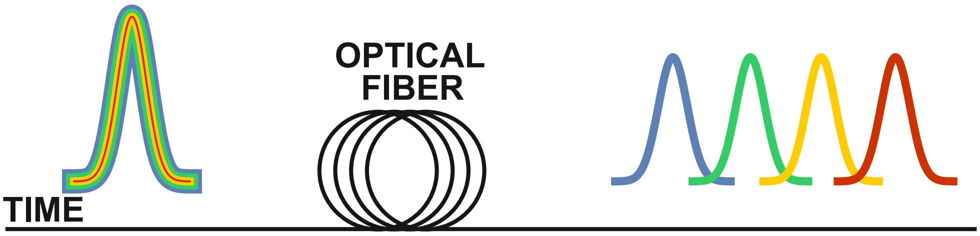

Figure 1.6: Chromatic disperssion caused by an optical fiber.

### Chirped fiber Bragg gratings

Chirped fiber Bragg grating is the idea of multiple sequential Bragg mirrors moved into an optical fiber system. It is obtained by periodically changing refractive index while also modifying the modulation period. Its principle of work is to reflect high frequencies at an earlier stage and lower frequencies at a later stage (see Fig. 1.7). In this way high amounts of dispersion can be introduced with relatively low losses. The best commercially available at this time CFBG@1550nm provides dispersion of 5750 $\mathrm{ps}^2$ with bandwidth of 2 nm (around 250 GHz) and 11500 $\mathrm{ps}^2$ with bandwidth 1 nm (around 125 GHz). Both introduces 3 dB (around 30%) losses, therefore transmittance of such a device is around 70%, which is much better than DCF introducing the same amount of dispersion (not to mention spools of a

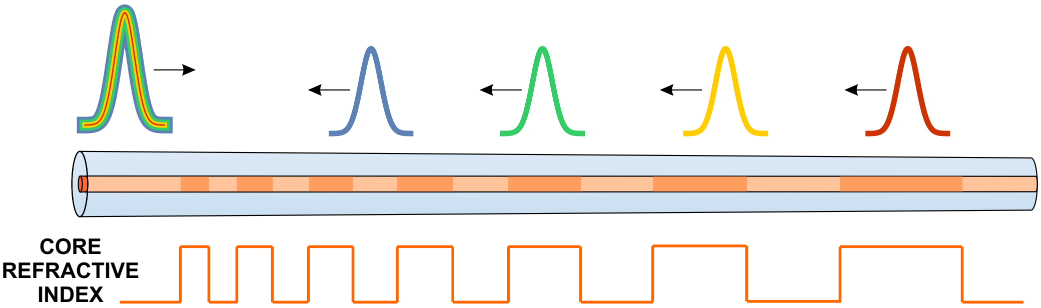

Figure 1.7: Working principle of chirped fiber Bragg grating.

very long optical fiber). However, one has to remember that CFBG is working in reflection, therefore an additional optical circulator is needed for each CFBG, which introduces 1-2 dB of loss, giving in total pessimistically 5 dB, but it still much better then losses in an optical fiber introducing the same amount of dispersion.

## 1.7. Previous works

The manipulating of spectrotemporal properties of optical pulses is a field, which can be divided into two subfields. The first one contains a manipulation of center frequency (wavelength) of light pulses. This subfield is now quite well developed, which results in many publications [30–36]. Now more effort is put into shaping single photons in spectrotemporal degree of freedom, especially converting their bandwidth [19–22, 37]. This kind of operations creates the second subfield of manipulating light pulses.

Both central frequency and bandwidth manipulation can be done by different techniques. One can use a direct phase modulation e.g. by using electrooptic phase modulation [19], which in principle should preserve quatumness of single photons simultaneously with low noise due to no need of introducing strong auxilary pump beams. The other technique uses nonlinear effects, e.g. sum-frequency generation, where as inputs are single photons and classical chirped pulses [20, 38]. This is however more challenging due to need of introducing an additional high power classical laser beam in order to achieve high conversion efficiency in the three wave mixing process. The additional laser is more challenging technically and also can introduce more noise to the quantum signal.

Here I briefly describe the main works creating the main achievements related to subject of this thesis. It includes bandwidth manipulation of quantum light by an electro-optic time lens and quantum optical waveform conversion using three wave mixing.

### 1.7.1. Bandwidth manipulation of quantum light by an electro-optic time lens

The main work leading to subject of this thesis was done by Karpiński *et al.* [19]. In his work a six-fold spectral bandwidth compression of heralded single-photon wavepackets was experimentally demonstrated. Space-time duality was used in order to obtain an idea of a time lens and a bandwidth converter, which was implemented by using an electrooptic phase modulator and dispersion coming from 256 m of an optical fiber. The quadratic phase of the time lens was obtained by approximating quadratic function by sine function. The whole setup was build in all-fiber technique, therefore it was possible to obtain low loss. The experiment was performed for photons with 830 nm (361 THz) central wavelength with input spectral bandwidth (FWHM) of 0.92 nm (401 GHz), generated from spontaneous parametric down conversion (SPDC). Six-fold compression into a bandwidth of 0.15 nm (65 GHZ) was obtained. This allowed a two-fold increase of photon flux in spectrally narrowband absorber, simulated by a bandpass filter.

This work confirmed a practical value of bandwidth converter scheme in use to manipulate a quantum light pulse. It was also shown a preservation of non-classical single photon statistics by measurement of the conditional degree of second order coherence $g^{(2)} \ll 1$.

### 1.7.2. Quantum optical waveform conversion using three wave mixing

One of the more important theoretical works on this subjcet was published by Kielpinski *et al.* [38]. It concerns a compressing and reshaping a single photons using three-wave mixing with a modulated classical field. As an input pulse was assumed to be a single photon emitted from quantum emitter as a monoexponentially decaying pulse. Then it undergoes a three-wave mixing process with a frequency-chirped classical light pulse. Once an appropriate phase of classical light pulse is chosen, the three-wave mixing product radiation obtains a spectral profile as a classical pulse, but retains the non-classical single photon statistics. Then the pulse is dechirped and the target pulse is obtained. They derived a phase profile which allows to reshape a monoexponentially decay into a Gaussian pulse, which is preferred for most of optical implementations of quantum network. This work also include a derivation of errors coming from dispersion.

The first stage of this idea was experimentally realized by Agha *et al.* [20]. It includes reshaping of an nanosecond light pulses with temporal shape of monoexponential decay with decay time of 6 ns and central wavelength of 980 nm. Then it is upconverted with 1550 nm chirped pulse in periodically poled lithium niobate obtaining a output pulse with translated central wavelength to 600 nm and spectrally broadened to desired spectral shape of 0.5 ns Lorentzian. The obtained efficiency was about 40%. The same was repeated for 1 ns input pulse and 0.25 ns output pulse. Whole experiment was done for weak classical light.

The same idea, but for quantum light was demonstrated by Lavoie *et al.* [22]. They experimentally demonstrated a spectral compression of single photons from 1740 GHz to 43 GHz with simultaneously central wavelength change from 800 nm to 400 nm. The obtained efficiency of the upconversion process was around 0.06% with 300 mW of average pump power. It is expected that one can reach higher efficiency by using periodically poled crystals and higher pump powers.

### 1.7.3. Temporal magnification

A slightly different application of a time lens was shown by Farsi *et al.* [29]. They shown a magnification of single photons temporal envelope function, but for different arrival times of photons. It results in increasing a temporal resolution and therefore they were able to distinguish two photons delayed by 2.7 ps by a detector with 90ps time jitter. The time lens was implemented by Bragg-Scattering four wave mixing (BS-FWM). It provides a great flexibility of choice of signal and pump frequencies and enables spectral manipulation over a large bandwidth. The observed efficiency was 70% for magnification factor of 119. The experiment was performed with weak classical light.

# Chapter 2

# Theory

This chapter contains derivations of equations describing the operating principle of an ideal bandwidth converter, both in forward and reverse mode, and for a temporal telescope. Because I did not find such derivation in literature, I used a formalism of propagators in order to obtain simple equations containing complete information about investigated device. In particular the transformations for Fourier-limited Gaussian pulses will be derived in the language of pulse widths.

## 2.1. Bandwidth limited Gaussian pulse

A bandwidth limited pulse, also known as a Fourier-limited or transform-limited pulse, is a pulse, whose duration is minimal for a given spectral bandwidth [39]. Here I will focus on Gaussian pulse shapes, because of e.g. calculation simplicity. The Gaussian transform-limited pulse does not contain any nonlinear phase, neither spectral nor temporal. A signal can be represented as a complex valued wavefunction [40]. It contains the knowledge about probability distribution of measuring single photons, in this case, in given time interval or spectral range, which can be obtained by taking squared absolute value of the wavefunction. It also contains an information about phase difference between different times or frequencies. It is coded as the argument of the wavefunction. The wavefunction of a Gaussian pulse represented in the spectral domain, is given by:

$$E(\omega) = \frac{1}{\pi^{1/4}\sqrt{\sigma_\omega}} \mathrm{e}^{-\frac{(\omega-\omega_0)^2}{2\sigma_\omega^2} - \mathrm{i}t_0(\omega-\omega_0)}, \tag{2.1}$$

where $\omega_0$ is the central frequency, $\sigma_\omega$ is the standard deviation of $\omega$, which contains information about spectral width of pulse and $t_0$ is the position of the pulse maximum in time. Its Fourier transform reads:

$$e(t) = \frac{1}{\pi^{1/4}\sqrt{\sigma_t}} \mathrm{e}^{-\frac{(t-t_0)^2}{2\sigma_t^2} + \mathrm{i}(t-t_0)\omega_0}, \tag{2.2}$$

where $\sigma_t$ is standard deviation in time, which contains information about the pulse duration. Later for simplicity we will be keeping $t_0 = 0$ and $\omega_0 = 0$, because it always can be reversed by simply a time or frequency shift, which is possible due to linearity of time and frequency scales. Therefore we can focus only on the variations with respect to the central frequency and time.

The simplified Gaussian pulse wavefunctions are as follows:

$$
\begin{aligned}
E(\omega) &= \frac{1}{\pi^{1/4}\sqrt{\sigma_\omega}}\,\mathrm{e}^{-\frac{\omega^2}{2\sigma_\omega^2}}, && (2.3)\\
e(t) &= \frac{1}{\pi^{1/4}\sqrt{\sigma_t}}\,\mathrm{e}^{-\frac{t^2}{2\sigma_t^2}}. && (2.4)
\end{aligned}
$$

The bandwidth limit in the language of Gaussians means, that the product of time and frequency width is minimal:

$$\sigma_t\sigma_\omega = 1 \tag{2.5}$$

However, what experimentalists typically use in a lab is full width at half maximum (FWHM) of the power spectrum of measured pulse, which is correlated with $\sigma$ in the following manner:

$$\mathrm{FWHM} = \frac{\sigma}{2\sqrt{\log 2}}. \tag{2.6}$$

The other change in experiment is the use of frequency $f$ rather, than angular frequency $\omega = 2\pi f$. Using this two properties we can express transform limitation (2.5) in a language of measured FWHMs:

$$\mathrm{FWHM}_t \;\cdot\; \mathrm{FWHM}_f = \frac{4\log 2}{2\pi} \approx 0.441. \tag{2.7}$$

It is also called the time bandwidth product (TBP).

## 2.2. Time and frequency propagators

In this section I will investigate different orders of Taylor expansion of the propagation constant $\beta(\omega)$:

$$
\begin{aligned}
\beta(\omega) = \beta(\omega_0) + \left.\frac{\mathrm{d}\beta}{\mathrm{d}\omega}\right|_{\omega=\omega_0}(\omega-\omega_0) + \frac{1}{2}\left.\frac{\mathrm{d}^2\beta}{\mathrm{d}\omega^2}\right|_{\omega=\omega_0}(\omega-\omega_0)^2 + \mathcal{O}\left(\omega^3\right) = \\
= \beta_0 + \beta_1(\omega-\omega_0) + \frac{\beta_2}{2}(\omega-\omega_0)^2 + \mathcal{O}\left(\omega^3\right).
\end{aligned} \tag{2.8}
$$

The propagating signal $E_{\text{in}}(\omega)$ in any dispersive medium is gaining an additional spectral phase $\psi(\omega)$, such that outgoing spectral signal reads:

$$E_{\text{out}}(\omega) = E_{\text{in}}(\omega)\mathrm{e}^{\mathrm{i}\psi(\omega)}. \tag{2.9}$$

And in time domain it is given by:

$$e_{\text{out}}(t') = \frac{1}{2\pi}\int \mathrm{d}t \int \mathrm{d}\omega\; e_{\text{in}}(t)\,\mathrm{e}^{\mathrm{i}\psi(\omega)}\mathrm{e}^{-\mathrm{i}\omega(t-t')}. \tag{2.10}$$

The additional spectral phase is proportional to the propagation constant $\beta(\omega)$ and propagation length $L$:

$$\psi(\omega) = -\beta(\omega)L. \tag{2.11}$$

It can by expanded in the same manner as the propagation constant:

$$\psi(\omega) = \psi_0 + \psi_1(\omega-\omega_0) + \frac{\psi_2}{2}(\omega-\omega_0)^2 + \mathcal{O}\left(\omega^3\right), \tag{2.12}$$

where

$$\psi_i = -\beta_i L \qquad \text{for} \qquad i = 0, 1, 2, 3, \ldots \tag{2.13}$$

### 2.2.1. Definition of propagator

A propagator is a description of evolution, in this case, of a signal pulse of light. It allows to relate the output pulse and the input pulse by some operation. One of the definitions says:

$$e_{\text{out}}(t') = \int \mathrm{d}t\ K_t(t, t')\, e_{\text{in}}(t), \tag{2.14}$$

where $K_t$ is the time propagator of some system. The same in the frequency domain:

$$E_{\text{out}}(\omega') = \int \mathrm{d}\omega\ K_\omega(\omega, \omega')\, e_{\text{in}}(\omega). \tag{2.15}$$

The language of propagators is very useful due to its versatility in terms of the input/output signals. In this section I will describe all the transformations by propagators.

### 2.2.2. Zeroth order of $\beta(\omega)$ expansion – $\beta_0$ – phase velocity

For propagation in nondispersive medium we will firstly consider only the zeroth order of the Taylor expansion of propagation constant $\beta(\omega)$ (2.8), which leaves only one component of the expansion:

$$\beta(\omega) = \beta_0 \qquad \Rightarrow \qquad \psi(\omega) = -\beta_0 L. \tag{2.16}$$

Then the signal acquires an additional phase depending only on the length of propagation $L$:

$$E_{\text{out}}(\omega) = E_{\text{in}}(\omega)\, \mathrm{e}^{-\mathrm{i}\beta_0 L}. \tag{2.17}$$

Therefore the frequency propagator reads:

$$VP_\omega(\omega, \omega') = \mathrm{e}^{-\mathrm{i}\beta_0 L}\, \delta(\omega - \omega'). \tag{2.18}$$

In the time domain it will result in:

$$e_{\text{out}}(t') = \frac{1}{2\pi} \int \mathrm{d}t \int \mathrm{d}\omega\ e_{\text{in}}(t) \mathrm{e}^{\mathrm{i}\psi(\omega)} \mathrm{e}^{-\mathrm{i}\,\omega(t-t')} = e_{\text{in}}(t')\, \mathrm{e}^{-\mathrm{i}\,\omega_0 \frac{L}{v_p}}, \tag{2.19}$$

where a phase velocity was introduced as follows:

$$\beta_0 = \frac{\omega_0}{v_p} \qquad \Rightarrow \qquad v_p = \frac{\omega_0}{\beta_0}. \tag{2.20}$$

Finally we achieved a time propagator for zeroth order of the expansion of the propagation constant:

$$VP_t(t, t') = \mathrm{e}^{-\mathrm{i}\,\omega_0 \frac{L}{v_p}}\, \delta(t - t'). \tag{2.21}$$

Therefore the zeroth order $\beta(\omega)$ expansion is responsible for moving wavefront with a phase velocity $v_p$.

### 2.2.3. First order of $\beta(\omega)$ expansion – $\beta_1$ – group velocity

If we consider propagation in the dispersive medium by taking into account only the linear dependency of the propagation constant on the frequency:

$$\beta(\omega) = \beta_1(\omega - \omega_0) \qquad \Rightarrow \qquad \psi(\omega) = -\beta_1(\omega - \omega_0)L. \tag{2.22}$$

Then for the signal in the frequency domain:

$$E_{\mathrm{out}}(\omega) = E_{\mathrm{in}}(\omega)\, \mathrm{e}^{-\mathrm{i}\beta_1 L(\omega-\omega_0)}. \tag{2.23}$$

Therefore the frequency propagator for the first order of the expansion reads:

$$GV_\omega(\omega,\omega') = \mathrm{e}^{-\mathrm{i}\beta_1 L(\omega-\omega_0)}\, \delta(\omega-\omega'). \tag{2.24}$$

Thus the $\beta_1$ is associated with a applying linear spectral phase. For the signal in the time domain we have:

$$e_{\mathrm{out}}(t') = \frac{1}{2\pi}\int \mathrm{d}t \int \mathrm{d}\omega\; e_{\mathrm{in}}(t)\mathrm{e}^{\mathrm{i}\psi(\omega)}\mathrm{e}^{-\mathrm{i}\,\omega(t-t')} = e_{\mathrm{in}}\left(t' - \frac{L}{v_g}\right)\, \mathrm{e}^{-\mathrm{i}\,\omega_0 \frac{L}{v_g}}, \tag{2.25}$$

where the group velocity $v_g$ was introduced as follows:

$$\beta_1 = \frac{1}{v_g} \qquad \Rightarrow \qquad v_g = \frac{1}{\beta_1}. \tag{2.26}$$

Finally the time propagator reads:

$$GV_t(t,t') = \mathrm{e}^{-\mathrm{i}\,\omega_0 \frac{L}{v_g}}\, \delta\left(t - \left(t' - \frac{L}{v_g}\right)\right). \tag{2.27}$$

Therefore the first order of $\beta(\omega)$ expansion is responsible for translation of the envelope function with group velocity $v_g$.

### 2.2.4. Second order of $\beta(\omega)$ expansion – $\beta_2$ – group velocity disperison

For propagation in a dispersive medium with taking into account only the quadratic frequency dependency of the propagation constant:

$$\beta(\omega) = \frac{\beta_2}{2}(\omega-\omega_0)^2 \qquad \Rightarrow \qquad \psi(\omega) = -\frac{\beta_2}{2}(\omega-\omega_0)^2 L. \tag{2.28}$$

Then for the signal in the frequency domain:

$$E_{\mathrm{out}}(\omega) = E_{\mathrm{in}}(\omega)\, \mathrm{e}^{-\mathrm{i}\frac{\beta_2 L}{2}(\omega-\omega_0)^2}. \tag{2.29}$$

Therefore the frequency propagator describing contribution of the second order of the $\beta(\omega)$ expansion reads:

$$GDD_\omega(\omega,\omega') = \mathrm{e}^{-\mathrm{i}\frac{\beta_2 L}{2}(\omega-\omega_0)^2}\, \delta\left(\omega-\omega'\right). \tag{2.30}$$

Thus $\beta_2$ is associated with applying a quadratic spectral phase. Then for the signal in the time domain:

$$e_{\mathrm{out}}(t') = \sqrt{\frac{1}{2\pi\mathrm{i}\beta_2 L}}\, \mathrm{e}^{-\mathrm{i}\frac{\beta_2 L}{2}\omega_0^2} \int \mathrm{d}t\; e_{\mathrm{in}}(t)\, \mathrm{e}^{\mathrm{i}\frac{\left(t'-(t-\omega_0\beta_2 L)\right)^2}{2\beta_2 L}}. \tag{2.31}$$

Finally the time propagator of group velocity dispersion reads:

$$GDD_t(t,t') = \sqrt{\frac{1}{2\pi\mathrm{i}\beta_2 L}}\mathrm{e}^{-\mathrm{i}\frac{\beta_2 L}{2}\omega_0^2}\, \mathrm{e}^{\mathrm{i}\frac{\left(t'-(t-\omega_0\beta_2 L)\right)^2}{2\beta_2 L}}. \tag{2.32}$$

It can be rewritten using the total amount of dispersion/chirp $\Phi = -\beta_2 L$:

$$GDD_t(t,t';\Phi) = \sqrt{\frac{\mathrm{i}}{2\pi\Phi}}\mathrm{e}^{\mathrm{i}\frac{\Phi}{2}\omega_0^2}\,\mathrm{e}^{-\mathrm{i}\frac{\left(t'-(t+\omega_0\Phi)\right)^2}{2\Phi}}. \tag{2.33}$$

Therefore applying a quadratic spectral phase causes a convolution of the time signal with a quadratic phase. Its consequence is a change of the pulse duration, in particular for a Fourier limited pulse it is resulting in broadening of the pulse in time. In other words different frequencies travels with different velocities – group velocity is dispersed.

### 2.2.5. Spectral shift – applying a linear phase in time

In analogy to the group velocity, where applying a linear phase in frequency causes a temporal shift of whole envelope function, one can postulate a spectral shift as applying a linear phase in time:

$$SS_t(t,t';\delta\omega) = \mathrm{e}^{-\mathrm{i}\,\Delta\omega\, t}\,\delta(t'-t). \tag{2.34}$$

Then in the frequency domain:

$$SS_\omega(\omega,\omega',\Delta\omega) = \frac{1}{2\pi}\int \mathrm{d}t \int \mathrm{d}t'\; SS_t(t,t';\Delta\omega) = \delta\big(\omega - (\omega' - \Delta\omega)\big). \tag{2.35}$$

Thus for the signal in the frequency domain:

$$E_{\mathrm{out}}(\omega') = E_{\mathrm{in}}(\omega' - \Delta\omega). \tag{2.36}$$

Therefore applying a linear phase in time causes a spectral shift by $\Delta\omega$.

### 2.2.6. Time lens – applying a quadratic phase in time

A time lens is proposed as a counterpart of a spatial lens using space-time duality and it is obtained by applying an additional, quadratic phase in time, thus its propagator in time simply reads (1.20):

$$TL_t(t,t';K) = \mathrm{e}^{\mathrm{i}\frac{Kt^2}{2}}\,\delta(t'-t). \tag{2.37}$$

Then for the signal in the frequency domain:

$$E_{\mathrm{out}}(\omega') = \frac{1}{2\pi}\int \mathrm{d}t \int \mathrm{d}\omega\; E_{\mathrm{in}}(\omega)\,\mathrm{e}^{\mathrm{i}\frac{Kt^2}{2}}\,\mathrm{e}^{\mathrm{i}(\omega-\omega')t} = \sqrt{\frac{\mathrm{i}}{2\pi K}}\int \mathrm{d}\omega\; E_{\mathrm{in}}(\omega)\,\mathrm{e}^{-\mathrm{i}\frac{(\omega-\omega')^2}{2K}}. \tag{2.38}$$

Therefore applying a quadratic phase in time causes a convolution with quadratic phase in frequency domain, which can result in spectral broadening or compressing of the signal pulse depending on the input signal. Finally the frequency propagator for the time lens is given by:

$$TL_\omega(\omega,\omega';K) = \sqrt{\frac{\mathrm{i}}{2\pi K}}\,\mathrm{e}^{-\mathrm{i}\frac{(\omega-\omega')^2}{2K}}. \tag{2.39}$$

Using the concept of spectral shift one can easily explain the working principle of the time lens. If one approximates a quadratic phase with linear sections, one can clearly see that different slopes for different times are responsible for shifting different parts of signal, in particular for chirped pulse to the center of signal, see Fig. 2.1.

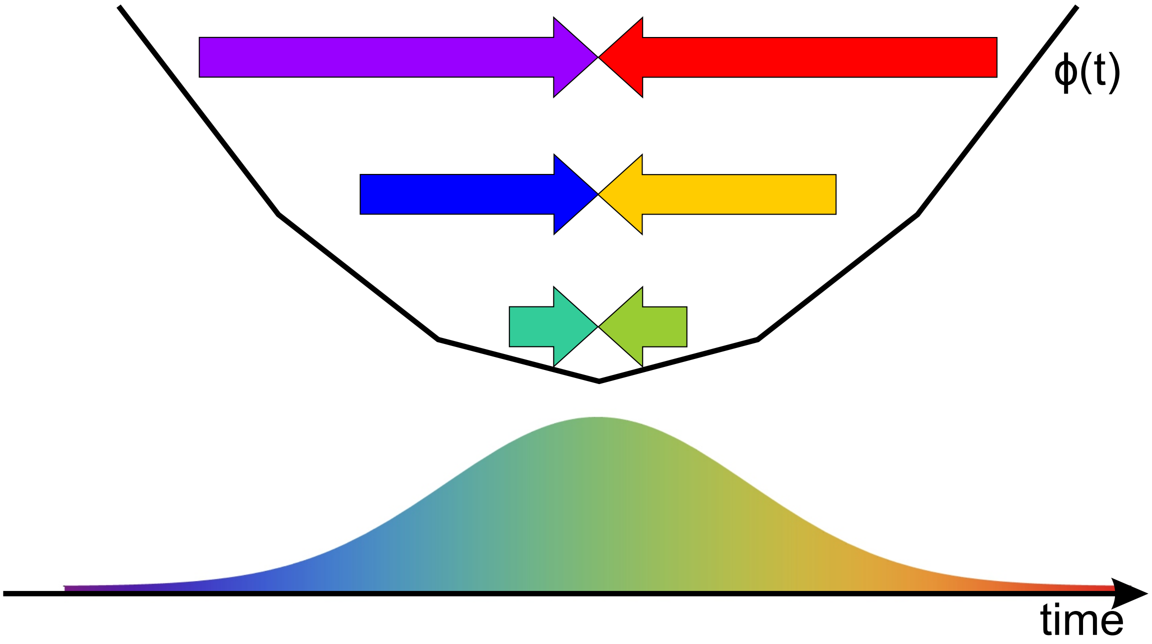

Figure 2.1: A working principle of a time lens. A quadratic phase can be divided into linear sections, each with different slope and therefore different spectral shift of a chirped pulse.

### 2.2.7. Summary

Applying different kinds of phase, both temporal and spectral, can be associated with different effects:

- constant spectral phase – phase velocity
- linear spectral phase – group velocity
- quadratic spectral phase – group velocity dispersion (GVD)
- linear temporal phase – spectral shift
- quadratic temporal phase – time lens

In order to take into account several effects (kinds of phases) one should integrate the required propagators using the same boundaries.

| Name | Frequency domain | Time domain |
|---|---|---|
| $VP$<br>phase velocity | $VP_\omega\left(\omega,\omega'\right) = \mathrm{e}^{-\mathrm{i}\frac{L}{v_p}\omega_0}\,\delta\left(\omega'-\omega\right)$ | $VP_t\left(t,t'\right) = \mathrm{e}^{-\mathrm{i}\omega_0\frac{L}{v_p}}\,\delta\left(t'-t\right)$ |
| $GV$<br>group velocity | $GV_\omega\left(\omega,\omega'\right) = \mathrm{e}^{-\mathrm{i}\frac{L}{v_g}(\omega-\omega_0)}\,\delta\left(\omega'-\omega\right)$ | $GV_t\left(t,t'\right) = \mathrm{e}^{-\mathrm{i}\omega_0\frac{L}{v_g}}\,\delta\left(t'-\frac{L}{v_g}\right)$ |
| $GDD$<br>group velocity dispersion | $GDD_\omega\left(\omega,\omega';\Phi\right) = \mathrm{e}^{\mathrm{i}\frac{\Phi}{2}(\omega-\omega_0)^2}\,\delta\left(\omega'-\omega\right)$ | $GDD_t\left(t,t';\Phi\right) = \sqrt{\frac{\mathrm{i}}{2\pi\Phi}}\,\mathrm{e}^{\mathrm{i}\frac{\Phi}{2}\omega_0^2}\,\mathrm{e}^{-\mathrm{i}\frac{\left(t'-(t+\omega_0\Phi)\right)^2}{2\phi}}$ |
| $SS$<br>spectral shift | $SS_\omega\left(\omega,\omega';\Delta\omega\right) = \delta\big(\omega-(\omega'-\Delta\omega)\big)$ | $SS_t\left(t,t';\Delta\omega\right) = \mathrm{e}^{-\mathrm{i}\,\Delta\omega\,t}\,\delta(t'-t)$ |
| $TL$<br>time lens | $TL_\omega\left(\omega,\omega';K\right) = \sqrt{\frac{\mathrm{i}}{2\pi K}}\,\mathrm{e}^{-\mathrm{i}\frac{\left(\omega-\omega'\right)^2}{2K}}$ | $TL_t\left(t,t';K\right) = \mathrm{e}^{\mathrm{i}\,\frac{Kt^2}{2}}\,\delta\left(t'-t\right)$ |

## 2.3. Temporal systems based on GVD and time lenses

In this section I will derive different kinds of temporal optical systems, such as bandwidth converter based on a single time lens or a temporal telescope consisting of two time lenses. Also effects of transformation of a Fourier limited pulse will be shown.

For simplicity from now on $\omega_0$ is assumed to be zero and $\omega$ can be interpreted as the variation of frequency from some central frequency. The same applies to $t_0$ and $t$.

### 2.3.1. Dispersion effects

As shown in (2.29) in the frequency domain the group velocity dispersion only applies an additional phase, so the power spectrum in frequency domain remains unchanged for any input signal, in particular for a bandwidth limited pulse. In the time domain, on the other hand, we have a convolution with a quadratic phase in time:

$$e_{\text{out}}(t') = \sqrt{\frac{\mathrm{i}}{2\pi\Phi}} \int \mathrm{d}t\; e_{\text{in}}(t)\; \mathrm{e}^{\mathrm{i}\frac{\left(t'-t\right)^2}{2\Phi}}. \tag{2.40}$$

For a transform-limited pulse (2.4) it results in temporal broadening:

$$e_{\text{out}}(t') = \frac{1}{\pi^{1/4}\sqrt{\sigma_t}} \sqrt{\frac{\mathrm{i}}{2\pi\Phi}} \int \mathrm{d}t\; \mathrm{e}^{-\frac{t^2}{2\sigma_t^2}}\; \mathrm{e}^{\mathrm{i}\frac{\left(t'-t\right)^2}{2\Phi}} = \frac{1}{\pi^{1/4}\sqrt{\sigma_t\left(1-\mathrm{i}\frac{\Phi}{\sigma_t^2}\right)}}\; \mathrm{e}^{-\frac{t'^2}{2\sigma_t^2\left(1+\frac{\Phi^2}{\sigma_t^4}\right)}-\mathrm{i}\frac{t'^2}{2\left(\Phi+\frac{\sigma_t^4}{\Phi}\right)}}. \tag{2.41}$$

One can clearly see a broadening of pulse duration:

$$\sigma_t^2 \to \sigma_t^2\left(1+\frac{\Phi^2}{\sigma_t^4}\right). \tag{2.42}$$

Thus the output pulse is no longer transform-limited:

$$\sigma_{t,\text{in}}\sigma_{\omega,\text{in}} = 1 \to\; \sigma_{t,\text{out}}\sigma_{\omega,\text{out}} = \sqrt{1+\frac{\Phi^2}{\sigma_{t,\text{in}}^4}}. \tag{2.43}$$

Also an additional quadratic temporal phase is applied.

### 2.3.2. Single time lens

A time propagator of a time lens is given by (2.37) and it applies an additional quadratic phase in time, so the power spectrum of any signal in the time domain remains unchanged. However, the output signal in the frequency domain will be a convolution with a quadratic spectral phase, as in (2.39):

$$E_{\text{out}}(\omega') = \sqrt{\frac{\mathrm{i}}{2\pi K}} \int \mathrm{d}\omega\; E_{\text{in}}(\omega)\; \mathrm{e}^{-\mathrm{i}\frac{(\omega'-\omega)^2}{2K}}. \tag{2.44}$$

In particular, for a transform limited-pulse (2.3) it results in spectral broadening:

$$E_{\text{out}}(\omega') = \frac{1}{\pi^{1/4}\sqrt{\sigma_\omega}} \sqrt{\frac{\mathrm{i}}{2\pi\, K}} \int \mathrm{d}t\; \mathrm{e}^{-\frac{t^2}{2\sigma_t^2}}\; \mathrm{e}^{\mathrm{i}\frac{\left(t'-t\right)^2}{2\Phi}} = \frac{1}{\pi^{1/4}} \sqrt{\frac{\mathrm{i}}{\sigma_\omega\left(\frac{K}{\sigma_\omega^2}-\mathrm{i}\right)}}\; \mathrm{e}^{-\frac{\omega'^2}{2\sigma_\omega^2\left(1+\frac{K^2}{\sigma_\omega^4}\right)}+\mathrm{i}\frac{\omega'^2}{2(\sigma_\omega^2+K)}}. \tag{2.45}$$

After passing through a time lens, the spectral width is broader:

$$\sigma_\omega^2 \to \sigma_\omega^2 \left(1 + \frac{K^2}{\sigma_\omega^4}\right). \tag{2.46}$$

And the output pulse is no longer transform-limited:

$$\sigma_{t,in}\sigma_{\omega,in} = 1 \to \ \sigma_{t,out}\sigma_{\omega,out} = \sqrt{1 + \frac{K^2}{\sigma_{\omega,in}^4}}. \tag{2.47}$$

Also an additional quadratic spectral phase is applied.

### 2.3.3. Bandwidth converter: dispersion → time lens

In order to change simultaneously (i.e. in one device) the spectral width and duration of a transform limited pulse, one can use a device consisting of optical fiber or other optical element that introduces dispersion and lengthens the input pulse, followed by a time lens, which compresses the spectral width.

In the spatial domain if one would set up a propagation (diffraction) length $f$ and a lens with focal length $f$ afterwards, than this system will work as an optical Fourier transform, where characteristic lengths are scaled with $f$. The same reasoning applies in the time domain – it is a Fourier transform from time to frequency and broadening in time.
The propagator of such system in the time domain reads:

$$BC_+(t', t; \Phi, K) = \sqrt{\frac{\mathrm{i}}{2\pi\Phi}}\ \mathrm{e}^{-\mathrm{i}\frac{(t'-t)^2}{2\Phi} + \mathrm{i}\frac{Kt'^2}{2}}. \tag{2.48}$$

It is obvious that a transform-limited pulse in the time domain will behave exactly as in case (2.41), where dispersion causes temporal broadening, and also a time lens will apply additional temporal phase, but it does not change the power spectrum of a pulse in time. Therefore the output pulse in the time domain is as follows (omitting the constant phase factor):

$$e_{\text{out}}(t') = e_{\text{in}}\left(\frac{t'}{\sqrt{1 + \frac{\Phi^2}{\sigma_t^4}}}\right)\ \mathrm{e}^{-\mathrm{i}\frac{t'^2}{2\left(\Phi + \frac{\sigma_t^4}{\Phi}\right)}}. \tag{2.49}$$

This additional phase, however, causes changes in the frequency domain. The propagator of bandwidth converter in frequency is given by:

$$BC_+(\omega, \omega'; \Phi, K) = \sqrt{\frac{\mathrm{i}}{2\pi K}}\ \mathrm{e}^{\mathrm{i}\frac{\Phi}{2}\omega^2 - \mathrm{i}\frac{(\omega'-\omega)^2}{2K}} = \sqrt{\frac{\mathrm{i}}{2\pi K}}\ \mathrm{e}^{\mathrm{i}\frac{\omega^2}{2}\left(\Phi - \frac{1}{K}\right) - \mathrm{i}\frac{\omega'}{2K}(\omega' - 2\omega)}. \tag{2.50}$$

For equivalent of spatial Fourier transform condition ($L = f$):

$$\Phi = \frac{1}{K}, \tag{2.51}$$

the propagator simplifies into the form:

$$BC_+\left(\omega, \omega'; \Phi = \frac{1}{K}\right) = \sqrt{\frac{\mathrm{i}}{2\pi K}}\ \mathrm{e}^{-\mathrm{i}\frac{\omega'^2}{2K} + \mathrm{i}\frac{\omega'}{K}\omega}. \tag{2.52}$$

For any input signal one gets:

$$E_{\text{out}}(\omega') = \sqrt{\frac{\mathrm{i}}{2\pi K}}\mathrm{e}^{-\mathrm{i}\frac{\omega'^2}{2K}} \int \mathrm{d}\omega\; E_{\text{in}}(\omega)\; \mathrm{e}^{\mathrm{i}\frac{\omega'}{K}\omega} = \sqrt{\frac{\mathrm{i}}{K}}\mathrm{e}^{-\mathrm{i}\frac{\omega'^2}{2K}}\; e_{\text{in}}\left(\frac{\omega'}{K}\right), \tag{2.53}$$

where $e_{\text{in}}(t)$ is the inverse Fourier transform of $E_{\text{in}}(\omega)$. It implies that the spectrum of the output signal is a Fourier transform of a temporal input signal rescaled by $\frac{1}{K}$ and with additional quadratic spectral phase. In particular for a transform limited Gaussian pulse one can get:

$$E_{\text{out}}(\omega') = \frac{1}{\pi^{1/4}}\sqrt{\frac{\mathrm{i}}{K\sigma_t}}\;\mathrm{e}^{-\frac{\omega^2}{2(K\sigma_t)^2}-\mathrm{i}\frac{\omega'^2}{2K}}. \tag{2.54}$$

In summary the pulse width broadens:

$$\sigma_t \to \sigma_t\sqrt{1+\frac{\Phi^2}{\sigma_t^4}}. \tag{2.55}$$

And the spectral width is scales:

$$\sigma_\omega \to K\sigma_t = \frac{K}{\sigma_\omega}. \tag{2.56}$$

The time-bandwidth product in angular frequency is given by:

$$\sigma_{t,\text{out}}\sigma_{\omega,\text{out}} = \sqrt{1+\frac{\sigma_{t,\text{in}}^4}{\Phi^2}} = \sqrt{1+\frac{K^2}{\sigma_{\omega,\text{in}}^4}}. \tag{2.57}$$

In order to get $p$-fold compression one needs:

$$K = \frac{\sigma_\omega^2}{p} \qquad \text{and} \qquad \Phi = \frac{p}{\sigma_\omega^2}. \tag{2.58}$$

### 2.3.4. Bandwidth converter: time lens → dispersion

This device is working in the same manner as the previous one, but in opposite direction. In the frequency domain:

$$BC_-(\omega,\omega';\Phi,K) = \sqrt{\frac{\mathrm{i}}{2\pi K}}\mathrm{e}^{-\mathrm{i}\frac{(\omega'-\omega)^2}{2K}+\mathrm{i}\frac{\Phi}{2}\omega'^2} = \sqrt{\frac{\mathrm{i}}{2\pi K}}\mathrm{e}^{-\mathrm{i}\frac{\omega'}{2}\left(\Phi-\frac{1}{K}\right)-\mathrm{i}\frac{\omega}{2K}(\omega-2\omega')}. \tag{2.59}$$

Which for the condition:

$$\Phi = \frac{1}{K}, \tag{2.60}$$

simplifies to:

$$BC_-\left(\omega,\omega';\Phi=\frac{1}{K}\right) = \sqrt{\frac{\mathrm{i}}{2\pi K}}\mathrm{e}^{-\mathrm{i}\frac{\omega^2}{2K}+2\frac{\omega'}{K}\omega}. \tag{2.61}$$

Then for a Fourier limited pulse (2.3) as an input one gets (omitting the constant phase factor):

$$E_{\text{out}}(\omega') = E_{\text{in}}\left(\frac{\omega'}{\sqrt{1+\frac{K^2}{\sigma_\omega^4}}}\right)\;\mathrm{e}^{\mathrm{i}\frac{\omega'^2}{2K\left(1+\frac{K^2}{\sigma_\omega^4}\right)}}. \tag{2.62}$$

In the temporal domain propagator of this device reads:

$$BC_{-}(t,t';\Phi,K) = \sqrt{\frac{\mathrm{i}}{2\pi\Phi}}\,\mathrm{e}^{-\mathrm{i}\frac{(t'-t)^2}{2\Phi}+\mathrm{i}\frac{Kt^2}{2}}. \tag{2.63}$$

Using the above condition (2.60) it simplifies to:

$$BC_{-}\left(t,t';\Phi=\frac{1}{K}\right) = \sqrt{\frac{\mathrm{i}}{2\pi\Phi}}\,\mathrm{e}^{\mathrm{i}\frac{t'^2}{2\Phi}-\mathrm{i}\frac{t'}{\Phi}t}. \tag{2.64}$$

For any input pulse one will get:

$$e_{\mathrm{out}}(t') = \sqrt{\frac{\mathrm{i}}{2\pi\Phi}}\,\mathrm{e}^{\mathrm{i}\frac{t'^2}{2\Phi}}\int \mathrm{d}t\; e_{\mathrm{in}}(t)\,\mathrm{e}^{-\mathrm{i}\frac{t'}{\Phi}t} = \sqrt{\frac{\mathrm{i}}{\Phi}}\,\mathrm{e}^{\mathrm{i}\frac{t'^2}{2\Phi}}\,E_{\mathrm{in}}\left(\frac{t'}{\Phi}\right), \tag{2.65}$$

where $E_{\mathrm{in}}(\omega)$ is a Fourier transform of an input pulse in the time domain $e_{\mathrm{in}}(t)$. It implies that the temporal output signal is a Fourier transform of a the spectrum of an input signal rescaled by $\frac{1}{\Phi}$ and with an additional quadratic spectral phase. In particular for a transform limited Gaussian pulse (2.4) one can get:

$$e_{\mathrm{out}}(t') = \frac{1}{\pi^{1/4}}\sqrt{\frac{\mathrm{i}}{\sigma_\omega\Phi}}\,\mathrm{e}^{-\frac{t'^2}{2(\Phi\sigma_\omega)^2}+\mathrm{i}\frac{t'^2}{2\Phi}}. \tag{2.66}$$

In summary the spectrum broadens:

$$\sigma_\omega \to \sigma_\omega\sqrt{1+\frac{K^2}{\sigma_\omega^4}}. \tag{2.67}$$

And the pulse width is scales:

$$\sigma_t \to \Phi\sigma_\omega = \frac{\Phi}{\sigma_t}. \tag{2.68}$$

Therefore the time bandwidth product (expressed in angular frequencies and $\sigma$ – not FWHM):

$$\sigma_{t,\mathrm{out}}\sigma_{\omega,\mathrm{out}} = \sqrt{1+\frac{\sigma_{\omega,\mathrm{in}}^4}{K^2}} = \sqrt{1+\frac{\Phi^2}{\sigma_{t,\mathrm{in}}^4}}. \tag{2.69}$$

### 2.3.5. Temporal telescope

A temporal telescope, as pointed out in section 1.5, consists of dispersion $\Phi$ and two time lenses – first diverging $K_1 < 0$, second converging $K_2 > 0$, which satisfy the temporal telescope condition (1.27):

$$\Phi = \frac{1}{K_1} + \frac{1}{K_2}. \tag{2.70}$$

The frequency propagator of the time telescope is as follows:

$$TT\left(\omega,\omega';K_1,K_2,\Phi=\frac{1}{K_1}+\frac{1}{K_2}\right) = \mathrm{i}\sqrt{\frac{K_1}{K_2}}\,\mathrm{e}^{-\frac{\mathrm{i}}{2}\left(\frac{\omega^2}{K_1}+\frac{\omega'^2}{K_2}\right)}\,\delta\left(\omega+\frac{K_1}{K_2}\omega'\right). \tag{2.71}$$

Therefore for any input pulse spectrum:

$$E_{\mathrm{out}}(\omega') = \mathrm{i}\sqrt{\frac{K_1}{K_2}}\,\mathrm{e}^{-\frac{\mathrm{i}}{2}\frac{\omega'^2}{K_2}\left(1+\frac{K_1}{K_2}\right)}\,E_{\mathrm{in}}\left(-\frac{K_1}{K_2}\omega\right). \tag{2.72}$$

The above transformation for $K_1 < 0$, $K_2 > 0$ and $|K_1| > |K_2|$ results in spectral compression of the input pulse. Particularly for a transform-limited input pulse one gets:

$$E_{\mathrm{out}}(\omega') = \frac{\mathrm{i}}{\pi^{1/4}\sqrt{\sigma_\omega}}\sqrt{\frac{K_1}{K_2}}\,\mathrm{e}^{-\frac{\omega'^2}{2\left(\frac{K_2}{K_1}\sigma_\omega\right)^2}-\frac{\mathrm{i}}{2}\frac{\omega'^2}{K_2}\left(1+\frac{K_1}{K_2}\right)}. \tag{2.73}$$

Therefore the change in spectral width of a bandwidth-limited pulse is as follows:

$$\sigma_\omega \rightarrow \frac{K_2}{K_1}\sigma_\omega. \tag{2.74}$$

The temporal propagator of a time telescope reads:

$$\begin{aligned} TT\left(t,t';K_1,K_2,\Phi=\frac{1}{K_1}+\frac{1}{K_2}\right) &= \sqrt{\frac{\mathrm{i}}{2\pi\Phi}}\,\mathrm{e}^{\mathrm{i}\frac{K_1t^2}{2}-\mathrm{i}\frac{(t'-t)^2}{2\Phi}+\mathrm{i}\frac{K_2t'^2}{2}} = \\ &= \sqrt{\frac{\mathrm{i}}{2\pi\Phi}}\,\mathrm{e}^{\mathrm{i}\frac{(K_1t+K_2t')^2}{2(K_1+K_2)}}. \end{aligned} \tag{2.75}$$

For bandwidth-limited pulse one gets (omitting a constant phase factor):

$$e_{\mathrm{out}}(t') = e_{\mathrm{in}}\left(\frac{t'}{\frac{K_1}{K_2}\sqrt{\left(1+\frac{(K_1+K_2)^2}{K_1^4\sigma_t^4}\right)}}\right)\,\mathrm{e}^{\mathrm{i}\frac{1+K_1K_2\left(1+\frac{(K_1+K_2)^2}{K_1^4\sigma_t^4}\right)}{\frac{K_1}{K_2}(K_1+K_2)\left(1+\frac{K_1+K_2}{K_1^4\sigma_t^4}\right)}t'^2}. \tag{2.76}$$

Which means that pulse duration lengthens:

$$\sigma_t \rightarrow \frac{K_1}{K_2}\sigma_t\sqrt{\left(1+\frac{(K_1+K_2)^2}{K_1^4\sigma_t^4}\right)}. \tag{2.77}$$

In order to obtain a $p$-fold compression in frequency the following conditions must be satisfied:

$$p = \frac{K_1}{K_2} \qquad \text{and} \qquad \Phi = \frac{1}{K_1}+\frac{1}{K_2}. \tag{2.78}$$

Therefore eq. (2.77) becomes:

$$\sigma_t \rightarrow p\sigma_t\sqrt{\left(1+\frac{\Phi^2}{p^2\sigma_t^4}\right)}. \tag{2.79}$$

The time bandwidth product (expressed in angular frequencies and $\sigma$ – not FWHM):

$$\sigma_{t,\mathrm{out}}\sigma_{\omega,\mathrm{out}} = \sqrt{\left(1+\frac{\Phi^2}{p^2\sigma_t^4}\right)}. \tag{2.80}$$

## 2.4. Summary

In this section I have derived analytical equations describing bandwidth converters and a temporal telescope. In the ideal case both provide large compress factors for any input pulse width. However, in a realistic case, where losses depend on the amount of needed dispersion, it is more challenging. Eq. (2.55) says, that the needed amount of dispersion in the bandwidth converter is increasing linearly with compress factor and quadratically with input spectral width. Therefore achievement of high compress factors for spectrally narrow pulses (e.g. 1 GHz) is impossible due to loss caused by dispersive elements. However in the case of the temporal telescope the amount of needed dispersion can be varied, see eq. (2.78). The compress factor is given by the ratio of time lenses parameters $K_1$ and $K_2$, and the needed chirp can be varied by scaling both of them. This scaling corresponds to spectral broadening of the input pulse by the first lens and therefore increasing the spectral width of the pulse between the lenses in order to achieve more chirp per the same length unit. But still this is a semirealistic case due to not taking into account electronic imperfections, such as frequency response of an AWG. This together with using different types of lenses, such as a Fresnel lens, where a quadratic phase is wrapped by using a modulo operation, it is extremely hard or even impossible to solve this problem analytically, therefore I addressed this challenge by using numerical simulations.

# Chapter 3

# Simulation methods

In this section I will describe methods of simulations. To do this firstly I will introduce the CUDA architecture, which was used to implement simulation in order to obtain good computation performance. Then I will describe the used algorithm and define the parameters used in the simulations.

## 3.1. CUDA architecture

Compute Unified Device Architecture (CUDA) is an architecture of graphical processing units (GPUs) developed by Nvidia. Its main purpose is to give to scientists a comfortable environment, which allows to use GPUs in simulations and numerical computations.

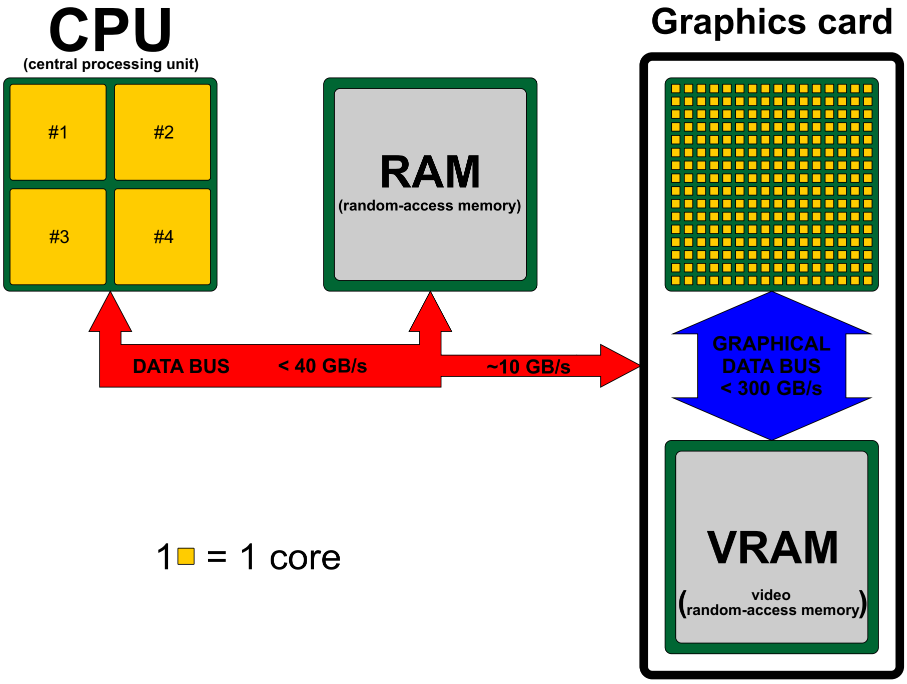

Figure 3.1: Schematic view of computer and CUDA architecture. Note that this is a very simplified view containing only the key elements. Widths of bus arrows are nearly proportional to their data bandwidth.

The GPU architecture differs from the CPU (central processing unit). Each of the few cores of the CPU is optimized in order to obtain a possibly small latency (time interval between querying and receiving a variable from memory). Also CPU's clocks (the number of operations per second) are very high. However one CPU has on average only 8 cores and each one operates fast, but still there is just a few of them. Also the data bus between the CPU and the memory (RAM) is fast, but it can transmit only single variables at once.

The GPU is built in an exactly opposite manner. The GPU consists of hundreds or even thousands of the CUDA cores. The graphical card has also its own memory (VRAM or GRAM) and a data bus between GPU and GRAM. The graphical data bus, in contrast to the main data bus (between CPU and RAM), is optimized to transmit a large amount of data in parallel, however it has a quite a large latency. Therefore the GPU is a great tool for massive calculations, especially those using linear algebra or Fast Fourier Transform (FFT). However in order to obtain a performance speed up of calculation one should use only the graphical memory, without often transferring the data to the main memory, because it costs a lot of time due to small bandwidth of the data bus connecting RAM and VRAM. Therefore one is limited to the particular amount of memory which is implemented physically on the graphics card.

## 3.2. Algorithm

All simulations are based on calculation of FFT for vary large sample. A typical algorithm for simulating the temporal telescope looks as follows:

1. create a one dimensional array and initialize it with a Gaussian wave packet (2.3) on a frequency grid $f$ from $-\Delta f$ to $\Delta f$ with $\frac{2\Delta f}{N}$ step, where $N$ is the number of used points,
2. calculate a normalized Inverse Fast Fourier Transform (IFFT) in order to get a time wavepacket on a grid in time $t$ from $-\frac{N}{4\Delta f}$ to $\frac{N}{4\Delta f}$ with $\frac{1}{2\Delta f}$ step,
3. for each cell of the wavepacket array add an appropriate phase, according to the chosen parameter $K_1$ of a first lens,
4. calculate a normalized FFT in order to get a frequency wavepacket,
5. for each cell of the wavepacket array add an appropriate phase, according to the chosen parameter $\Phi$ of dispersion,
6. calculate a normalized IFFT in order to get a temporal wavepacket,
7. for each cell of the wavepacket array add an appropriate phase, according to the chosen parameter $K_2$ of a second lens,
8. take the desired time measurement (e.g. peak power, FWHM etc.),
9. calculate a normalized FFT in order to get a frequency wavepacket,
10. take the desired frequency measurement (e.g. peak power, power in a bandpass filter etc.).

Algorithm for simulating the bandwidth converter is the same as above, but without steps 2 to 4.

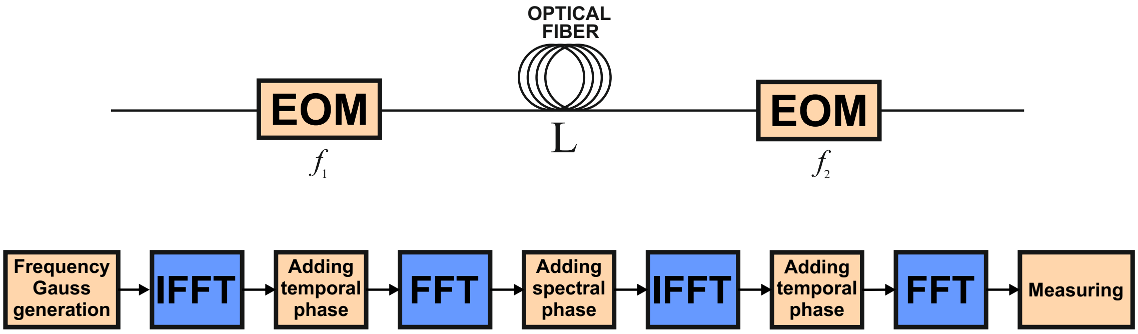

Figure 3.2: Schematic view of temporal telescope and its algorithmic counterpart.

The results of the simulation are highly dependent on the precision of the added phase, both in time and frequency. A particular challenge was to simulate a telescope, that compresses the spectral width of a Gaussian pulse by 4 orders of magnitude (e.g. from FWHM = 50 GHz to FWHM = 5 MHz). To do this one needs a good resolution in frequency for the output pulse in order to cover a Gaussian peak with at least several points, and also a good resolution in time for input pulse in order to well reproduce an applied phase by first lens. The second task – resolution in time – is critical, because any errors made at this stage cause a domino effect resulting in high noise in the output pulse.

To achieve the desired accuracy a large sample size is needed. Discretization of a frequency signal is described by two parameters – the number of points $N$ and the frequency range $2\Delta f$ if the grid is made as $f \in (-\Delta f, \Delta f)$. Then the frequency resolution is $\frac{2\Delta f}{N}$. These two parameters define also a time range, given by $t \in \left(-\frac{N}{4\Delta f}, \frac{N}{4\Delta f}\right)$ and the time resolution given by $\frac{1}{2\Delta f}$. In particular one needs $10^6$ points in order to obtain a picosecond time resolution and a megahertz spectral resolution. In all my simulations I used $N = 72 \times 1024 \times 1024$. A sample of this size (more than $2^{26}$) causes the need for of lot of computation power. However, the whole algorithm can be easily paralellized. Therefore I have chosen an approach based on the CUDA environment [41], taking advantage of graphical processing unit, which offers a massive paralellization of computation and also optimized libraries with implemented FFT (called cuFFT [42]) and linear algebra (cuBLAS [43]). It provides shorter computation times than by doing it in the traditional, sequential way. However this approach is limited by the memory contained on the graphics card. A typical graphics card has 4 GB of memory, when only an array of $N = 72 \times 1024 \times 1024$ double precision complex numbers takes more than 1.125 GB. A memory overhead is needed in order to perform computations such as FFT. In this work I used GeForce GTX 970 graphic card with total VRAM of 4 GB, but there exists others with more memory such as GeForce GTX TITAN X with 12 GB of memory or even specially designed for the numerical computations such as Tesla K40 from Nvidia. Code in CUDA/C++ used in simulations is listed in Appendix A.

## 3.3. Dispersion and losses assumptions

Chirped fiber Bragg gratings (see section 1.6) provide a large amount of dispersion with relatively low levels of loss. The most important feature for this thesis is loss per amount of dispersion. However, in contrast to optical fiber, they cannot be divided into smaller pieces, so dispersion they provide is discretized, which is problematic in visualizing simulation results.

Therefore in order to get a good measure of transmission for any dispersion I use a value of dispersion per transmittance (or loss) of 50% – $\phi$. Then the total transmittance $T$ of dispersion $\Phi$ is given by formula:

$$T = 0.5^{\frac{\Phi}{\phi}} \tag{3.1}$$

From now $\phi$ will be called a dispersion unit. For realistic at this time values for bandwidth of 125 GHz $\phi$ becomes 11500 ps$^2$ and for bandwidth of 250 GHz $\phi$ becomes 5750 ps$^2$. The numbers are based on information obtained from CFBG manufacturers.

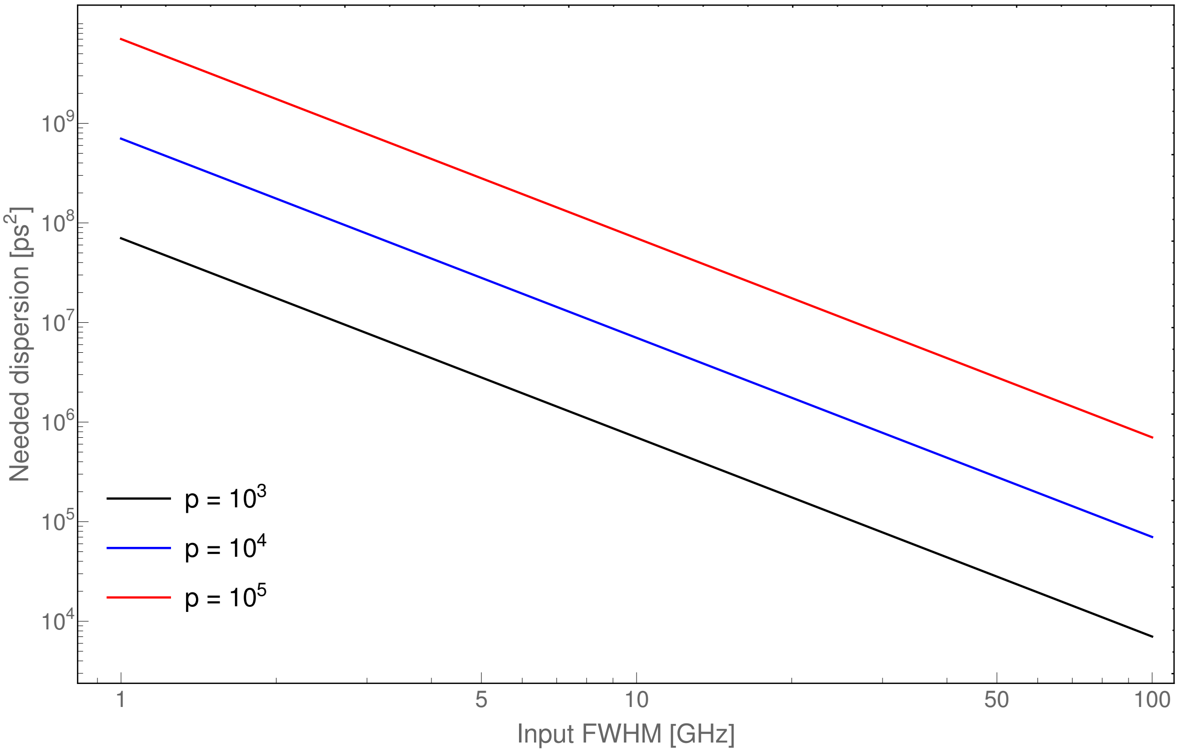

Figure 3.3: The amount of dispersion required for different input widths and compression factors $p$.

### Pulse duration broadening

According to eq. (2.42) in order to obtain $p$-fold broadening of pulse duration one needs:

$$p = \sqrt{1 + \frac{\Phi^2}{\sigma_t^4}} \quad \overset{p \gg 1}{\Rightarrow} \quad \Phi = p\sigma_t^2 = \frac{p}{\sigma_\omega^2} \tag{3.2}$$

where $\sigma$'s are input widths of a transform limited pulse and $\Phi$ is the total amount of dispersion. It is critical to notice the needed amount of dispersion scales quadratically with the inverse input spectral width.

## 3.4. Performance measures

In order to measure performance of a bandwidth converter and a temporal telescope I introduced two measures, the efficiency and the enhancement. Both are based on measuring the power contained in the targeted FWHM. The efficiency is than the ratio between powers measured in the targeted FWHM of a simulated output signal $P_{\text{out,sim}}$ for given frequency response and other parameters and an output pulse $P_{\text{out,ideal}}$ for time lens with ideal quadratic phase:

$$\text{Efficiency} = \frac{P_{\text{out,sim}}}{P_{\text{out,ideal}}} \tag{3.3}$$

It does not include loss of dispersive elements and its value is between 0 (bad) and 1 (good).

The enhancement describes an improvement given by using an interface (a bandwidth converter or a temporal telescope), in comparison to the case without any interface device. It can also include the transmittance of dispersive elements, therefore it is defined as:

$$\text{Enhancement} = T\frac{P_{\text{out,sim}}}{P_{\text{in,ideal}}} \tag{3.4}$$

where $P_{\text{in,ideal}}$ is power of input pulse contained in target FWHM and $T$ is total transmittance of dispersive elements. If enhancement value is more the 1, then it is better to use an interface device, rather than not using it (assuming that there are no other losses except those from dispersive elements).

## 3.5. Agreement with theory

In order to check the agreement of numerical simulations with theory (see chapter 2) a bandwidth converter was simulated. Width of the input pulse was set to 100 GHz and the compression rate was set to 100, therefore width of the output in ideal case should be a 1 GHz width Gaussian pulse. Output power spectra of both bandwidth converters based on an ideal quadratic lens and ideal Fresnel lens (see section 4.3) are perfectly consistent with the theory, see Fig. 3.4 . If one will take into account the AWG frequency response, the output pulse

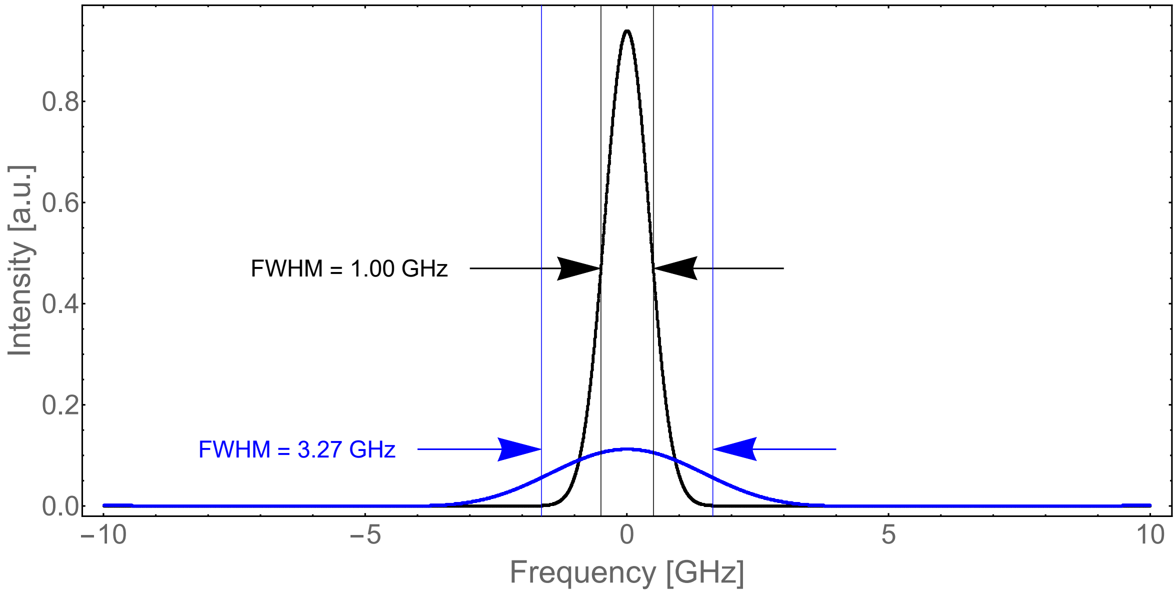

Figure 3.4: Output signal of compression a 100 GHz width Gaussian pulse with compression rate 100 into 1 GHz width for ideal frequency response (black) and realistic with cutoff frequency 30 GHz (blue).

should be wider then ideal due to distortions in the phase introduced by the time lens. The output signal is wider and therefore weaker in the targeted FWHM, but its sum is still normalized to one, therefore part of signal was not compressed and it should create additional "bumps" away from the main peak, see Fig. 3.5.

Basing on these results I conclude that proposed algorithm is working very well due to good agreement with theory and it is able to simulate more advanced systems, such as a temporal telescope.

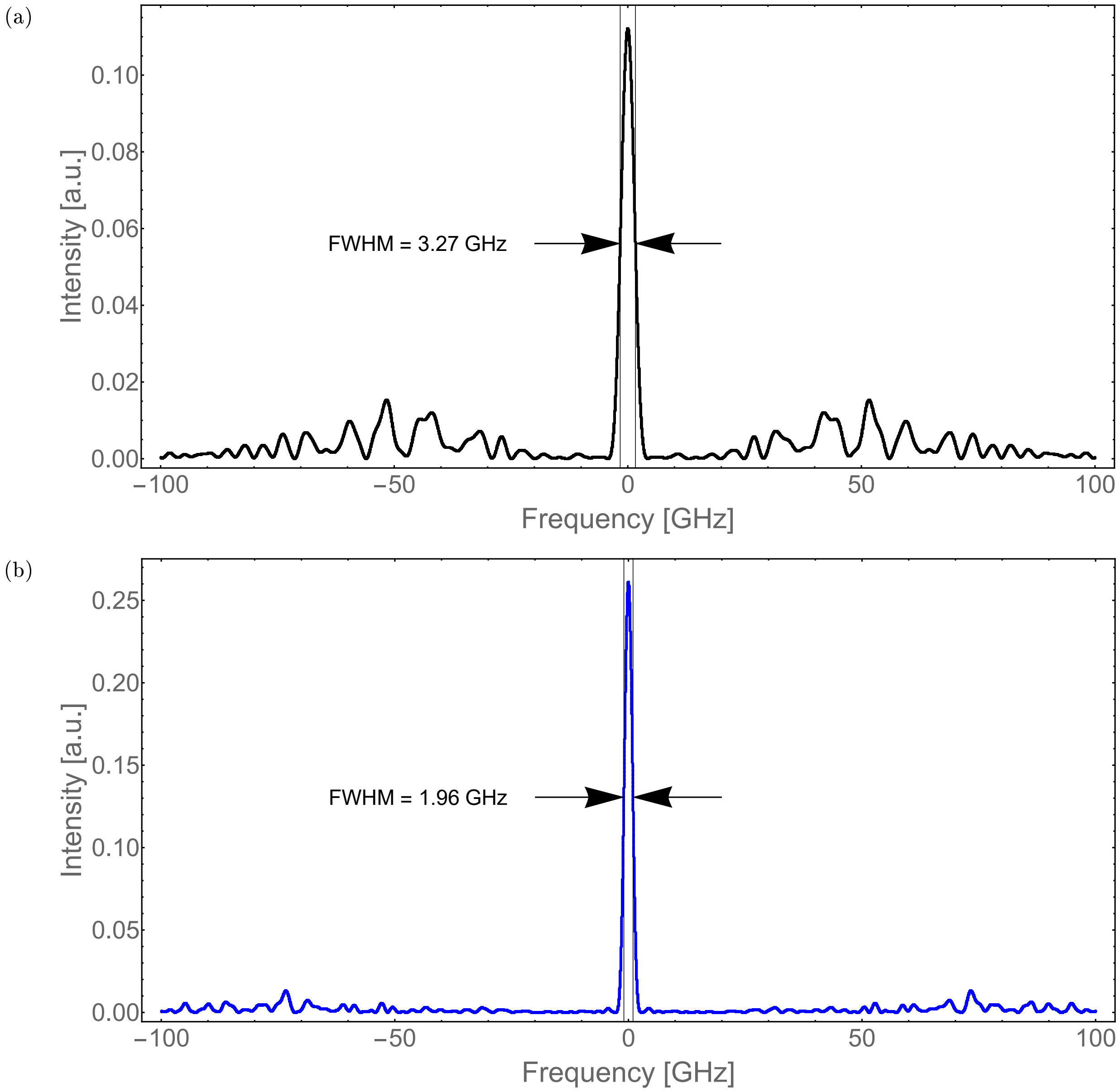

Figure 3.5: Output signal of compression a 100 GHz width Gaussian pulse with compression rate 100 into 1 GHz width for frequency response with cutoff (a) 30 GHz and (b) 60 GHz.

# Chapter 4

# Simulations of bandwidth converter

In this section I will investigate how different kinds of time lenses behave by investigating their performance as a part of the previously introduced bandwidth converter , where transform limited pulse goes through a dispersive element of total amount of dispersion $\Phi$ and then the chirped pulse passes the time lens of investigated type. Also I will estimate problems related to the particular time lens implementations and I will consider the use of elements, whose parameters are expected to become accessible in the foreseeable future in order to look for future implementations and to check if this is a promising area of research.

I will begin with analyzing effects of different kind of time lenses only, without including dispersion losses. Then I will investigate a bandwidth converter with taking into account losses generated by dispersive elements. At the end of this chapter I will look at the performance of a temporal telescope including dispersion losses.

## 4.1. Ideal parabolic time lens

The most straightforward implementation of the time lens is to use an AWG in order to apply a quadratic temporal phase to a given optical pulse. This approach is however very sensitive to the maximal depth modulation A, that is limited by the EOM. It results in reduction of the aperture of the time lens, because of allowed maximal phase modulation depth, see Fig. 4.1. Since the time lens is placed at the end of whole device (both bandwidth converter and temporal telescope), where pulses are wide in time (the narrower spectral width, the wider pulse duration) it is a very inefficient way of implementing a time lens as shown in Fig. 4.2. Efficiency of such a lens for accessible phase depth modulation of tens of radians does not exeed 5%.

Even with accessible very high amplitudes, there is another problem with frequency re-

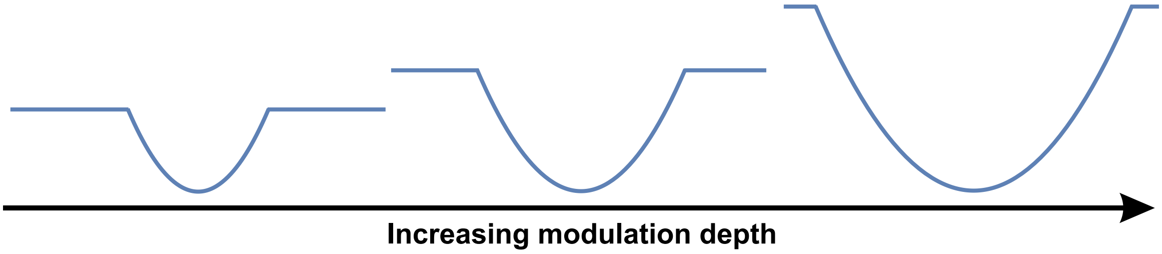

Figure 4.1: Aperture depending on the phase modulation depth.

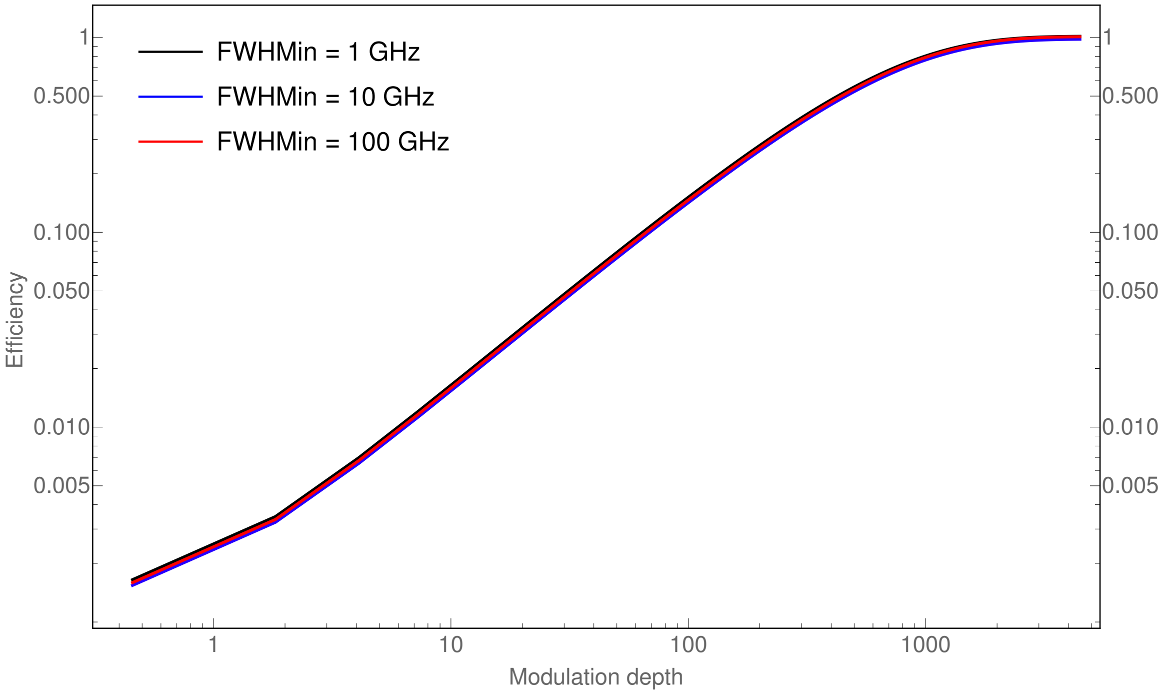

Figure 4.2: Efficiency for different phase modulation depths.

sponse of AWG, which also limits an aperture of time lens. This is because the steeper slope contains more higher frequencies and frequency response causes deviations from the ideal phase. Its effects however are much smaller then those caused by maximum amplitude restriction.

## 4.2. Sinusoidal time lens

The easiest way to implement time lens in a lab is to use a sinusoidal waveform generator instead of an arbitrary waveform generator and than by using an approximation of sin(cos) function:

$$\cos(x) = 1 - \frac{x^2}{2} + \mathcal{O}\left(x^4\right), \tag{4.1}$$

one can obtain a parabolic phase $\frac{Kt^2}{2}$ with sinusoidal signal with frequency $f_r$ and amplitude A with certain aperture:

$$\frac{Kt^2}{2} \approx A \sin\left(2\pi f_r t\right) \qquad \text{with condition} \qquad f_r = \frac{1}{2\pi}\sqrt{\frac{K}{A}}. \tag{4.2}$$

This approach has however its flaws. Despite the ease of generating and amplifying a sinusoidal signal, still the amplitude restriction causes that the above approximation works fine only for a limited aperture, see Fig. 4.3. For small maximum amplitudes of sine signal the time aperture shortens, therefore a time lens based on a sinusoidal signal works with almost 100% efficiency for small compression rates around $p = 10$. Than it starts linearly decrease, which means that enhancement (without losses) remains at a constant value, see Fig. 4.4.

However, this the efficiency of this kind of time lens does not depend on the width of the input pulse. It results from the fact that if we choose twice smaller input and output widths than also the frequency of sine signal is changed in the same way, therefore it is the same case,

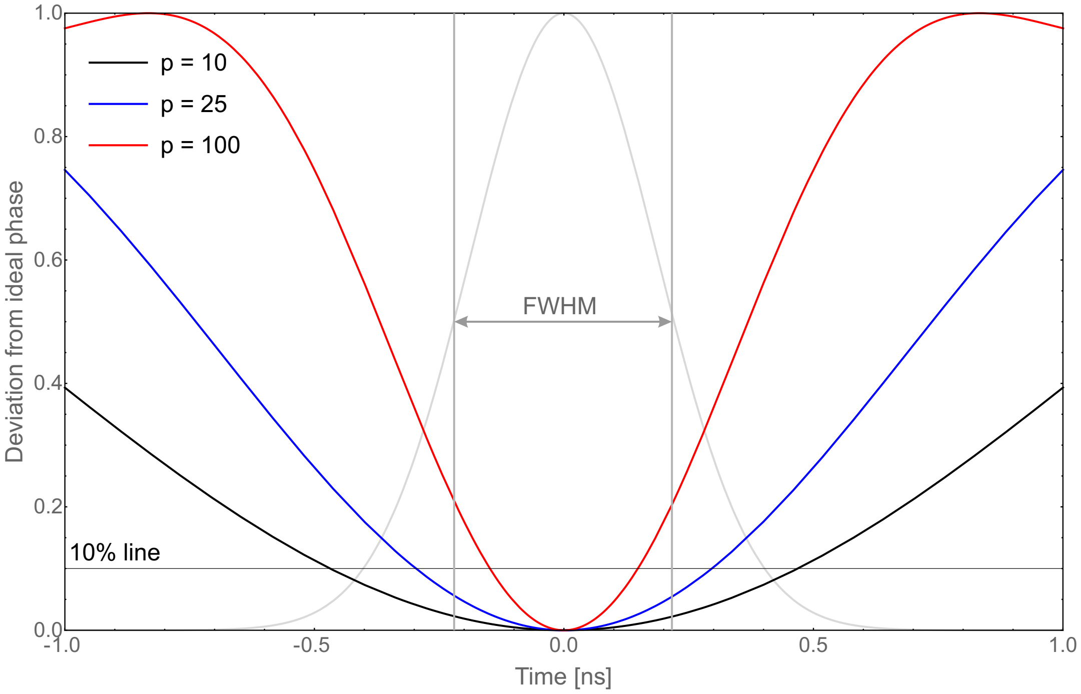

Figure 4.3: Deviation from ideal quadratic phase for different compression factors. The input pulse width is (black) 10 GHz, (blue) 25 GHz and (black) 100 GHz. The output pulse width is 1 GHz (440 ps) (grey line). Note, that for 100-fold compression the 10% deviation is reached much earlier than the FWHM point placed at 0.22 ns.

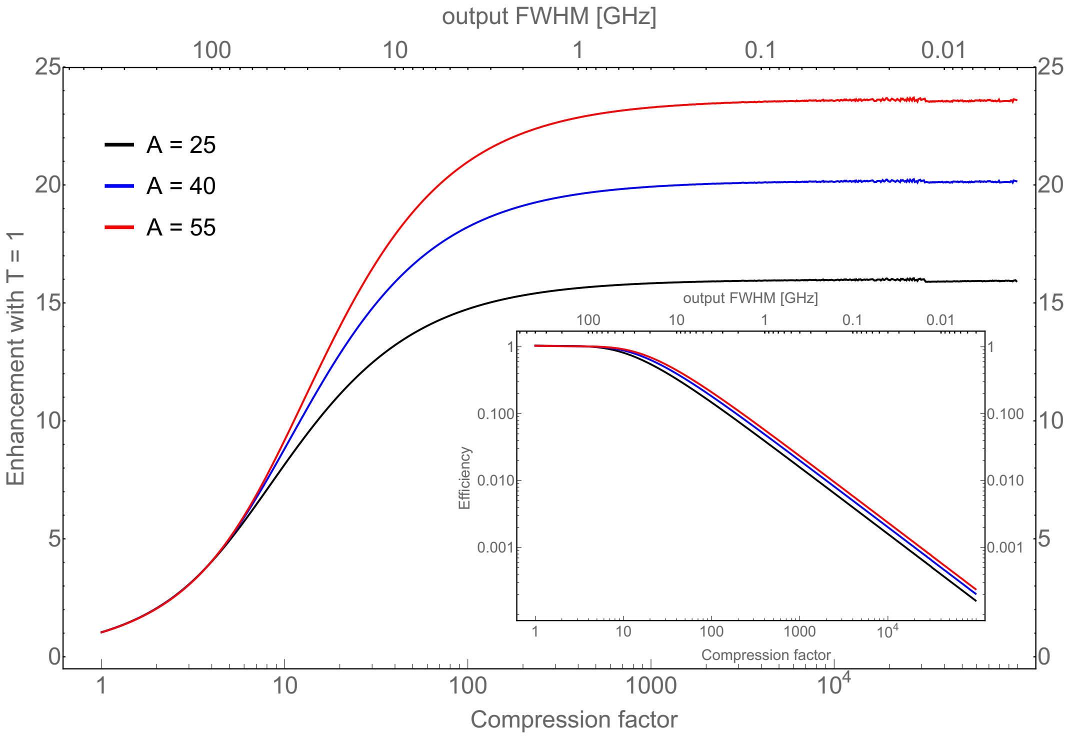

Figure 4.4: Enhancement (without losses from dispersion elements) by using (sinusoidal) bandwidth converter for compressing a pulse with any FWHM using different phase modulation depths. The inset shows the efficiency of such device.

but in different time and frequency scales. However the total amount of dispersion needed does change, which will result in significantly smaller efficiency for narrower spectral pulses due to exponential increase in the amount of needed dispersion. This effect will be discussed later, see section 4.4.

High efficiency of the sinusoidal time lens for small compression factors shows that this kind of lens can be useful for compressing (or broadening) a spectral pulse by maximum two orders of magnitude.

## 4.3. Fresnel time lens

The idea of a Fresnel time lens is borrowed from spatial domain by space-time duality [37,44]. In the spatial domain this kind of lenses provide wide apertures and short focal lengths. A Fresnel lens uses fact that the phase is periodic, therefore any change in phase is equal to a change of phase in only the first period range $(0, 2\pi)$ by using the modulo operation. After translating this idea to the temporal domain a challenge arises, because modulo operation causes incintinuities in the time waveform, which in realistic implementation become an issue due to frequency response of electronic part of implementation, especially of an AWG (see sec. 1.6.2).

Frequency response of an electronic device works like a lowpass filter, therefore its smoothing effect results in modification of time waveform, see Fig. 4.5. There are two kind of modifications:

- decrease of the amplitude in time – linear in the case of a realistic response or more complicated in the case of cutoff response and compensated real response,
- smoothing of turning points.

The first negative effect can be decreased only by using mathematical compensation. Instead of setting an AWG to generate an ideal Fresnel phase one can firstly increase contribution of high frequencies, e.g. by amplifying amplitudes of Fourier transform of the required phase and subsequently transforming it back to the time domain by using an inverse Fourier transform. Now the frequency response and frequency amplification should compensate, what can by clearly seen of Fig. 4.5 (b) – without compensation and (c) – with compensation to 30 GHz. A downside of this approach is that the maximal compensated frequency is limited and also such compensation increase maximum amplitude of signal before DAC, therefore an vertical resolution is decreasing.

The second negative effect can be fought by slightly modifying an idea of Fresnel lens. Instead of using modulation depth of $2\pi$ one can use multiples of $2\pi$, e.g. $4\pi$, $6\pi$ etc. This approach requires more sophisticated, faster amplifiers, because AWGs and EOMs are usually designed to get a change of phase of $\pi$. However, it can benefit in increasing the efficiency of the time lens. The reason for this is that each turning point corresponds to an unwanted opposite slope, which causes that a signal in a region of this slope is spectrally shifted in other way than we want it – instead of compression in this regions one gets broadening due to changed sign of slope. By using a larger modulation depth one gets fewer turning points, than in the case of using modulation depth of $2\pi$. It gets even better, because the most significant in view of efficiency, are turning points closest to center of light pulse and by using higher modulation depths the first turning point moves outside pulse center, where it becomes less significant. However higher modulation depth means also that the region of the slope connected to turning point is wider, therefore one needs to optimize it, see Fig. 4.6.

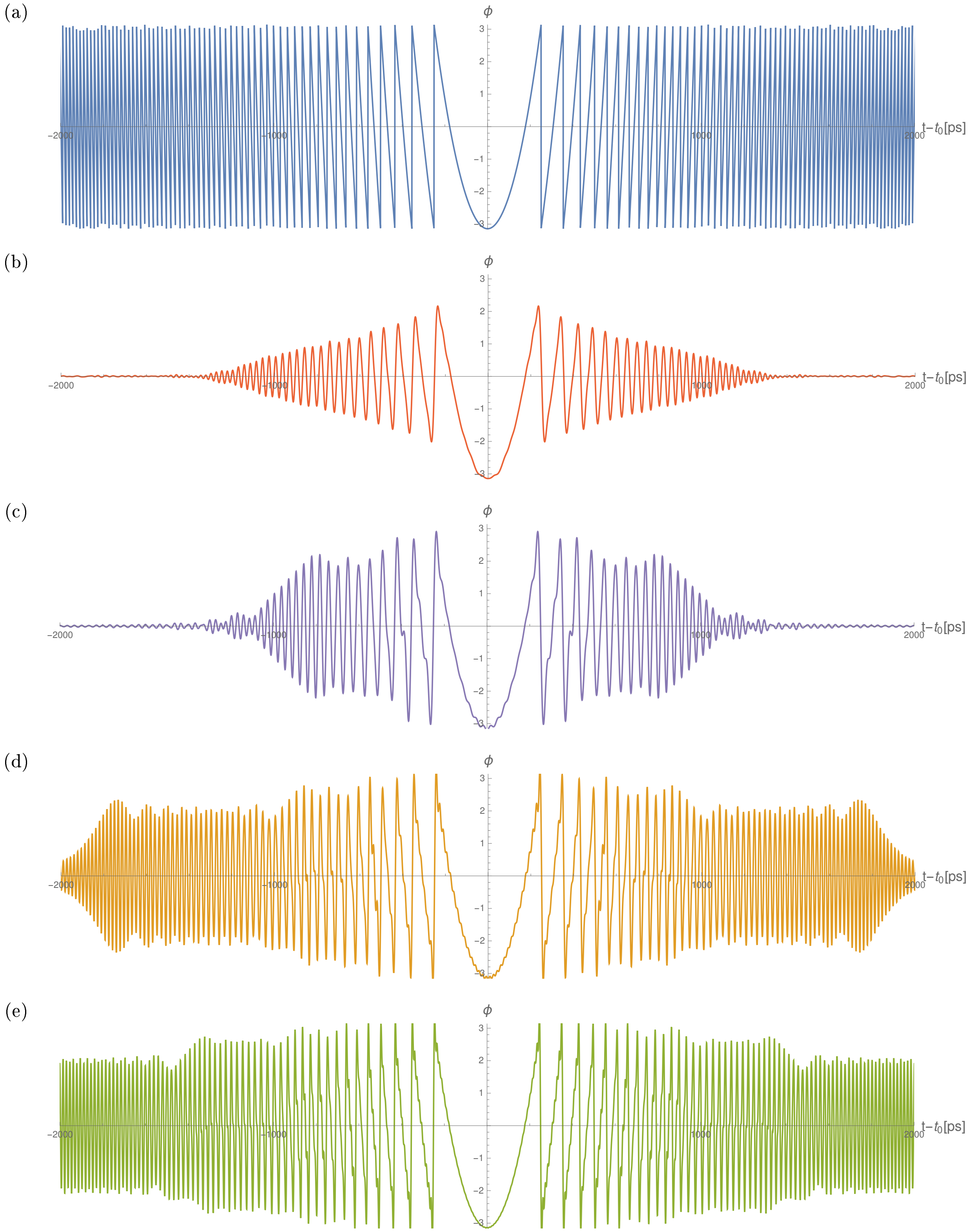

Figure 4.5: Fresnel phase for $K = 200\,\mathrm{GHz}^2$ and for different frequency conditions: (a) ideal case, (b) real frequency response of Keysight M8196A, (c) real frequency response of Keysight M8196A with mathematical compensation, (d) frequency response with cutoff at $f_c = 60\,\mathrm{GHz}$, (e) frequency response with cutoff at $f_c = 90\,\mathrm{GHz}$.

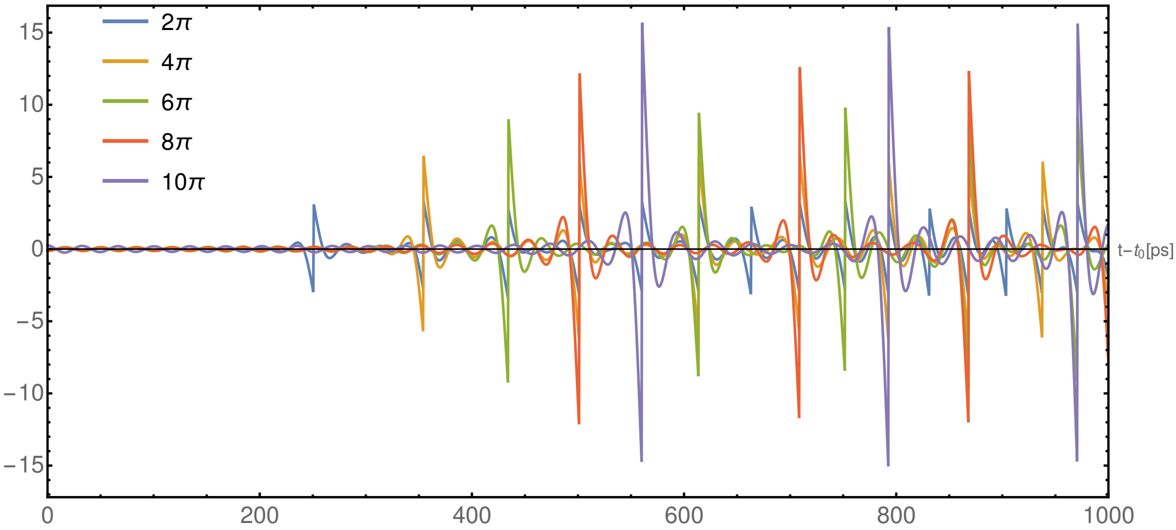

Figure 4.6: Deviation from ideal Fresnel phase for different modulation depths.

This approach is limited by the maximal modulation depth and an vertical resolution of the AWG.

By using higher modulation depth the bandwidth converter efficiency can be increased several times, however other effects occur – the oscillation of efficiency with compression rate. The amplitude of this oscillation is however decreasing with compression rate. It is caused by

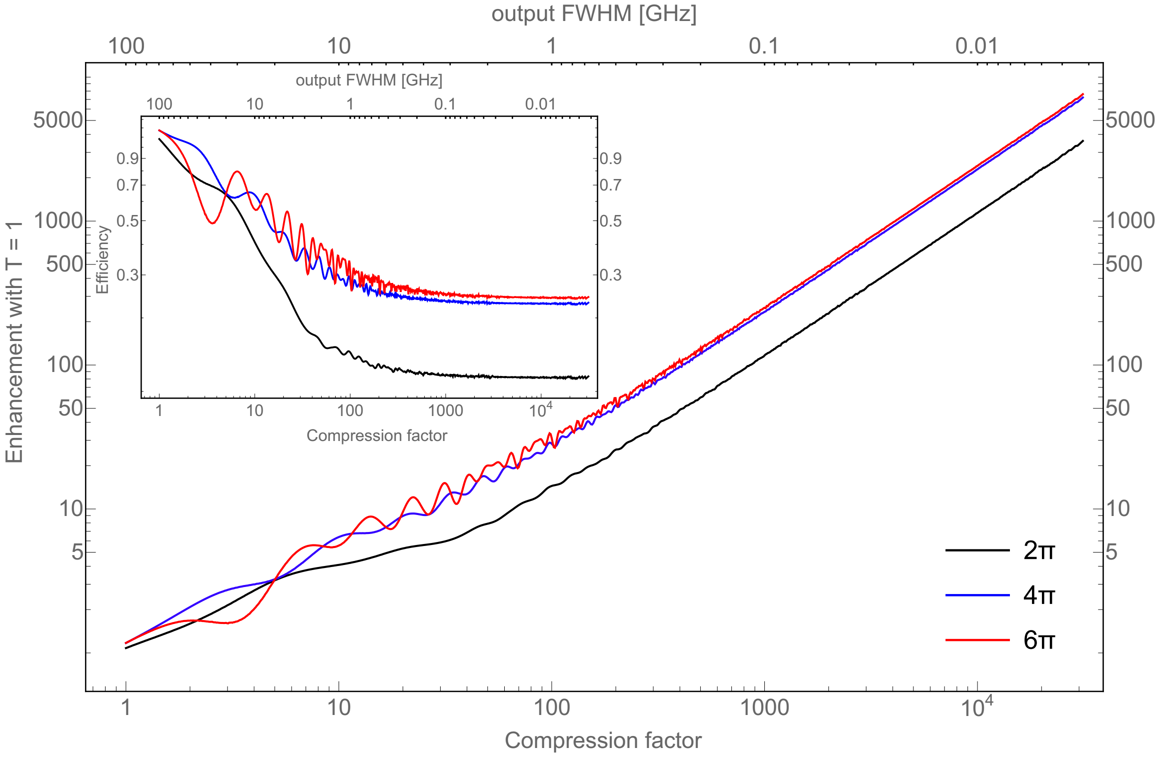

Figure 4.7: Enhancement and efficiency of Fresnel time lens for different modulation depths. Width of input pulse is 100 GHz and the frequency response is realistic with 30 GHz cutoff.

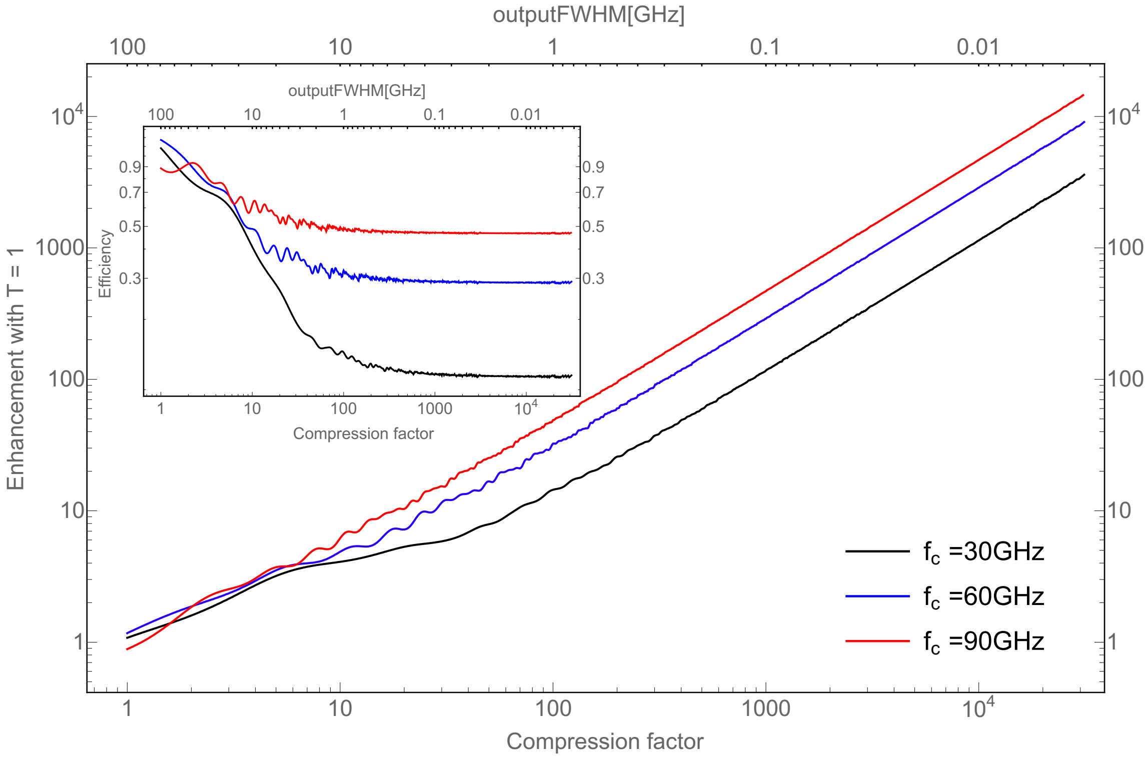

Figure 4.8: Enhancement and efficiency of a Fresnel time lens for different frequency responses. Width of input pulse is 100 GHz and the modulation depth is $2\pi$.

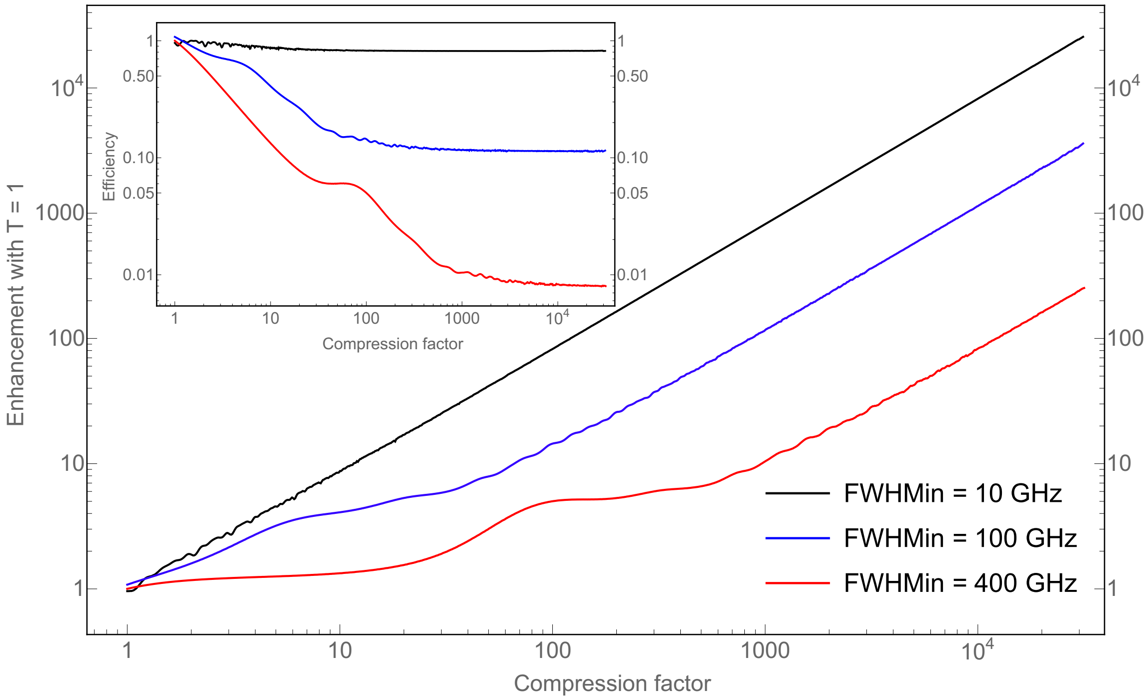

Figure 4.9: Enhancement and efficiency of a Fresnel time lens for different spectral widths of input pulses for real frequency response, compensated to 30 GHz.

appearance of the next turning point within the pulse width.

The efficiency of the Fresnel lens for small compression factors is high and it decreases nearly linearly, however with increasing compression factor this decrease is slowing down and efficiency is reaching an asymptotic value. In case of a 100 GHz input at compression factor of $10^5$ efficiency of real (achievable now) time lens is 12%, which seems small, however it gives an enhancement at level of 5000 (one has to remember that dispersion losses are not included here).

An improvement of efficiency can be certainly obtained by using better, faster AWGs (with higher cutoff frequencies). Here I present compressing a 100 GHz width pulse using AWGs that are expected to become available in the in future, with cutoff frequencies $f_c = 60\,\mathrm{GHz}$ (near future) and $f_c = 90\,\mathrm{GHz}$ (further future). It can be clearly seen that by using twice as fast AWGs one can obtain more then twice better performance of the Fresnel time lens, see Fig. 4.8

The Fresnel lens, unlike sinusoidal lens, exhibits different efficiency for different widths of input pulse. As in sinusoidal lens, where scaling the input pulse bandwidth causes the same scaling of the driving signal, in Fresnel lens one gets the same scaling effect, however it is true only for the ideal case, with ideal frequency response. In the real case one has a finite frequency response, which in contrast to the signal, does not scale, therefore for narrower spectral pulses, the phase is closer to ideal phase. This entails the better performance of the time lens, see Fig. 4.9.

In view of another application of Fresnel lens beside the bandwidth converter, a time telescope which has one additional free parameter – a temporary transition spectral width, it

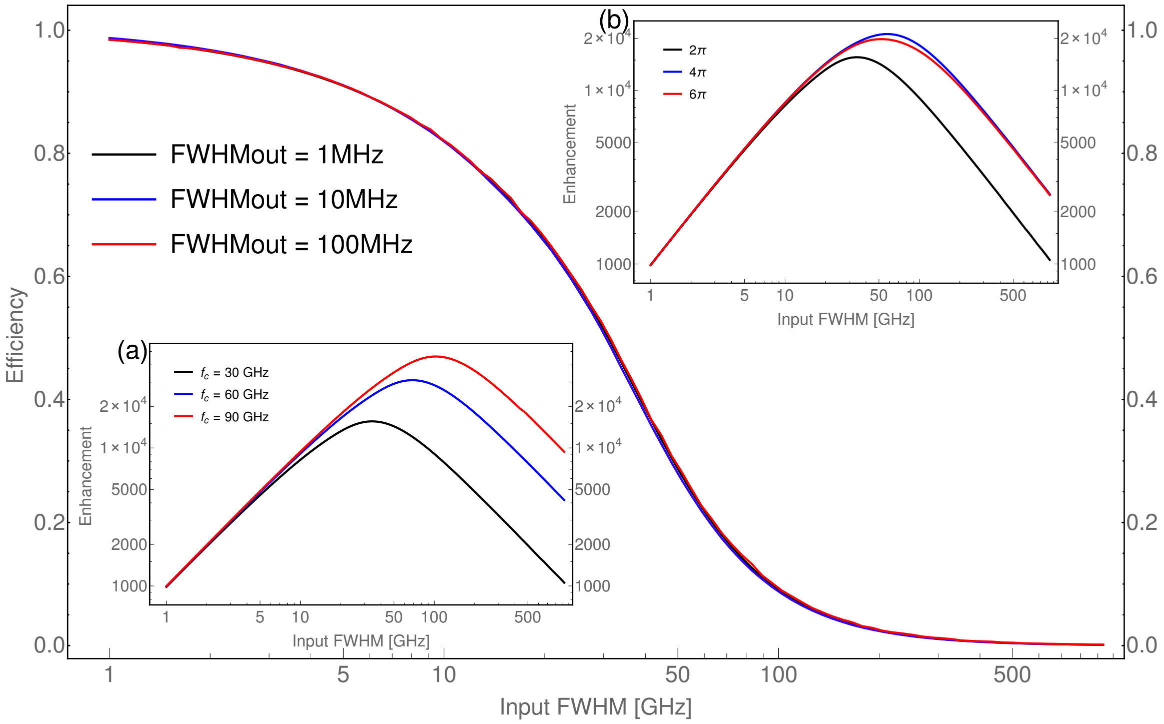

Figure 4.10: Efficiency for different widths of input pulse and fixed output pulse width of 1 MHz, 10 MHz and 100 MHz. Insets: enhancement for different input bandwidths and fixed output bandwidth of 1 MHz (a) for different frequency responses, (b) for different phase modulation depths.

is important to look at behavior of single time lens for different input spectral widths and one fixed output spectral width, because it will tell us about the amount of temporal stretch that can be applied at the first lens in telescope which can be recompressed by the second lens to a narrower spectral pulse. In Fig. 4.10 one can clearly see that at first the bandwidth converter for input FHWMs up to around 50 GHz gives a linearly increasing enhancement (efficiency remains at a high level), however after 50 GHz enhancement starts to decrease in a linear way too.

## 4.4. Summary with dispersion losses

At first it is interesting to look at realistic, achievable at this time lenses and different amount of dispersion with losses at 50% according to section 3.3, both available at this time and in the future. It is trivial that by increasing $\phi$ (increasing dispersion for fixed transmittance or increasing transmittance for fixed chirp) one can exponentially increase the enhancement given by the bandwidth converter for a high compression factor. The second effect which can be seen in Fig. 4.11 is that for a Fresnel lens the maximum enhancement is achieved for higher compression rates, while in the case of a sinusoidal lens lower dispersion losses change only the range for which the enhancement is constant and at around 15. In comparison of both lenses it appears that for small compression factors – up to around 100 – the sinusoidal lens shows better performance than the Fresnel one. However, for larger compression factors the enhancement given by a sinusoidal lens remains at the same level, but enhancement given by a Fresnel lens increases linearly with compression rate. It is a promising result in view of second application of time lens – telescope, which will be investigated in the next chapter. In order to improve the performance of the bandwidth converter it is needed to consider faster

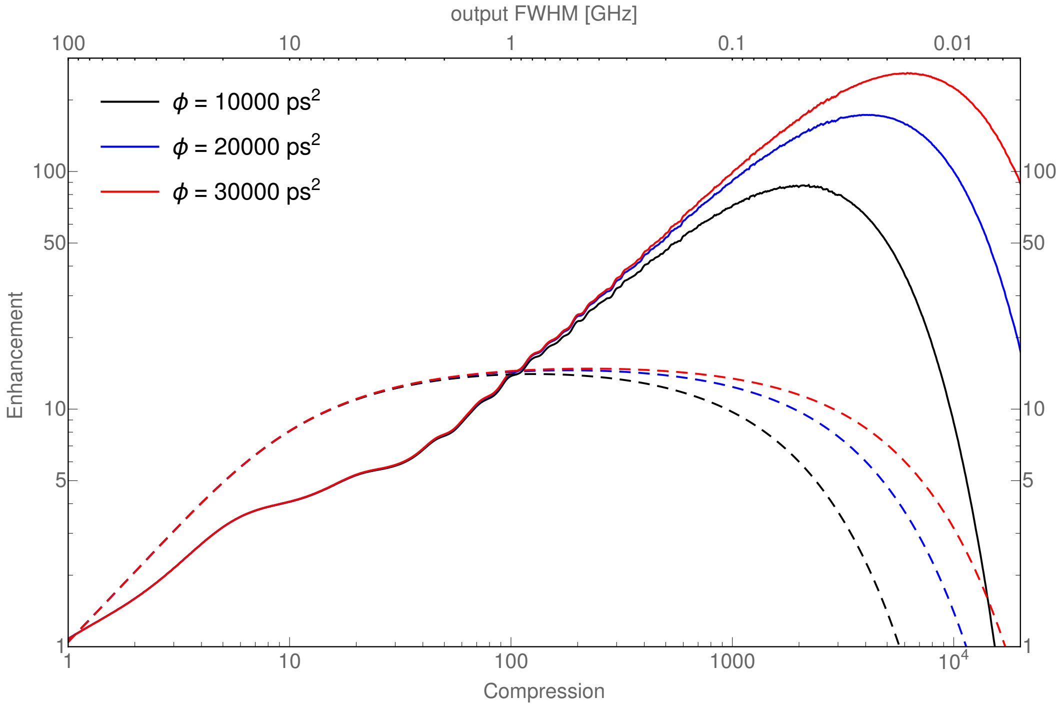

Figure 4.11: Enhancement with a realistic Fresnel (solid) and sinusoidal (dashed) (with $A =$ 25) time lens for different dispersion units. The input pulse width is 100 GHz.

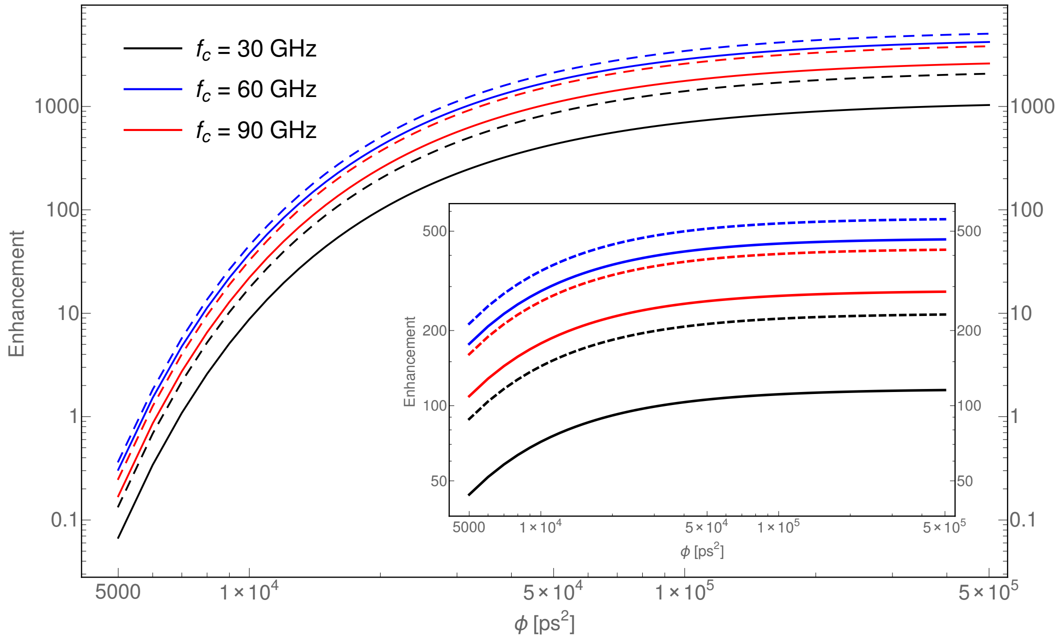

Figure 4.12: Enhancement versus dispersion unit for different frequency response cutoffs for compression rate (main plot) $10^4$ (inset) $10^3$. Width of input pulse is 100 GHz.

AWGs or less lossy dispersion elements. Fig. 4.12 represents an enhancement by using better AWGs or better dispersion or both. It is clear that for lower compression factors, around $10^3$ a significant improvement can be done only by using faster AWGs. However for higher compression factors more important is to minimize losses coming from dispersive elements. Therefore efficient compression from 100 GHz to hundreds of MHz can be performed, if not at this time, then in several years, However in order to compress a 1 GHz width pulse the bandwidth converter needs $10^4$ times more dispersion, therefore more losses which will annihilate any enhancement, so it is not possible to compress a single GHz pulse by using a bandwidth converter.

# Chapter 5

# Temporal telescope

In this section I will investigate if there is a possibility to compress a spectral pulse of several GHz to several MHz in an efficient way by a temporal telescope. Its principle of work is to firstly expand the spectrum of the pulse, so that dispersion can chirp more effectively and the second lens does a massive spectral compression. I will look at two types of telescopes, one consisting of one sinusoidal lens and one Fresnel lens and second one consisting of two Fresnel lenses. Also I will consider the use of elements, whose parameters are expected to become accessible in the near future in order to look for future implementations.

## 5.1. Sinus-Fresnel telescope

The first type of telescope I took into consideration is a combination of a sinusoidal lens, a Fresnel lens and of course a dispersive element between them. Results of a bandwidth converter shows that a sinusoidal lens works very well for small magnifications, which is required from first lens of telescope. The Fresnel lens can however achieve large apertures, what is required for second lens of the temporal telescope. Therefore their combination should exhibit high performance.

The case of the temporal telescope is more complicated due to an additional parameter, which is the transitional spectral width. Therefore I considered only few pairs of input and output widths. Note that I only included losses from dispersion, therefore total loss is higher due to losses e.g. in EOM, but there are insertion losses, which remain constant and around 50%.

The most interesting case would be a compression from 1 GHz to 1 MHz, however a temporal telescope is very sensible for computation precision and this compression cannot be simulated at this moment, due to insufficient number of points. Instead I focused on compression from 5 GHz to 5 MHz, see Fig. 5.1. It is clearly visible, that for realistic at this time implementations there is actually no enhancement. Even for improved electronics maximum gain of several times will be compensated by losses not included in these results. It is caused by losses of dispersion for small transition FWHM and by low efficiency of lenses for high transition FWHMs. There is also another effect – a reduction of needed transition FWHM due to less loss coming from dispersion elements, see inset of Fig. 5.2.

There can be two solutions: using a Fresnel lens instead a sinusoidal lens, which can increase performance for high transition FWHMs, or look for less lossy dispersive elements. The first way will be investigated in next section, the latter however shows (see Fig. 5.2) an increase of enhancement which is nearly linear with increase of dispersion unit $\phi$. In further future it should be possible to achieve enhancement of several dozens for compression from

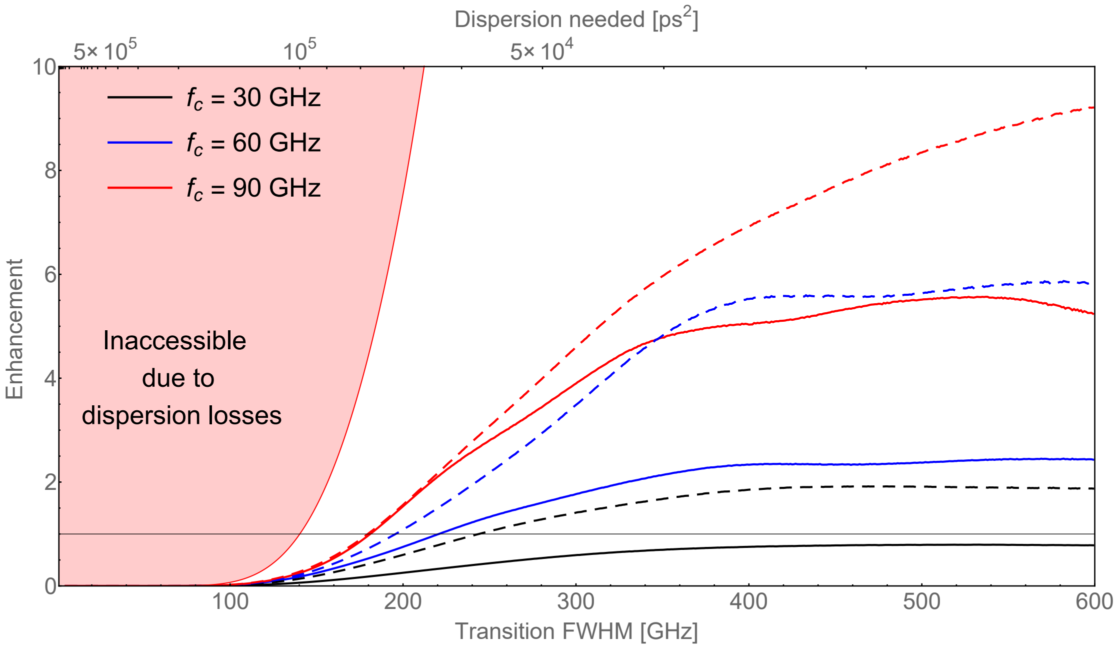

Figure 5.1: Enhancement versus transition FWHM for different frequency response cutoffs of second, Fresnel lens with phase modulation depth (solid) $2\pi$ and (dashed) $4\pi$ for compression from 5 GHz to 5 MHz. Amplitude of sinusoidal lens $A = 25$ and dispersion unit $\phi = 10^4\,\mathrm{ps}^2$.

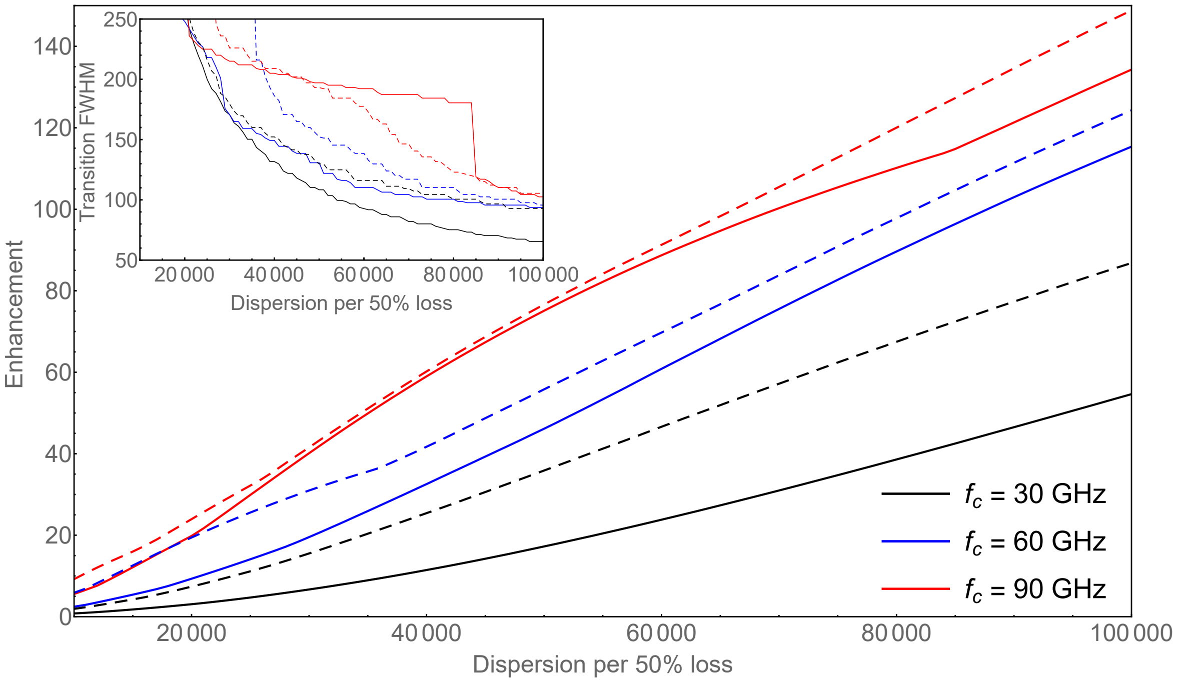

Figure 5.2: Maximal enhancement versus dispersion unit for different frequency response cutoffs of second, Fresnel lens with phase modulation depth (solid) $2\pi$ and (dashed) $4\pi$ for compression from 5 GHz to 5 MHz. Amplitude of sinusoidal lens $A = 25$ and dispersion unit $\phi = 10^4\,\mathrm{ps}^2$. Inset – Transition FWHM for maximal enhancement. The step is caused by a jump from one maximum of enhancement into another, closer one.

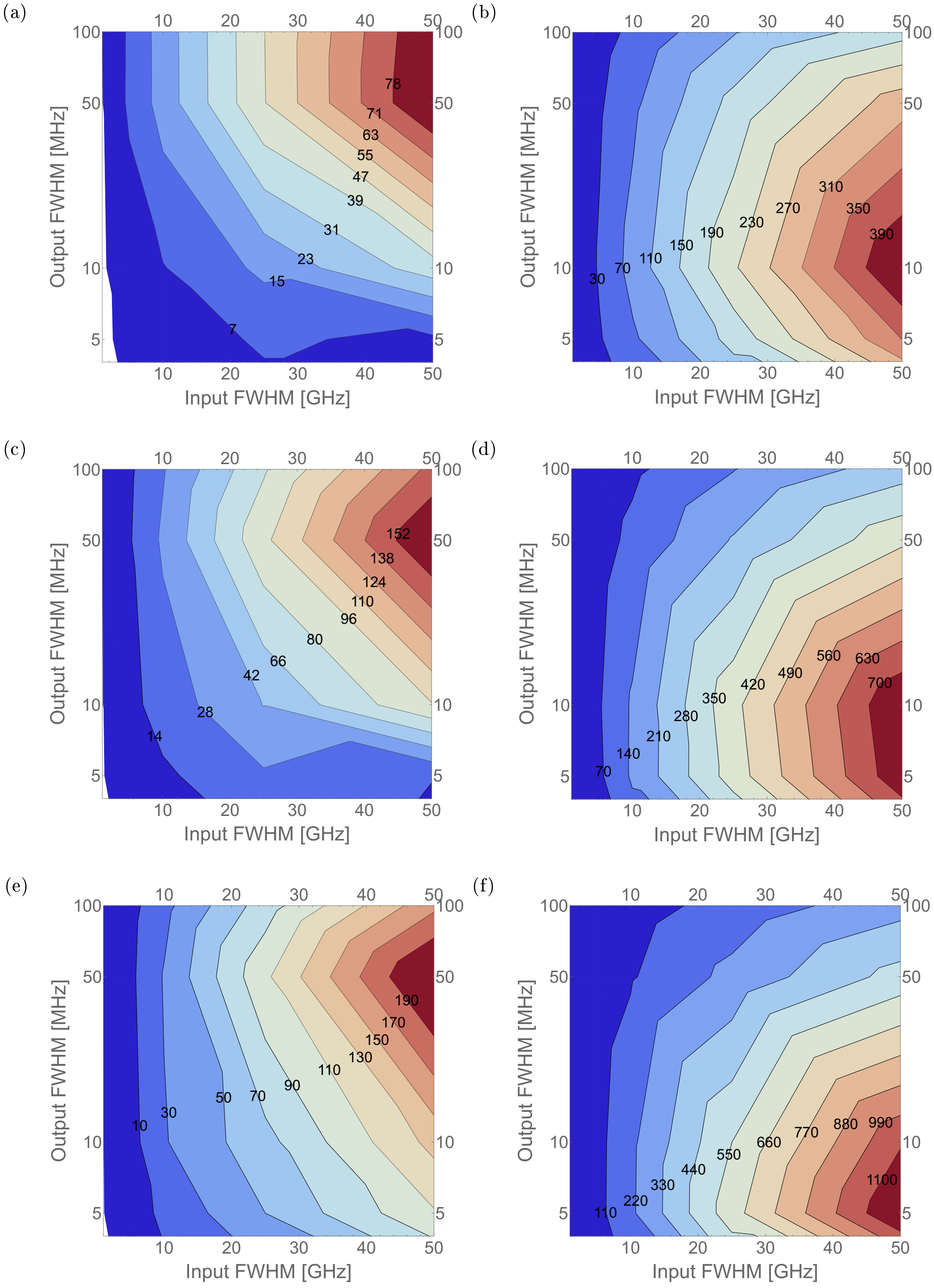

Figure 5.3: Enhancement versus input and output FWHMs for frequency response: (a,b) – 30 GHz, (c-d)60 GHz, (e-f) 90 GHz, and for different dispersion units: (a,c,e) – $10^4$ ps$^2$ and (b,d,f) $5 \times 10^4$ ps$^2$. Phase modulation depth is $4\pi$ and sine amplitude is $A = 25$.

single GHz into single MHz.

In Fig. 5.3 one can see a comparison of enhancement for different frequency responses, dispersion units and pairs of input and output spectral widths. For realistic at this time parameters, shown in subfig. (a), a temporal telescope is useful for compressing a input pulse of tens GHz spectral width into hundreds of MHz one. However in the best case scenario, with frequency response of 90 GHz and dispersion unit $\phi = 5 \times 10^4\,\mathrm{ps}^2$ the temporal telescope is achieving even thousand times better performance in comparison to not using it.

It is interesting to investigate which improvement, of AWGs or of dispersive elements, is more important. One should note first, that achieving higher dispersion per the same loss is more like manufacturing problem than a physical limitation. In case of electronics however, it is well known that it has a physical limitations due to minimal width of electric paths. On the other hand from Fig. 5.1 and Fig. 5.2 one can see that three times better frequency response of AWG gives nearly the same results as three times less lossy dispersion. Therefore after taking into account problems connected with gaining better performace it is more important to achive higher dispersion per the same losses.

Very similiar conclusions can be drawn from comparision of enhancement for different FWHMs, both input and output with different frequency responses and dispersion units.

## 5.2. Fresnel-Fresnel telescope

At first I want to investigate the same pair of input-output FWHMs as in previous section – 5 GHz input and 5 MHz output – in order to look for any performance improvements.

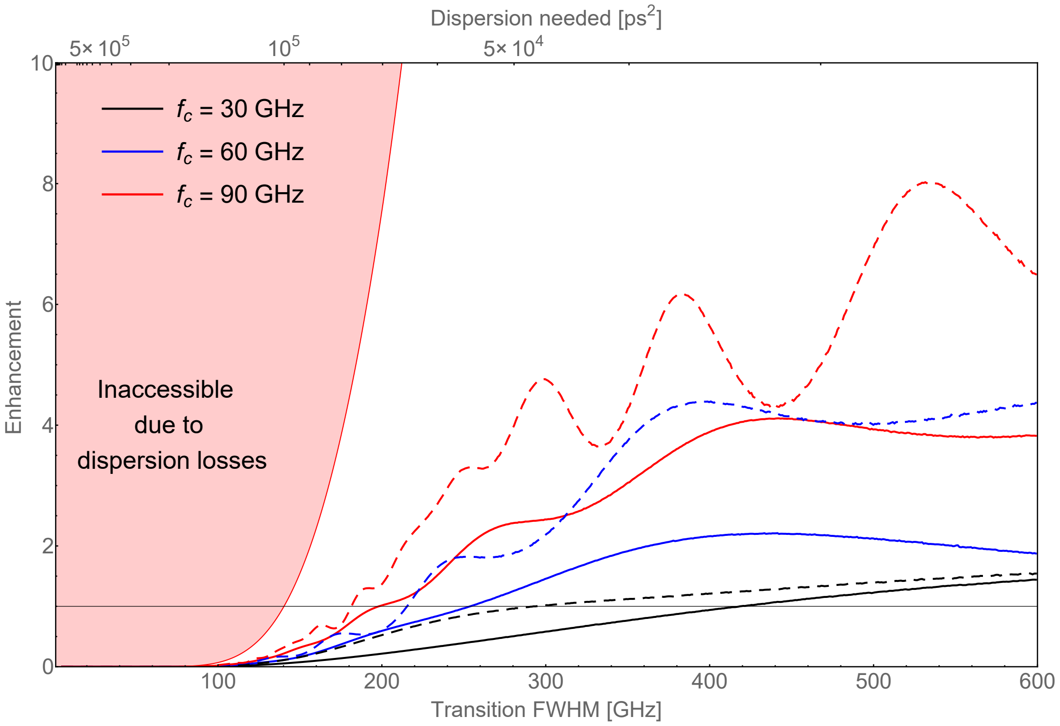

Figure 5.4: Enhancement versus transition FWHM for different frequency response cutoffs of the second, Fresnel lens with phase modulation depth (solid) $2\pi$ and (dashed) $4\pi$ for compression from 5 GHz to 5 MHz. Amplitude of sinusoidal lens $A = 25$ and dispersion per 50% transmittance $\phi = 10^4\,\mathrm{ps}^2$

An enhancement shown in Fig. 5.4 for realistic frequency response of 30 GHz clearly reaches higher values, than in case of sine-Fresnel telescope, because it crosses the line between gain and loss (enhancement = 1). However for future frequency responses this type of telescope shows lower performance, then the previous one. Oscillations of enhancement for higher phase modulation depth are caused by incoming next turning points of first lens into signal envelope ruining performance, but in average any changes in enhancement in reference to sine-Fresnel telescope are minor.

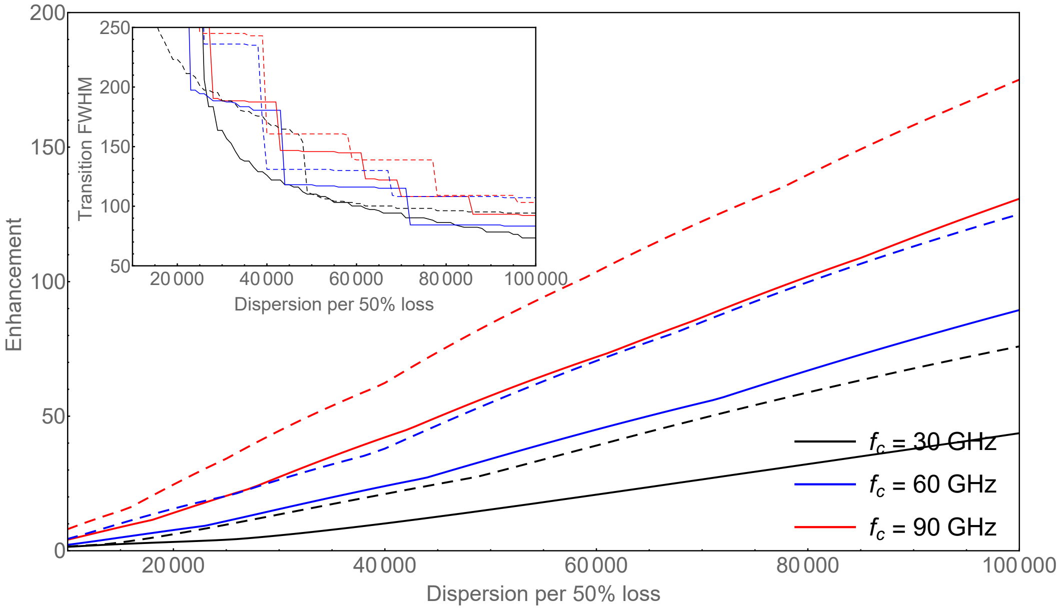

Figure 5.5: Maximal enhancement versus dispersion for 50% loss for different frequency response cutoffs of second, Fresnel lens with phase modulation depth (solid) $2\pi$ and (dashed) $4\pi$ for compression from 5 GHz to 5 MHz. Amplitude of sinusoidal lens $A = 25$ and dispersion per 50% transmittance $\phi = 10^4\,\mathrm{ps}^2$. Inset – Transition FWHM for maximal enhancement. Steps are caused by jumps from one maximum of enhancement into another, closer one.

Maximal enhancement for different dispersion units, Fig. 5.5, shows that behavior of this kind of telescope is almost identical as a previous one, except for oscillation of enhancement. Jumps in transition FWHM for which the best performance is obtained are caused by oscillations, more precise when one maximum of those is gaining better enhancement than the other, usually for transition FWHMs closer to initial FWHM.

Comparison of enhancement versus input and outputs widths for different frequency responses and dispersion units shows very similar behavior as previously.

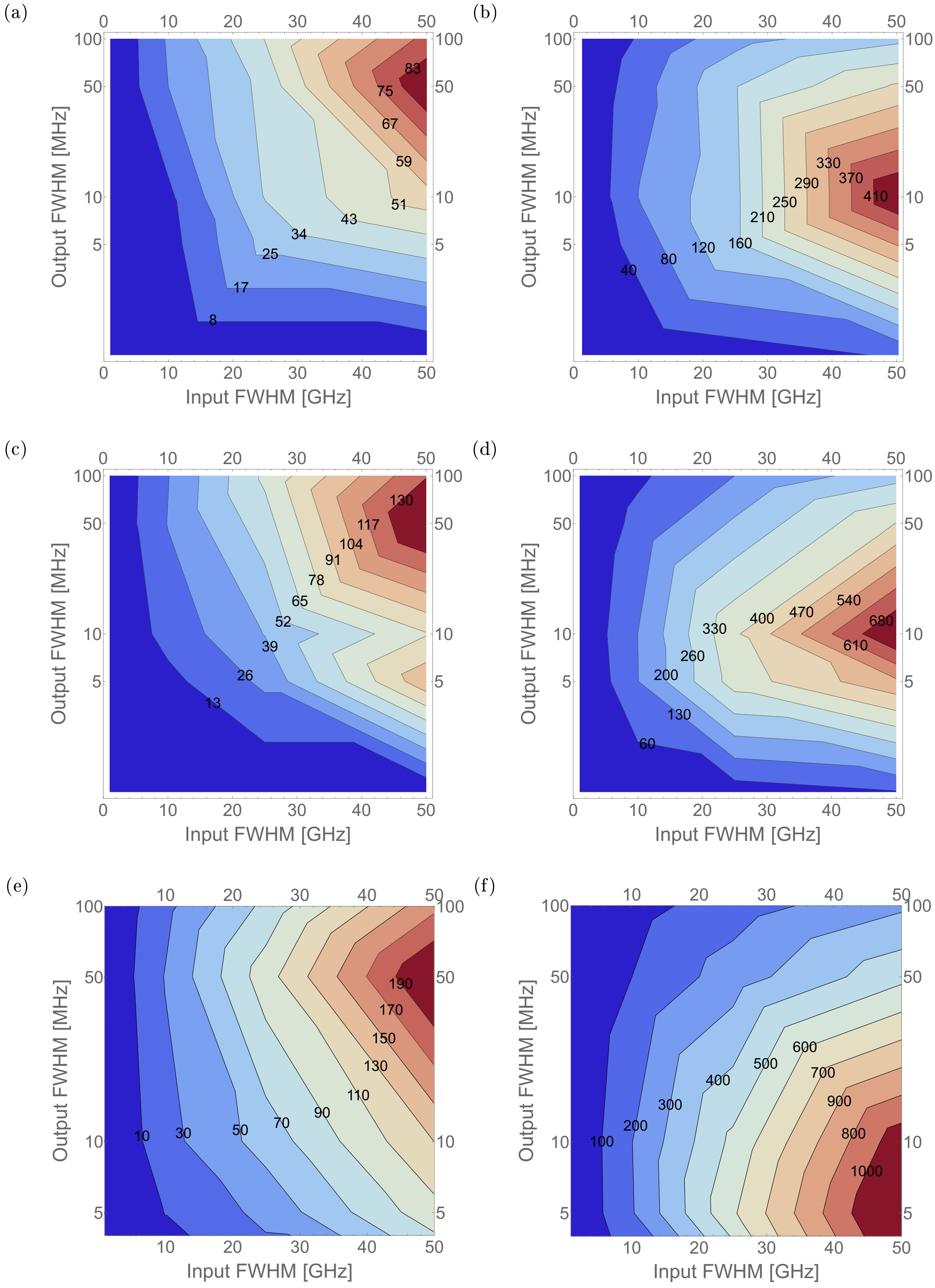

Figure 5.6: Enhancement versus input and output FWHMs for frequency response: (a,b) – 30 GHz, (c-d) 60 GHz, (e-f) 90 GHz, and for different dispersion for 50% loss: (a,c,e) – $10^4\,\mathrm{ps}^2$ and $5 \times 10^4\,\mathrm{ps}^2$. Phase modulation depth is $4\pi$.

# Chapter 6

# Conclusions and outlook

An electrooptic bandwidth converter and the temporal telescope were simulated for different spectral widths of input and output pulses. Both realistic and future implementations were considered.

At first different kinds of temporal lenses were investigated. It was shown that there is no use of ideal parabolic lens due to maximal phase modulation depth of implementation elements such as AWG and EOM.

Then sinusoidal approximation of square function was considered, which results in a constant enhancement (where $T = 1$) in function of the compression factor (or equivalently output pulse width) for fixed width of input pulse, therefore the efficiency of this lens linearly decreases with compression factor. The performance of such lens is independent of the input bandwidth, because a phase added by the lens is scaling the same way as the input width. The only improvement can be done by increasing the phase modulation depth.

The third type of investigated time lens was a Fresnel lens. It shows a high dependence on the frequency response of the phase modulating device (e.g. AWG). The efficiency of such lens is decreasing from 1 to some constant value, above some compression rate, where it remains constant. It means that the enhancement given by this kind of lens is increasing linearly with compression rate. The performance of this lens depends on both input pulse and compression rate. The first is caused by the given constant frequency response, which does not scale with the width of input pulse. It is causing a performance decrease with the increase of input pulse width. The large performance improvement can be obtained by using phase modulation depth of $4\pi$ instead $2\pi$, however it results in efficiency oscillations due to impairments at modulo points of the Fresnel phase. Also an improvement can be achieved by using better AWGs with higher frequency response cutoffs.

The bandwidth converter including losses from dispersive elements was considered. For compression from 100 GHz width pulse the bandwidth converter using a sinusoidal lens shows a constant enhancement of around 15 for wide range of compression rates, from 10 to 1000. On the other hand by using a Fresnel lens one can obtain an enhancement scaling linearly with compression rate up to some maximum value, after which it falls down fastly due to dispersion losses. The maximum for realistic at this time implementations is placed at compression factors of around 2000 with enhancement of around 100. Therefore it is an efficient way to compress a 100 GHz pulses to 100 MHz ones. This results in exciting possibility of connecting long distance communication using (D)WDM with single photon nonlinearities. The maximum moves to the higher compression rates with less lossy dispersive elements allowing for higher compress rates with high efficiency. For high input widths in order to get better results it is more important to improve electronics and their frequency response, but for lower input widths losses from dispersion become dominating, therefore in this region it is more important to minimize the loss in chirping elements. The bandwidth converter is not useful in compressing 1 GHz pulses due to need of very large amount dispersion, introducing extremely high losses.

The temporal telescope was proposed and simulated in order to compress single GHz pulses into single MHz ones. The precision of computations allows to simulate a 5 GHz pulse into 5 MHz. Two types of temporal telescopes were investigated, the first consisting of one sinusoidal and one Fresnel lens and the second consisting of two Fresnel lenses. Both shows very similar performance, therefore the main role in this system in terms of efficiency plays the second lens, which in both cases is the same. It was shown that for realistic at this time parameters its enhancement is around one, therefore there is no improvement now. However implementations in further future, with better AWG and less lossy dispersive elements can obtain the enhancement in order of hundreds. The most important development direction in view of temporal telescope, should be minimizing the loss coming from chirping elements. However the less lossy dispersion is, the more important electronics became. For narrow input pulses the temporal telescope shows better performance than the bandwidth converter.

The next steps in order to investigate the temporal telescope more closely should be certainly simulating noise added to signal by AWG and also simulating non-uniformity of the frequency chirp caused by realistic dispersive elements. After this one should confirm these results by performing an experiment. Simultaneously one can look for different implementations of time lens or upgrading an idea of Fresnel lens e.g. by implementing an genetic algorithm, which will optimize a phase before DAC for different frequency responses.

# Bibliography

[1] H. J. Kimble, “The quantum internet.,” *Nature*, vol. 453, no. 7198, pp. 1023–1030, 2008.

[2] C. Santori, D. Fattal, J. Vucković, G. S. Solomon, and Y. Yamamoto, “Indistinguishable photons from a single-photon device.,” *Nature*, vol. 419, no. 6907, pp. 594–597, 2002.

[3] Y.-M. He, Y. He, Y.-J. Wei, D. Wu, M. Atatüre, C. Schneider, S. Höfling, M. Kamp, C.-Y. Lu, and J.-W. Pan, “On-demand semiconductor single-photon source with near-unity indistinguishability,” *Nature Nanotechnology*, vol. 8, no. 3, pp. 213–217, 2013.

[4] A. Kiraz, S. Fälth, C. Becher, B. Gayral, W. V. Schoenfeld, P. M. Petroff, L. Zhang, E. Hu, C. Becher, B. Gayral, W. V. Schoenfeld, P. M. Petroff, L. Zhang, E. Hu, and A. Imamoğlu, “Photon correlation spectroscopy of a single quantum dot,” *Physical Review B*, vol. 65, no. 16, p. 161303, 2002.

[5] R. B. Patel, A. J. Bennett, K. Cooper, P. Atkinson, C. A. Nicoll, D. A. Ritchie, and A. J. Shields, “Postselective two-photon interference from a continuous nonclassical stream of photons emitted by a quantum dot,” *Physical Review Letters*, vol. 100, no. 20, pp. 1–4, 2008.

[6] R. B. Patel, A. J. Bennett, I. Farrer, C. A. Nicoll, D. A. Ritchie, and A. J. Shields, “Two-photon interference of the emission from electrically tunable remote quantum dots,” *Nature Photonics*, vol. 4, no. 9, pp. 632–635, 2010.

[7] M. T. Rakher, L. Ma, M. Davanço, O. Slattery, X. Tang, and K. Srinivasan, “Simultaneous wavelength translation and amplitude modulation of single photons from a quantum dot,” *Physical Review Letters*, vol. 107, no. 8, pp. 1–5, 2011.

[8] A. J. Shields, “Semiconductor quantum light sources,” *Nature Photonics*, vol. 1, no. 4, pp. 215–223, 2007.

[9] C. Kurtsiefer, S. Mayer, P. Zarda, and H. Weinfurter, “Stable solid-state source of single photons,” *Physical Review Letters*, vol. 85, no. 2, pp. 290–293, 2000.

[10] A. T. Collins, M. F. Thomaz, and M. I. B. Jorge, “Luminescence decay time of the 1.945 eV centre in type Ib diamond,” *Journal of Physics C: Solid State Physics*, vol. 16, no. 11, pp. 2177–2181, 1983.

[11] H. Bernien, L. Childress, L. Robledo, M. Markham, D. Twitchen, and R. Hanson, “Two-photon quantum interference from separate nitrogen vacancy centers in diamond,” *Physical Review Letters*, vol. 108, no. 4, pp. 1–5, 2012.

[12] A. Sipahigil, M. L. Goldman, E. Togan, Y. Chu, M. Markham, D. J. Twitchen, A. S. Zibrov, A. Kubanek, and M. D. Lukin, “Quantum interference of single photons from

remote nitrogen-vacancy centers in diamond," *Physical Review Letters*, vol. 108, no. 14, pp. 1–5, 2012.

[13] J. McKeever, A. Boca, A. D. Boozer, R. Miller, J. R. Buck, A. Kuzmich, and H. J. Kimble, "Deterministic Generation of Single Photons from One Atom Trapped in a Cavity," *Science*, vol. 57, no. March, pp. 1992–1995, 2004.

[14] A. Kiraz, M. Ehrl, T. Hellerer, O. E. Müstecaplioglu, C. Bräuchle, and A. Zumbusch, "Indistinguishable photons from a single molecule," *Physical Review Letters*, vol. 94, no. 22, pp. 1–4, 2005.

[15] R. Lettow, Y. L. A. Rezus, A. Renn, G. Zumofen, E. Ikonen, S. Götzinger, and V. Sandoghdar, "Quantum interference of tunably indistinguishable photons from remote organic molecules," *Physical Review Letters*, vol. 104, no. 12, pp. 26–29, 2010.

[16] K. F. Reim, J. Nunn, V. O. Lorenz, B. J. Sussman, K. C. Lee, N. K. Langford, D. Jaksch, and I. a. Walmsley, "Towards high-speed optical quantum memories," *Nature Photonics*, vol. 4, no. 4, pp. 218–221, 2010.

[17] L. M. Duan, M. D. Lukin, J. I. Cirac, and P. Zoller, "Long-distance quantum communication with atomic ensembles and linear optics.," *Nature*, vol. 414, no. 6862, pp. 413–418, 2001.

[18] M. G. Raymer and K. Srinivasan, "Manipulating the color and shape of single photons," *Physics Today*, vol. 65, no. 11, pp. 32–37, 2012.

[19] M. Karpinski, M. Jachura, L. J. Wright, and B. J. Smith, "Bandwidth manipulation of quantum light by an electro-optic time lens," p. 6, 2016. arXiv:1604.02459.

[20] I. Agha, S. Ates, L. Sapienza, and K. Srinivasan, "Spectral broadening and shaping of nanosecond pulses: toward shaping of single photons from quantum emitters," *Optics Letters*, vol. 39, no. 19, p. 5677, 2014.

[21] N. Matsuda, "Deterministic reshaping of single-photon spectra using cross-phase modulation," *Science Advances*, vol. 2, no. 3, p. e1501223, 2016.

[22] J. Lavoie, J. M. Donohue, L. G. Wright, a. Fedrizzi, and K. J. Resch, "Spectral compression of single photons," *Nature Photonics*, vol. 7, no. 5, pp. 363–366, 2013.

[23] B. H. Kolner, "Space-Time Duality and the Theory of Temporal Imaging," *IEEE Journal of Quantum Electronics*, vol. 30, no. 8, pp. 1951–1963, 1994.

[24] M. D. Eisaman, J. Fan, A. Migdall, and S. V. Polyakov, "Invited Review Article: Single-photon sources and detectors," *Review of Scientific Instruments*, vol. 82, no. 7, p. 071101, 2011.

[25] A. Feizpour, M. Hallaji, G. Dmochowski, and A. M. Steinberg, "Observation of the nonlinear phase shift due to single post-selected photons," *Nature Physics*, vol. 11, no. 11, pp. 905–909, 2015.

[26] D. Tiarks, S. Schmidt, G. Rempe, and S. Durr, "Optical phase shift created with a single-photon pulse," *Science Advances*, vol. 2, no. 4, pp. e1600036–e1600036, 2016.

[27] V. Torres-Company, J. Lancis, and P. Andrés, *Space-time analogies in optics*, vol. 56. 2011.

[28] M. A. Foster, R. Salem, and A. L. Gaeta, "Ultrahigh-Speed Optical Processing Using Space-Time Duality," *Opt. Photon. News*, vol. 22, no. 5, pp. 29–35, 2011.

[29] A. Farsi, C. Joshi, T. Gerrits, R. Mirin, S. W. Nam, V. Verma, S. Ramelow, and A. Gaeta, "Temporal Magnification with Picosecond Resolution at the Single-photon Level," *CLEO: QELS_Fundamental Science 2016*, 2016.

[30] B. Brecht, A. Eckstein, A. Christ, H. Suche, and C. Silberhorn, "From quantum pulse gate to quantum pulse shaper - Engineered frequency conversion in nonlinear optical waveguides," *New Journal of Physics*, vol. 13, pp. 0–22, 2011.

[31] A. H. Gnauck, R. M. Jopson, C. J. McKinstrie, J. C. Centanni, and S. Radic, "Demonstration of low-noise frequency conversion by bragg scattering in a fiber.," *Optics express*, vol. 14, no. 20, pp. 8989–94, 2006.

[32] J. Huang and P. Kumar, "Observation of quantum frequency conversion," *Physical Review Letters*, vol. 68, no. 14, pp. 2153–2156, 1992.

[33] R. Ikuta, Y. Kusaka, T. Kitano, H. Kato, T. Yamamoto, M. Koashi, and N. Imoto, "Wide-band quantum interface for visible-to-telecommunication wavelength conversion.," *Nature communications*, vol. 2, p. 1, 2011.

[34] H. J. McGuinness, M. G. Raymer, C. J. McKinstrie, and S. Radic, "Quantum frequency translation of single-photon states in a photonic crystal fiber," *Physical Review Letters*, vol. 105, no. 9, pp. 1–4, 2010.

[35] S. Ramelow, A. Fedrizzi, A. Poppe, N. K. Langford, and A. Zeilinger, "Polarization-entanglement-conserving frequency conversion of photons," *Physical Review A - Atomic, Molecular, and Optical Physics*, vol. 85, no. 1, pp. 1–5, 2012.

[36] S. Zaske, A. Lenhard, C. A. Keßler, J. Kettler, C. Hepp, C. Arend, R. Albrecht, W. M. Schulz, M. Jetter, P. Michler, and C. Becher, "Visible-to-telecom quantum frequency conversion of light from a single quantum emitter," *Physical Review Letters*, vol. 109, no. 14, pp. 1–5, 2012.

[37] B. Li, M. Li, S. Lou, and J. Azana, "Linear optical pulse compression based on temporal zone plates," *Optics Express*, vol. 21, no. 14, pp. 564–566, 2013.

[38] D. Kielpinski, J. F. Corney, and H. M. Wiseman, "Quantum optical waveform conversion," *Physical Review Letters*, vol. 106, no. 13, pp. 1–4, 2011.

[39] A. M. Weiner, *Ultrafast Optics.* 2004.

[40] B. J. Smith and M. G. Raymer, "Photon wave functions, wave-packet quantization of light, and coherence theory," *New Journal of Physics*, vol. 9, 2007.

[41] NVIDIA, "CUDA C Programming Guide," 2015.

[42] NVIDIA, "CUFFT Library User's Guide," 2015.

[43] NVIDIA, "CUBLAS Library User's Guide," 2015.

[44] B. Li, S. Lou, and J. Azana, "Novel temporal zone plate designs with improved energy efficiency and noise performance," *Journal of Lightwave Technology*, vol. 32, no. 24, pp. 4201–4207, 2014.

# Appendix A: simulation code

```
/*

        STANDARD FREQUENCY UNIT -- GHz
        STANDARD TIME UNIT -- ns

*/


#include <cublas_v2.h>
#include <cuda.h>
#include <cufft.h>
#include <cmath>
#include <fstream>
#include <iomanip>
#include <iostream>
#include <sstream>
#include <stdio.h>
#include <string>
#include <sys/time.h>
#include <vector>

using namespace std;


/*##############################     ERROR HANDLING     ##############################*/

#define CUDA_ERROR_CHECK

#define CudaSafeCall( err ) __cudaSafeCall( err, __FILE__, __LINE__ )
#define CudaCheckError()    __cudaCheckError( __FILE__, __LINE__ )

inline void __cudaSafeCall( cudaError err, const char *file, const int line )
{
#ifdef CUDA_ERROR_CHECK
    if ( cudaSuccess != err )
    {
        fprintf( stderr, "cudaSafeCall()␣failed␣at␣%s:%i␣:␣%s\n",
                 file, line, cudaGetErrorString( err ) );
        exit( -1 );
    }
#endif

    return;
}

inline void __cudaCheckError( const char *file, const int line )
{
#ifdef CUDA_ERROR_CHECK
    cudaError err = cudaGetLastError();
    if ( cudaSuccess != err )
    {
        fprintf( stderr, "cudaCheckError()␣failed␣at␣%s:%i␣:␣%s\n",
                 file, line, cudaGetErrorString( err ) );
        exit( -1 );
    }
    // More careful checking. However, this will affect performance.
    // Comment away if needed.
    err = cudaDeviceSynchronize();
    if( cudaSuccess != err )
    {
        fprintf( stderr, "cudaCheckError()␣with␣sync␣failed␣at␣%s:%i␣:␣%s\n",
                 file, line, cudaGetErrorString( err ) );
        exit( -1 );
    }
#endif

    return;
}




/*#################################################################################*/


void memory_management();
void load_freqency_response(const char* nazwapliku);
double double_reconstruction( cufftDoubleComplex x );



void FWHM_filter_match(
        double& match, double& maximum, double FWHMfilter,
        int type1, int type2, double FWHM, double GDD,
        double K1, double cutoff1, double amplification1,
        double K2, double cutoff2, double amplification2,
        int resolution, double param1, double param2);

/* LENS */

void lens_simulation(
        string filename, int type, double FWHM, double GDD, double K,
        double cutoff, double amplification, int resolution = 8, double param = 0);
void lens_simulation_time(
        int type, double FWHM,  double GDD, double K, double cutoff,
        double amplification, int resolution = 8, double param = 0);
void lens_simulation_freq(
        int type, double FWHM,  double GDD, double K, double cutoff,
        double amplification, int resolution = 8, double param = 0);


/* TELESCOPE */

void telescope_simulation(
        string filename, int type1, int type2, double FWHM,  double GDD,
        double K1, double cutoff1, double amplification1,
        double K2, double cutoff2, double amplification2,
        int resolution, double param1, double param2 );
void telescope_simulation_time(int type1, int type2, double FWHM,  double GDD,
        double K1, double cutoff1, double amplification1,
        double K2, double cutoff2, double amplification2,
        int resolution, double param1, double param2 );
void telescope_simulation_freq(int type1, int type2, double FWHM,  double GDD,
        double K1, double cutoff1, double amplification1,
        double K2, double cutoff2, double amplification2,
        int resolution, double param1, double param2 );


/* COMPUTATIONS */

void gauss_generation(double FWHM);
void chirping(double GDD);
void normalize(double norm);


/* PHASE GENERATION */

void prepare_phase(
        cufftDoubleComplex* phase, int type, double K, double cutoff,
        double amplification, int resolution = 8, double param = 0);
void put_prepared_phase(cufftDoubleComplex* phase);
void frequency_response(double cutoff);
void phase_amplification(double cutoff, double amplification);
void phase_resolution( int bits );


/* WRITE TO FILE */

void write_to_file( int type, const char * filename , cufftDoubleComplex* data);


/* KERNELS */

__global__ void gaussian_wavefunction_kernel(int Nk, double A, double B, double zero,
        double sigma, cufftDoubleComplex *vector);

__global__ void put_linear_phase_kernel(int Nk, double A, double B, double zero,
        double K, cufftDoubleComplex *vector);
__global__ void put_quadratic_phase_kernel(int Nk, double A, double B, double zero,
        double K, double C, double D, cufftDoubleComplex *vector);
__global__ void put_phase_kernel(int Nk, cufftDoubleComplex *phase,
        cufftDoubleComplex *vector);
__global__ void phase_stepping(int Nk, double step_size,
        cufftDoubleComplex* vector);
__global__ void FWHM_filter(int Nk, double A, double B, double zero, double FWHM,
        double* vector);

__global__ void cufft_multiply_by_scalar(int Nk, double scalar,
        cufftDoubleComplex* vec);
__global__ void cufft_multiply_by_vector(int Nk, double* vecD,
        cufftDoubleComplex* vecC);
__global__ void cufft_devide_by_vector(int Nk, double* vecD,
        cufftDoubleComplex* vecC);
__global__ void cufft_modulus_squared_to_double(int Nk, double* vecD,
        cufftDoubleComplex* vecC);
__global__ void cufft_to_double(int Nk, double* vecD, double shift,
        cufftDoubleComplex* vecC);

__global__ void phase_quadratic(int Nk, int i, double range, double K,
        double aperture, cufftDoubleComplex *phase);
__global__ void phase_fresnel(int Nk, int i, double range, double K, int factor,
        double aperture, cufftDoubleComplex *phase);
__global__ void phase_sinus(int Nk, int i, double range, double amplitude,
        double frequency, cufftDoubleComplex *phase);
__global__ void phase_gabor(int Nk, int i, double range, double amplitude,
        double frequency, cufftDoubleComplex *phase);




#define cuPI 3.14159265358979323846264433
#define cuPIshift 0

#define N (72*1024*1024)

#define fA (-500) // computational frequency range (fA,fB) (GHz)
#define fB ( 500)

const bool DESCRIPTIONS = 0;

double F_RANGE = 0.05; // (-F_RANGE, F_RANGE) - frequency range to be written to file
```

```
    (GHz)
double T_RANGE = 3; // (-T_RANGE, T_RANGE) - time range to be written to file (ns)

int F_STEP = 1; // frequency step of writting to file (points)
int T_STEP = 1; // time step of writting to file (points)

const int PRECISION = 10; // number of significant digits when writting to file

const int VECNUM = 8; // number of computational vectors (has to be divisible by 2)
const int VECLEN = N/VECNUM; // length of single computational vector

const int THREADS = 1024; // number of threads per block (max = 1024)
const int BLOCKS = VECLEN/THREADS; // number of block

const int N_RESP = 3465889; // number of points in file with frequency response
double F_AMP = 30; // frequency cutoff of amplification (GHz)
int N_AMP = F_AMP/(fB-fA)*N; // number of points in amplification (has to be less than
     VECLEN)

int PI_FACTOR = 1; // 1 = modulo 2pi, 2 = modulo 4pi etc.
double APERTURE_BEFORE = 0; // time aperture before AWG
double APERTURE_AFTER = 0;// time aperture after AWG


cufftDoubleComplex* data_H;
cufftDoubleComplex* phase1_H;
cufftDoubleComplex* phase2_H;

double* response_D;
double* dataD_D;
cufftDoubleComplex* data_D;
cufftDoubleComplex* dataN_D;

cufftHandle cufftplan;
cublasHandle_t cublashandle;


/* ----------------------------------- MAIN ----------------------------------------*/

int main()
{
        struct timeval tic,toc;
        gettimeofday(&tic,0);

        cufftPlan1d(&cufftplan, N, CUFFT_Z2Z, 1);
        cublasCreate(&cublashandle);

        memory_management();
        load_freqency_response("fr_res.txt");

        cout << "Frequency␣resolution:␣" << ((fB-fA)/(double)N) << "GHz" << endl;
        cout << "Time␣respolution:␣" << ( 1000.0/(fB-fA) ) << "ps"<< endl << endl;

        data_H = new cufftDoubleComplex[N];
        phase1_H = new cufftDoubleComplex[N];
        phase2_H = new cufftDoubleComplex[N];
        CudaSafeCall( cudaMalloc( (void**) &data_D, VECLEN*sizeof(cufftDoubleComplex))
             );
        CudaSafeCall( cudaMalloc( (void**) &dataD_D, VECLEN*sizeof(double)) );
        CudaSafeCall( cudaMalloc( (void**) &dataN_D, N*sizeof(cufftDoubleComplex)) );


// ====================================================================================


// ====================================================================================

        gettimeofday(&toc,0);
        double t = toc.tv_sec - tic.tv_sec + (toc.tv_usec - tic.tv_usec)/1000000.0;
        cout << endl << "Czas␣wykonania␣programu:␣" << t << endl;

        cublasDestroy(cublashandle);
        cufftDestroy(cufftplan);

        delete data_H;
        delete phase1_H;
        delete phase2_H;
        CudaSafeCall( cudaFree(data_D) );
        CudaSafeCall( cudaFree(dataN_D) );
        CudaSafeCall( cudaFree(dataD_D) );
        CudaSafeCall( cudaFree(response_D) );

        return 0;
}
/* ---------------------------------------------------------------------------------*/


/*####################        ADDITIONAL SIMULATIONS      ##########################*/

void FWHM_filter_match(
        double& match, double& maximum, double FWHMfilter,
        int type1, int type2, double FWHM, double GDD,
        double K1, double cutoff1, double amplification1,
        double K2, double cutoff2, double amplification2,
        int resolution, double param1, double param2)
{

        if(type2 < 0 ) lens_simulation_freq( type1, FWHM, GDD,
                                                K1, cutoff1, amplification1,
                                                    resolution, param1);
        else telescope_simulation_freq( type1, type2, FWHM, GDD,
                        K1, cutoff1, amplification1,
                        K2, cutoff2, amplification2, resolution, param1, param2);

        match = 0.0;
        double temp = 0.0;
        int maxindex;

        cufft_modulus_squared_to_double<<<BLOCKS,THREADS>>>(VECLEN, dataD_D,
                &dataN_D[N/2 - VECLEN/2]);
        CudaCheckError();

        cublasIdamax(cublashandle, VECLEN, dataD_D, 1, &maxindex);

        CudaSafeCall( cudaMemcpy( &maximum, &dataD_D[maxindex], sizeof(double),
        cudaMemcpyDeviceToHost ) );

        int index1 = (N-VECLEN)/2.0 + maxindex - (double)N*FWHMfilter/(fB-fA)/2.0 ;
        int index2 = (N-VECLEN)/2.0 + maxindex + (double)N*FWHMfilter/(fB-fA)/2.0 ;

        double maxpos = (fA+fB)/2.0 + (maxindex-VECLEN/2.0)*(double)(fB-fA)/(double)N;

        for(int i = 0; i < VECNUM; i++)
        {
                double temp1 = 0;

                if( !( index1 > (i+1)*VECLEN) && !(index2 < i*VECLEN) )
                {
                        cufft_modulus_squared_to_double<<<BLOCKS,THREADS>>>(VECLEN,
                            dataD_D,
                                &dataN_D[i*VECLEN]);
                        CudaCheckError();
                        temp1 = 0.0;
                        double range = (fB-fA)/(double)VECNUM;
                        FWHM_filter<<<BLOCKS,THREADS>>>(VECLEN, fA + i*range, fA + (i
                            +1)*range,
                                maxpos, FWHMfilter, dataD_D);
                        CudaCheckError();
                        cublasDasum(cublashandle, VECLEN, dataD_D, 1, &temp1);
                        match += temp1;

                        temp1 = 0.0;
                        gaussian_wavefunction_kernel<<<BLOCKS,THREADS>>>(VECLEN, fA+i*
                            range,
                                fA+(i+1)*range, maxpos, FWHMfilter/2.0/sqrt(log(2.0)),
                                     data_D);
                        CudaCheckError();
                        cufft_modulus_squared_to_double<<<BLOCKS,THREADS>>>(VECLEN,
                            dataD_D,
                                data_D);
                        CudaCheckError();
                        FWHM_filter<<<BLOCKS,THREADS>>>(VECLEN, fA + i*range, fA + (i
                            +1)*range,
                                maxpos, FWHMfilter, dataD_D);
                        CudaCheckError();
                        cublasDasum(cublashandle, VECLEN, dataD_D, 1, &temp1);
                        temp += temp1;

                }
        }
        match /= (double)temp;
}


/*#####################       ADDITIONAL HOST FUNCTIONS      ########################*/


void memory_management()
{
        size_t free;
        size_t total;

        cudaMemGetInfo(&free,&total);

        double total_memory = (double)total/1024/1024/1024;
        double free_memory = (double)free/1024/1024/1024;
        cout << "Calkowita␣pamiec␣GPU:␣" << total_memory << "GB" << endl;
        cout << "Wolna␣pamiec␣GPU:␣" << free_memory << "GB" << endl;
        cout << "Pamiec,␣ktora␣bedzie␣zaalokowana:␣";
        cout << ((double)((N+VECLEN)*sizeof(cufftDoubleComplex)+
                (VECLEN+N_RESP)*sizeof(double))/1024/1024/1024) << "GB" << endl;
        cout << endl;
}



void load_freqency_response(const char* nazwapliku)
{
        fstream plikin;
        plikin.open(nazwapliku, fstream::in);

        double* response_H = new double[N_RESP];
        for(int i = 0; i < N_RESP; i++)
        {
                double temp;
                plikin >> temp;
                response_H[i] = temp;
        }
        plikin.close();

        CudaSafeCall( cudaMalloc( (void**) &response_D, N_RESP*sizeof(double)) );
        CudaSafeCall( cudaMemcpy( response_D, response_H, N_RESP*sizeof(double),
                cudaMemcpyHostToDevice) );

        delete response_H;
}
```

```
double double_reconstruction( cufftDoubleComplex x )
{
        double modulus = sqrt( pow(x.x,2.0) + pow(x.y,2.0) );
        double phase = atan2( x.y, x.x);
        if(phase > cuPI/2.0) modulus = -modulus;
        return modulus;
}



/*##########################        LENS        ####################################*/


void lens_simulation(string filename, int type, double FWHM, double GDD, double K,
        double cutoff, double amplification, int resolution, double param)
{
        stringstream ss;

        lens_simulation_time( type, FWHM, GDD, K, cutoff, amplification, resolution,
                param);

        ss << filename << "_czas.txt";
        CudaSafeCall( cudaMemcpy( data_H, dataN_D, N*sizeof(cufftDoubleComplex),
                cudaMemcpyDeviceToHost) );
        write_to_file(0, ss.str().c_str(), data_H);
        ss.str("");

        //FFT
        cufftExecZ2Z(cufftplan, dataN_D, dataN_D, CUFFT_INVERSE);

        //normalization to unitary FFT
        normalize(sqrt(N));

        ss << filename << "_czestotliwosc.txt";
        CudaSafeCall( cudaMemcpy( data_H, dataN_D, N*sizeof(cufftDoubleComplex),
                cudaMemcpyDeviceToHost) );
        write_to_file(1, ss.str().c_str(), data_H);
        ss.str("");

        ss << filename << "_faza.txt";
        write_to_file(2, ss.str().c_str(), phase1_H);
        ss.str("");
}


void lens_simulation_time(int type, double FWHM,  double GDD, double K, double cutoff,
        double amplification, int resolution, double param )
{

        prepare_phase(phase1_H, type, K, cutoff, amplification, resolution, param);

        gauss_generation(FWHM);

        chirping(GDD);

        //FFT
        cufftExecZ2Z(cufftplan, dataN_D, dataN_D, CUFFT_INVERSE);

        normalize(sqrt(N));

        put_prepared_phase(phase1_H);

}


void lens_simulation_freq(int type, double FWHM,  double GDD, double K, double cutoff,
        double amplification, int resolution, double param )
{
        lens_simulation_time( type, FWHM, GDD, K, cutoff, amplification, resolution,
                param);

        cufftExecZ2Z(cufftplan, dataN_D, dataN_D, CUFFT_FORWARD);

        normalize(sqrt(N));
}



/*#########################        TELESCOPE        ##################################*/


void telescope_simulation(
        string filename, int type1, int type2, double FWHM,  double GDD,
        double K1, double cutoff1, double amplification1,
        double K2, double cutoff2, double amplification2,
        int resolution, double param1, double param2 )
{
        stringstream ss;

        telescope_simulation_time( type1, type2, FWHM, GDD, K1, cutoff1,
            amplification1,
                K2, cutoff2, amplification2, resolution, param1, param2);

        ss << filename << "_czas.txt";
        CudaSafeCall( cudaMemcpy( data_H, dataN_D, N*sizeof(cufftDoubleComplex),
                cudaMemcpyDeviceToHost) );
        write_to_file(0, ss.str().c_str(), data_H);
        ss.str("");

        cufftExecZ2Z(cufftplan, dataN_D, dataN_D, CUFFT_INVERSE);

        normalize(sqrt(N));

        ss << filename << "_czestotliwosc.txt";
        CudaSafeCall( cudaMemcpy( data_H, dataN_D, N*sizeof(cufftDoubleComplex),
                cudaMemcpyDeviceToHost) );
        write_to_file(1, ss.str().c_str(), data_H);
        ss.str("");

        ss << filename << "_faza1.txt";
        write_to_file(2, ss.str().c_str(), phase1_H);
        ss.str("");

        ss << filename << "_faza2.txt";
        write_to_file(2, ss.str().c_str(), phase2_H);
        ss.str("");
}


void telescope_simulation_time(
        int type1, int type2, double FWHM,  double GDD,
        double K1, double cutoff1, double amplification1,
        double K2, double cutoff2, double amplification2,
        int resolution, double param1, double param2 )
{
        prepare_phase(phase1_H, type1, K1, cutoff1, amplification1, resolution, param1
            );
        prepare_phase(phase2_H, type2, K2, cutoff2, amplification2, resolution, param2
            );

        gauss_generation(FWHM);

        //IFFT
        cufftExecZ2Z(cufftplan, dataN_D, dataN_D, CUFFT_INVERSE);
        put_prepared_phase(phase1_H);

        //FFT
        cufftExecZ2Z(cufftplan, dataN_D, dataN_D, CUFFT_FORWARD);

        chirping(GDD);

        //IFFT
        cufftExecZ2Z(cufftplan, dataN_D, dataN_D, CUFFT_INVERSE);

        put_prepared_phase(phase2_H);

        normalize(pow(N,1.5));
}


void telescope_simulation_freq(
        int type1, int type2, double FWHM, double GDD,
        double K1, double cutoff1, double amplification1,
        double K2, double cutoff2, double amplification2,
        int resolution, double param1, double param2 )
{
        telescope_simulation_time( type1, type2, FWHM, GDD,
                K1, cutoff1, amplification1, K2, cutoff2, amplification2,
                resolution, param1, param2);

        cufftExecZ2Z(cufftplan, dataN_D, dataN_D, CUFFT_FORWARD);

        normalize(sqrt(N));
}



/*#######################        FUNCTIONS        #################################*/


void gauss_generation(double FWHM)
{
        if(DESCRIPTIONS) cout << "Generation␣of␣Gaussian␣wavefunction␣with␣FWHM:␣" <<
            FWHM << endl;

        double range = (fB-fA)/(double)VECNUM;
        double sigma = FWHM/2.0/sqrt(log(2.0));

        for(int i = 0; i < VECNUM; i++)
        {
                gaussian_wavefunction_kernel<<<BLOCKS,THREADS>>>(VECLEN, fA+i*range,
                        fA+(i+1)*range, 0, sigma, &dataN_D[i*VECLEN]);
                CudaCheckError();
        }
}


void chirping(double GDD)
{
        if(DESCRIPTIONS) cout << "Chirping" << endl;

        double range = (double)(fB-fA)/VECNUM;

        for(int i = 0; i < VECNUM; i++)
        {
                put_quadratic_phase_kernel<<<BLOCKS, THREADS>>>(VECLEN, fA+i*range,
                        fA+(i+1)*range, 0, pow(2.0*cuPI, 2.0)*GDD/2.0, 0, 0, &dataN_D[
                            i*VECLEN]);
                CudaCheckError();
        }
}
```

```
void normalize(double norm)
{
        if(DESCRIPTIONS) cout << "Normalization" << endl;

        for(int i = 0; i < VECNUM; i++)
        {
                cufft_multiply_by_scalar<<<BLOCKS,THREADS>>>(VECLEN, 1.0/norm,
                        &dataN_D[i*VECLEN]);
                CudaCheckError();
        }
}


void prepare_phase(cufftDoubleComplex* phase, int type, double K, double cutoff,
        double amplification, int resolution, double param)
{
        if(DESCRIPTIONS) cout << "Preparing␣phase" << endl;

        double range = (double) N/ (double) VECNUM /(fB-fA);

        for(int i = 0; i < VECNUM/2; i++)
        {
                if(type == 0) phase_quadratic<<<BLOCKS,THREADS>>>(VECLEN, i, range, K
                    /2.0,
                        APERTURE_BEFORE, dataN_D);
                if(type == 1) phase_fresnel<<<BLOCKS,THREADS>>>(VECLEN, i, range, K
                    /2.0,
                        PI_FACTOR, APERTURE_BEFORE, dataN_D);
                if(type == 2) phase_sinus<<<BLOCKS,THREADS>>>(VECLEN, i, range, param,
                     K,
                        dataN_D); // K = frequency, param = amplitude
                CudaCheckError();
        }

        if(amplification != 0 ) phase_amplification(cutoff, amplification);

        if(resolution > 0) phase_resolution(resolution);

        if(cutoff != 0) frequency_response(cutoff);

        if(APERTURE_AFTER > 0)
        {
                int aperture = APERTURE_AFTER*(fB-fA);
                CudaSafeCall( cudaMemset( dataN_D, 0,
                        (N/2 - aperture)*sizeof(cufftDoubleComplex)) );
                CudaSafeCall( cudaMemset( &dataN_D[N/2 + aperture - 1], 0,
                        (N/2 - aperture)*sizeof(cufftDoubleComplex)) );
        }

        CudaSafeCall( cudaMemcpy( phase, dataN_D, N*sizeof(cufftDoubleComplex),
                cudaMemcpyDeviceToHost) );
}


void put_prepared_phase(cufftDoubleComplex* phase)
{
        if(DESCRIPTIONS) cout << "Applying␣prepared␣phase" << endl;

        if( APERTURE_AFTER > 0 )
        {
                int aperture = APERTURE_AFTER*(fB-fA);
                int k = floor((double)aperture/(double)VECLEN);
                for(int i = 0; i <= k; i++)
                {
                        if( i == k )
                        {
                                CudaSafeCall( cudaMemcpy(data_D, &phase[N/2 - aperture
                                     + k*VECLEN],
                                        (aperture - k*VECLEN)*sizeof(
                                            cufftDoubleComplex),
                                        cudaMemcpyHostToDevice) );
                                put_phase_kernel<<<BLOCKS,THREADS>>>(VECLEN, data_D,
                                        &dataN_D[N/2 - aperture + k*VECLEN]);
                                CudaCheckError();
                        }
                        else
                        {
                                CudaSafeCall( cudaMemcpy(data_D, &phase[N/2 - aperture
                                     + i*VECLEN],
                                        VECLEN*sizeof(cufftDoubleComplex),
                                            cudaMemcpyHostToDevice) );
                                put_phase_kernel<<<BLOCKS,THREADS>>>(VECLEN, data_D,
                                        &dataN_D[N/2 - aperture + i*VECLEN]);
                                CudaCheckError();
                        }
                }
        }
        else
        {
                for(int i = 0; i < VECNUM; i++)
                {
                        CudaSafeCall( cudaMemcpy(data_D, &phase[i*VECLEN],
                                VECLEN*sizeof(cufftDoubleComplex),
                                    cudaMemcpyHostToDevice) );
                        put_phase_kernel<<<BLOCKS,THREADS>>>(VECLEN, data_D, &dataN_D[
                            i*VECLEN]);
                        CudaCheckError();
                }
        }
}


void frequency_response(double cutoff)
{
        if( cutoff > 0 )
        {
                int index = floor(cutoff/(fB-fA)*(double)N);

                cufftExecZ2Z(cufftplan, dataN_D, dataN_D, CUFFT_FORWARD);
                CudaSafeCall( cudaMemset( &dataN_D[index], 0,
                        2*(N/2-index)*sizeof(cufftDoubleComplex)) );
                cufftExecZ2Z(cufftplan, dataN_D, dataN_D, CUFFT_INVERSE);
                normalize( (double) N );
        }
        else if( cutoff < 0 )
        {
                cufftExecZ2Z(cufftplan, dataN_D, dataN_D, CUFFT_FORWARD);
                CudaSafeCall( cudaMemset( &dataN_D[N_RESP], 0,
                        (N-2*N_RESP)*sizeof(cufftDoubleComplex)) );
                cufft_multiply_by_vector<<<(N_RESP+1023)/1024,1024>>>(N_RESP,
                    response_D,
                        dataN_D);
                CudaCheckError();
                cufftExecZ2Z(cufftplan, dataN_D, dataN_D, CUFFT_INVERSE);
                normalize( (double) N );
        }

}


void phase_amplification(double cutoff, double amplification)
{

        if (amplification < 0 && N_AMP > 0)
        {
                if (DESCRIPTIONS) cout << "Amplifikation␣with␣inverse␣of␣freq␣resp" <<
```

```
                     endl;
                cufftExecZ2Z(cufftplan, dataN_D, dataN_D, CUFFT_FORWARD);
                cufft_devide_by_vector<<<(N_AMP+1023)/1024,1024>>>(N_AMP, response_D,
                        dataN_D);
                cufftExecZ2Z(cufftplan, dataN_D, dataN_D, CUFFT_INVERSE);
                normalize( (double) N );
                CudaCheckError();
        }

        if (amplification != 0)  normalize( 1.0/abs(amplification) );

}


void phase_resolution( int bits )
{
        double maximum=1000, minimum=1000;

        for(int i = 0; i < VECNUM; i++)
        {
                int maxindex, minindex;
                double tempD;

                cufft_to_double<<<BLOCKS,THREADS>>>(VECLEN, dataD_D, 1000,
                        &dataN_D[i*VECLEN]);
                CudaCheckError();

                cublasIdamax(cublashandle, VECLEN, dataD_D, 1, &maxindex);
                cublasIdamin(cublashandle, VECLEN, dataD_D, 1, &minindex);

                cudaMemcpy( &tempD, &dataD_D[maxindex-1], sizeof(double),
                        cudaMemcpyDeviceToHost);
                if( maximum < tempD ) maximum = tempD;

                cudaMemcpy( &tempD, &dataD_D[minindex-1], sizeof(double),
                        cudaMemcpyDeviceToHost);
                if( minimum > tempD ) minimum = tempD;
        }

        double phase_range = maximum - minimum;
        int steps = pow(2, bits);
        double step_size = phase_range/(double)steps;
        for(int i = 0; i < VECNUM; i++)
        {
                phase_stepping<<<BLOCKS,THREADS>>>(VECLEN, step_size, &dataN_D[i*
                    VECLEN]);
        }
}


void write_to_file( int type, const char * filename , cufftDoubleComplex* data)
{
        if(DESCRIPTIONS) cout << "Zapis␣do␣pliku:␣" << filename << endl;
        fstream plik;
        plik.open(filename, fstream::out | fstream::trunc);



        if(type == 1)
        {
                plik << setprecision(PRECISION) << N << endl << fA << endl << fB <<
                    endl;
                plik << setprecision(PRECISION) << F_RANGE << endl;
                plik << F_STEP*(fB-fA)/(double)N << endl;

                int temp = F_RANGE/(fB-fA)*(double)N;

                for(int i = -temp; i < temp; i += F_STEP)
                {
                        double modulus = pow(data[N/2+i].x,2.0) + pow(data[N/2+i].y
                            ,2.0);
                        double phase = atan2(data[N/2+i].y, data[N/2+i].x);
                        plik << setprecision(PRECISION) << (double)i*(fB-fA)/(double)N
                             << ",␣";
                        plik<< modulus << ",␣" << phase << endl;
                }
        }
        else if(type == 0)
        {
                plik << setprecision(PRECISION) << N << endl << fA << endl << fB <<
                    endl;
                plik << setprecision(PRECISION) << T_RANGE << endl;
                plik << (double) T_STEP/(fB-fA) << endl;

                int temp = T_RANGE*(fB-fA);


                for(int i = 0; i < temp; i += T_STEP)
                {
                        double modulus = pow(data[N-temp+i].x,2.0) + pow(data[N-temp+i
                            ].y,2.0);
                        double phase = atan2(data[N-temp+i].y, data[N-temp+i].x);
                        plik << setprecision(PRECISION) << (double)(i-temp)/(fB-fA) <<
                             ",␣"
                        plik << modulus << ",␣" << phase << endl;
                }
                for(int i = 0; i < temp; i += T_STEP)
                {
                        double modulus = pow(data[i].x,2.0) + pow(data[i].y,2.0);
                        double phase = atan2(data[i].y, data[i].x);
                        plik << setprecision(PRECISION) << (double)i/(fB-fA) << ",␣"
                        plik << modulus << ",␣" << phase << endl;
                }
        }
        else if(type == 2)
        {
                plik << setprecision(PRECISION) << N << endl << fA << endl << fB <<
                    endl;
                plik << setprecision(PRECISION) << T_RANGE << endl;
                plik << (double) T_STEP/(fB-fA) << endl;

                int temp = T_RANGE*(fB-fA);

                for(int i = 0; i < temp; i += T_STEP)
                {
                        double phase = atan2( data[N-temp+i].y, data[N-temp+i].x );
                        double modulus = sqrt( pow(data[N-temp+i].x,2.0)
                                + pow(data[N-temp+i].y,2.0) );
                        if( abs(phase) > cuPI/2.0 ) modulus = - modulus;
                        plik << setprecision(PRECISION) << (double)(i-temp)/(fB-fA) <<
                             ",␣";
                        plik << modulus << endl;
                }
                for(int i = 0; i < temp; i += T_STEP)
                {
                        double phase = atan2( data[i].y, data[i].x );
                        double modulus = sqrt( pow(data[i].x,2.0) + pow(data[i].y,2.0)
                             );
                        if( abs(phase) > cuPI/2.0 ) modulus = - modulus;
                        plik << setprecision(PRECISION) << (double)i/(fB-fA) << ",␣";
                        plik << modulus << endl;
                }
        }

        plik.close();
}
```

```
/*############################        KERNELE        ##################################*/

__global__ void gaussian_wavefunction_kernel(int Nk, double A, double B, double zero,
        double sigma, cufftDoubleComplex *vector)
{
        int tid = threadIdx.x + blockIdx.x*blockDim.x;
        if(tid < Nk)
        {
                double x = A + (B-A)*(double)tid/(double)Nk;
                x = 1.0/sqrt( sqrt(cuPI) * sigma) * exp( - pow( x/sigma, 2.0) /2.0 );

                vector[tid].x = x;
                vector[tid].y = 0.0;
        }
}


__global__ void put_linear_phase_kernel(int Nk, double A, double B, double zero,
        double K, cufftDoubleComplex *vector)
{
        int tid = threadIdx.x + blockIdx.x*blockDim.x;
        if(tid < Nk)
        {
                double x = A + (B-A) * (double)tid/(double)Nk;
                x = (x - zero) * K;
                double tempx = vector[tid].x;
                double tempy = vector[tid].y;
                vector[tid].x = tempx * cos(x) - tempy * sin(x);
                vector[tid].y = tempx * sin(x) + tempy * cos(x);
        }
}


__global__ void put_quadratic_phase_kernel(int Nk, double A, double B, double zero,
        double K, double C, double D, cufftDoubleComplex *vector)
{
        int tid = threadIdx.x + blockIdx.x*blockDim.x;
        if(tid < Nk)
        {
                double x = A + (B-A) * (double)tid/(double)Nk;
                x = K * pow(x - zero, 2.0) + C * (x - zero) + D;
                cufftDoubleComplex temp = vector[tid];
                cufftDoubleComplex temp2;
                temp2.x = temp.x * cos(x) - temp.y * sin(x);
                temp2.y = temp.x * sin(x) + temp.y * cos(x);
                vector[tid] = temp2;
        }

}

__global__ void put_phase_kernel(int Nk, cufftDoubleComplex *phase,
        cufftDoubleComplex *vector)
{
        int tid = threadIdx.x + blockIdx.x*blockDim.x;
        if(tid < Nk)
        {
                cufftDoubleComplex temp = phase[tid];
                double modulusD = sqrt( pow(temp.x, 2.0) + pow(temp.y, 2.0) );
                double phaseD = atan2( temp.y, temp.x );
                if( abs(phaseD) > cuPI/2.0 ) modulusD = - modulusD;
                temp = vector[tid];
                vector[tid].x = cos(modulusD) * temp.x - sin(modulusD) * temp.y;
                vector[tid].y = sin(modulusD) * temp.x + cos(modulusD) * temp.y;
        }

}

__global__ void phase_stepping(int Nk, double step_size, cufftDoubleComplex* vector)
```

```
{
        int tid = threadIdx.x + blockIdx.x*blockDim.x;
        if(tid < Nk)
        {
                cufftDoubleComplex temp = vector[tid];
                double modulus = sqrt( pow(temp.x, 2.0) + pow(temp.y, 2.0) );
                double phase = atan2( temp.y, temp.x );
                if( abs(phase) > cuPI/2.0 ) modulus = -modulus;
                int steps = (modulus-cuPIshift)/step_size;
                temp.x = steps*step_size+cuPIshift;
                temp.y = 0.0;
                vector[tid] = temp;
        }
}

__global__ void FWHM_filter(int Nk, double A, double B, double zero, double FWHM,
        double* vector)
{
        int tid = threadIdx.x + blockIdx.x*blockDim.x;
        if(tid < Nk)
        {
                double x = A + (B-A)*(double)tid/(double)Nk;

                if( x < -FWHM/2.0 || x > FWHM/2.0 )
                {
                        vector[tid] = 0.0;
                }
        }
}

__global__ void cufft_multiply_by_scalar(int Nk, double scalar,
        cufftDoubleComplex* vec)
{
        int tid = threadIdx.x + blockIdx.x*blockDim.x;
        if(tid < Nk)
        {
                cufftDoubleComplex temp = vec[tid];
                temp.x = scalar*temp.x;
                temp.y = scalar*temp.y;
                vec[tid] = temp;
        }
}

__global__ void cufft_multiply_by_vector(int Nk, double* vecD,
        cufftDoubleComplex* vecC)
{
        int tid = threadIdx.x + blockIdx.x*blockDim.x;
        if(tid < Nk)
        {
                cufftDoubleComplex temp = vecC[tid];
                double tempD = vecD[tid];
                temp.x = tempD*temp.x;
                temp.y = tempD*temp.y;
                vecC[tid] = temp;

                temp = vecC[N-tid-1];
                temp.x = tempD*temp.x;
                temp.y = tempD*temp.y;
                vecC[N-tid-1] = temp;
        }
}

__global__ void cufft_devide_by_vector(int Nk, double* vecD,
        cufftDoubleComplex* vecC)
{
        int tid = threadIdx.x + blockIdx.x*blockDim.x;
        if(tid < Nk)
        {
                cufftDoubleComplex temp = vecC[tid];
                double tempD = vecD[tid];
                temp.x = temp.x/tempD;
                temp.y = temp.y/tempD;
                vecC[tid] = temp;

                temp = vecC[N-tid-1];
                temp.x = temp.x/tempD;
                temp.y = temp.y/tempD;
                vecC[N-tid-1] = temp;
        }
}

__global__ void cufft_modulus_squared_to_double(int Nk, double* vecD,
        cufftDoubleComplex* vecC)
{
        int tid = threadIdx.x + blockIdx.x*blockDim.x;
        if(tid < Nk)
        {
                cufftDoubleComplex temp = vecC[tid];
                vecD[tid] = pow(temp.x, 2.0) + pow(temp.y, 2.0);
        }

}

__global__ void cufft_to_double(int Nk, double* vecD, double shift,
        cufftDoubleComplex* vecC)
{
        int tid = threadIdx.x + blockIdx.x*blockDim.x;
        if(tid < Nk)
        {
                cufftDoubleComplex temp = vecC[tid];
                double tempD = sqrt(pow(temp.x, 2.0) + pow(temp.y, 2.0));
                double phase = atan2( temp.y, temp.x);
                if( abs(phase) > cuPI/2.0 ) tempD = -tempD;
                vecD[tid] = tempD + shift;
        }

}


/*########################  PHASE GENERATION KERNELS  ##############################*/


__global__ void phase_quadratic(int Nk, int i, double range, double K,
        double aperture, cufftDoubleComplex *phase)
{
        int tid = threadIdx.x + blockIdx.x*blockDim.x;
        if(tid < Nk)
        {
                double x = i*range + range*(double)tid/(double)Nk;
                if( (aperture != 0) && (x > aperture) ) x = aperture;
                x = K * pow( x, 2.0);

                cufftDoubleComplex temp;
                temp.x = x + cuPIshift;
                temp.y = 0.0;

                phase[i*Nk+tid] = temp;
                phase[N-1-i*Nk-tid] = temp;
        }
}

__global__ void phase_fresnel(int Nk, int i, double range, double K, int factor,
        double aperture, cufftDoubleComplex *phase)
{
        int tid = threadIdx.x + blockIdx.x*blockDim.x;
        if(tid < Nk)
        {
                double x = i*range + range*(double)tid/(double)Nk;
                if( (aperture != 0) && (x > aperture) ) x = aperture;
                x =  K * pow( x, 2.0);
                x = fmod( x, factor*2.0*cuPI);

                cufftDoubleComplex temp;
                temp.x = x + cuPIshift - factor*cuPI;
                temp.y = 0.0;

                phase[i*Nk+tid] = temp;
                phase[N-1-i*Nk-tid] = temp;
        }
}

__global__ void phase_sinus(int Nk, int i, double range, double amplitude,
        double frequency, cufftDoubleComplex *phase)
{
        int tid = threadIdx.x + blockIdx.x*blockDim.x;
        if(tid < Nk)
        {
                double x = i*range + range*(double)tid/(double)Nk;
                x = amplitude*(1 - cos(2*cuPI*frequency*x) );

                cufftDoubleComplex temp;
                temp.x = x + cuPIshift;
                temp.y = 0.0;

                phase[i*Nk+tid] = temp;
                phase[N-1-i*Nk-tid] = temp;
        }
}
```