GRATING-PAIR CHIRPED PULSE AMPLIFICATION SYSTEM FOR A LOW POWER FEMTO-SECOND PULSE SOURCE

A Thesis

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Master of Science

by

Ishan Sharma

August 2011

ABSTRACT

A grating-pair Chirped Pulse Amplification (CPA) system is simulated and built with the goal to amplify a 535 fs sech^2 intensity pulse with a pulse energy of 3.36 pJ to a final output pulse of 400 fs width and 30 nJ of energy. The MATLAB simulation code, schematic design of the setup, experimental measurements, and analyses are presented. It is the hope that the reader will be able to easily simulate and build a CPA system by using the code and information presented in this thesis.

BIOGRAPHICAL SKETCH

Ishan Sharma, a resident of Edmonton, Alberta, Canada, joined Cornell University, Ithaca, NY in August of 2006 as an undergraduate in the College of Engineering. He affiliated with the School of Applied and Engineering Physics, while taking courses in both the School of Applied and Engineering Physics and the Department of Electrical and Computer Engineering. On May 30, 2010, Ishan graduated Class of 2010 from Cornell University with a Bachelor of Science degree in Applied and Engineering Physics with a minor in Electrical and Computer Engineering. Upon graduation, Ishan chose to further his understanding of the physical world and hence enrolled at Cornell University again in August of 2010. While pursuing a Master of Science degree in Applied Physics, he took courses in various subjects including nanofabrication, quantum optics, nonlinear optics, and solid state physics. In addition to taking courses, he has been working in the research laboratory of Prof. Chris Xu in the School of Applied and Engineering Physics. Ishan likes to balance his academic life with playing the piano, learning to play the guitar, exercising at the gym, cooking and taking easy strolls on the beautiful Cornell campus and the surrounding areas.

This thesis is dedicated to my parents, Rashmi and Pankaj Sharma, who supported me very generously in my academic and extracurricular pursuits. I would not have made it here without your unconditional support. I hope you forgive me for my mistakes and I hope I have been able to make you proud.

ACKNOWLEDGEMENTS

I would like to thank Prof. Chris Xu for giving me the opportunity and resources to work on this project. I would also like to thank all the members of the Xu group, especially Adam Straub, Ke Wang, Ji Cheng, Kriti Charan, and James van Howe, for their mentorship and insightful discussions. In addition, I would like to acknowledge Prof. David Muller for his instruction, advice and discussions on research projects, presentations, coursework, and career prospects.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 The Technique - Chirped Pulse Amplification

Fiber-based ultrafast technology is more robust and compact than its solid-state counterpart. Ultrashort-pulse solid state systems are quite complex with a large number of components inside a long free-space optical cavity. The fiber-based systems, on the other hand, use fiber as a gain medium thus allowing a compact integrated cavity design without the need for constant re-alignment. Despite their advantages, fiber based amplification systems are more prone to peak-power induced nonlinear effects than solid state systems. An order-of-magnitude comparison of typical mode sizes of $\sim 10\,\mu$m versus $1-3\,$mm and signal propagation lengths of $\sim 1-10\,$m versus $1-10\,$cm in fiber-based and bulk solid-state medium, respectively, clearly shows that a fiber amplifier is deemed to be 10^6-10^7 times more sensitive to nonlinear effects. Therefore, the key to avoiding these nonlinearities is to scale down the peak intensity inside the fiber core, so that the nonlinear effects can be reduced. This can be accomplished by a process called **Chirped-Pulse Amplification (CPA)** [1].

To outline the process of CPA (see Figure 1.1), an initial transform-limited pulse[1], given in Figure 1.1(a), is passed through a *pulse stretcher*, which causes the temporal pulse-width to increase as shown in Figure 1.1(b). Due to energy conservation, the total pulse energy should remain the same; however, the increased

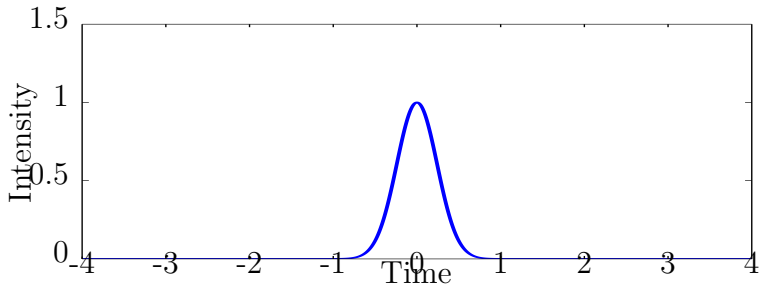
[1]A transform-limited pulse is the shortest pulse permitted by its bandwidth as dictated by the Fourier Transform. In other words, the time-bandwidth product is a minimum for a transform-limited pulse. The Full-Width-at-Half-Maximum (FWHM) time-bandwidth product is ~ 0.315 for a sech^2 pulse and is ~ 0.44 for a Gaussian pulse

temporal pulse-width would require the peak-power to decrease (the energy would be spread over a longer time period). This pulse can then be passed through a fiber-based amplifier, such as an Erbium-Doped Fiber Amplifier (EDFA) or an Erbium-Doped Waveguide Amplifier (EDWA), to increase the pulse energy to the desired value as shown in Figure 1.1(c). Although the pulse energy is increased, the peak-power remains low enough such that the nonlinearities are kept at a minimum. Finally, the pulse is compressed back to its transform-limited width as shown in Figure 1.1(d). Thus the result is an amplified transform limited pulse with minimal nonlinearities.

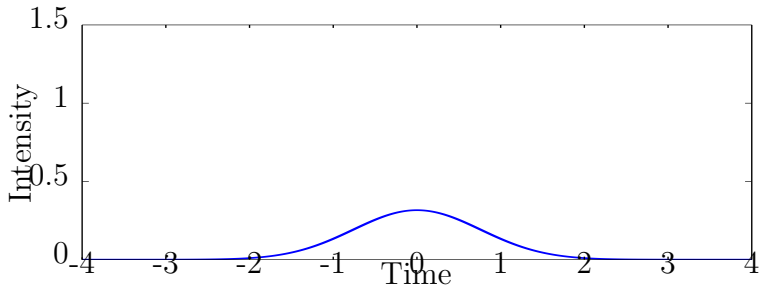
1.2 The Specifications

The input pulse is obtained from the seed monitoring output of the *Calmar Laser Model FLCPA-01C* with a measured spectral width of $\Delta\lambda = 4.713\,\text{nm}$ and a repetition rate of 20 MHz. Assuming a transform-limited pulse, a sech^2 pulse-shape has a pulse-width $T_{fwhm} = 535$ fs and a Gaussian pulse-shape has a pulse-width $T_{fwhm} = 747$ fs. The average power (including connector losses) is $67.10\,\mu\text{W}$ or -11.73 dBm. This means that the pulse energy is $E_p = 3.36\,\text{pJ}$.

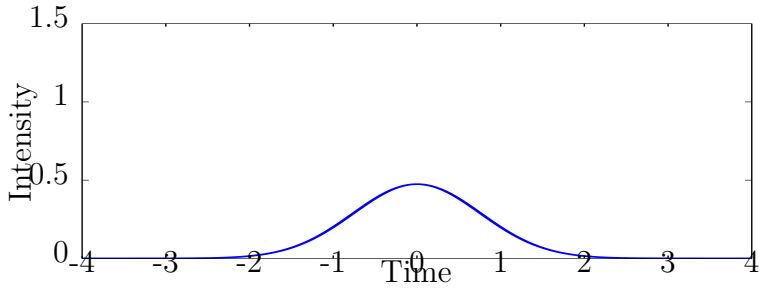
The desired output is an unchirped pulse with FWHM pulse width of 400 fs, a pulse energy of 30-50 nJ, and a repetition rate of 20 MHz.



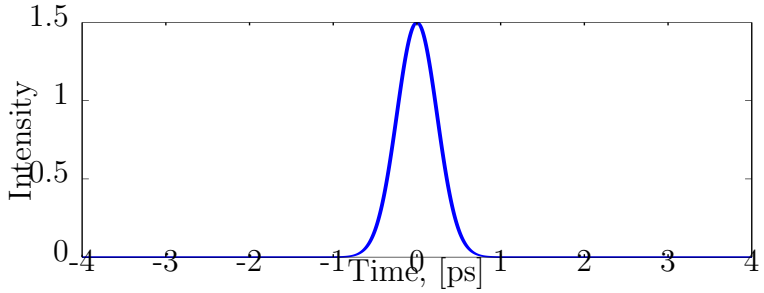
((a)) Initial Transform-Limited Pulse



((b)) Pulse after Stretching



((c)) Pulse after Amplification



((d)) Output Pulse

Figure 1.1: The evolution of a pulse in 1.1(a) as it passes through a stretcher, 1.1(b), a fiber based amplifier, 1.1(c), and a compressor to give an amplified pulse of the initial width, 1.1(d), as an output. *Note: The nonlinearities are ignored for simplicity.*

CHAPTER 2

THEORY AND SIMULATION

2.1 The Pulse-Propagation Equation

An electromagnetic pulse can be written mathematically as:

$$\mathbf{E}(\mathbf{r},t) = \frac{1}{2}\hat{x}[E(\mathbf{r},t)\exp(-i\omega_0 t) + c.c.] \tag{2.1}$$

where $\hat{x}$ is the polarization unit vector, $E(\mathbf{r},t)$ is a function of space and a slowly varying function of time, and ω_0 is the center frequency. The slowly varying envelope approximation is used here since the pulse width used ($\sim$600 fs) is two orders of magnitude larger than the optical period ($\sim$5 fs for 1.55 μm wavelength light). To study the pulse propagation, it is insightful to take the Fourier transform of the slowly varying amplitude, $E(\mathbf{r},t)$, of the electric field:

$$\tilde{E}(\mathbf{r},\omega-\omega_0) = \int_{-\infty}^{\infty} E(\mathbf{r},t)\exp(i(\omega-\omega_0)t)\,dt \tag{2.2}$$

The frequency domain amplitude of the electric field given in Equation 2.2 can be written in a form that separates it into an $x-$ and $y-$ dependant part, and a $z-$ dependent part as shown below:

$$\tilde{E}(\mathbf{r},\omega-\omega_0) = F(x,y)\tilde{A}(z,\omega-\omega_0)\exp(i\beta_0 z) \tag{2.3}$$

which solves the Helmholtz equation,

$$\nabla^2 \tilde{E} + \varepsilon(\omega) k_0^2 \tilde{E} = 0 \tag{2.4}$$

where $k_0 = \omega/c$ and $\varepsilon(\omega)$ is the dielectric constant. In Equation 2.3, $F(x, y)$ is the transverse modal distribution[1], $\tilde{A}(z, \omega - \omega_0)$ is a slowly varying function of z (commonly refered to as the pulse-shape, i.e. gaussian, hyperbolic secant, etc.), and β_0 is the wave number.

Combining Equations 2.1, 2.2 and 2.3 gives:

$$\mathbf{E}(\mathbf{r}, t) = \frac{1}{2}\hat{x}[F(x, y)A(z, t)\exp(i(\beta_0 z - \omega_0 t)) + c.c.] \tag{2.5}$$

where $A(z, t)$ is the inverse Fourier transform of $\tilde{A}(z, \omega - \omega_0)$ as given by:

$$A(z, t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \tilde{A}(z, \omega - \omega_0)\exp(-i(\omega - \omega_0)t)\, d\omega \tag{2.6}$$

With the electric field expressed in the form of Equation 2.5, the analysis given in Section 2.3 of Agrawal can followed to derive the generalized pulse-progation equation (also referred to as the generalized **Nonlinear Schrödinger Equation (NLSE)**: [2]

$$\frac{\partial A}{\partial z} + \frac{\alpha}{2}A + \frac{i\beta_2}{2}\frac{\partial^2 A}{\partial T^2} - \frac{\beta_3}{6}\frac{\partial^3 A}{\partial T^3} = i\gamma\left(|A|^2 A + \frac{i}{\omega_0}\frac{\partial}{\partial T}(|A|^2 A) - T_R A\frac{\partial |A|^2}{\partial T}\right) \tag{2.7}$$

where α is the absorption coefficient of the medium, γ is the nonlinear parameter,

[1]For a single-mode fiber, $F(x, y)$ corresponds to the modal distribution of the fundamental fiber mode HE_{11}, often approximated by a Gaussian distribution in $x-$ and $y-$ for simplicity.

β_2 and β_3 are the second- and third-order dispersion terms, respectively, T is the retarded time, $T \equiv t - z/v_g$, in the frame of reference moving with the pulse at group velocity v_g, and T_R is the first moment of the nonlinear repose function, $R(t)$ defined by:

$$T_R \equiv \int_0^\infty tR(t)\,dt \tag{2.8}$$

where $R(t)$ is the nonlinear response function dependent on the electronic and nuclear responses due to Raman scattering most dominant in ultrashort pulses (<1ps). For more information on Raman scattering, refer to Chapter 8 and 12 of Agrawal [2].

To obtain the second- and third-order dispersion terms, the wave number $\beta(\omega)$ is Taylor expanded around the carrier frequency ω_0 as given by:

$$\beta(\omega) = \beta_0 + (\omega - \omega_0)\beta_1 + \frac{1}{2}(\omega - \omega_0)^2\beta_2 + \frac{1}{6}(\omega - \omega_0)^3\beta_3 + \ldots \tag{2.9}$$

where the various β_m parameters are given by:

$$\beta_m = \left(\frac{d^m\beta}{d\omega^m}\right)_{\omega=\omega_0} \qquad (m = 1, 2, \ldots) \tag{2.10}$$

2.2 The Nonlinear Schrödinger Equation Solver

Although a simplified version of the NLSE can be solved analytically for certain pulse shapes, for example the Gaussian pulse shape, the solution to the generalized NLSE involving other pulse shapes, such as sech^2, has to be obtained numerically. More details on this can be found in Section 2.4 of Agrawal [2]. The numerical

NLSE solvers have been coded in various programming languages. The solver that has been used in this project is from Appendix 1 of Travers et al [3]. This particular solver is written in the MATLAB scripting language. A slightly modified version of the source code is given as Listing A.1 in Appendix A.

2.3 The Dispersive and Nonlinear Effects

A closer look at the NLSE given as Equation 2.7 reveals that there are three main effects that govern the evolution of a pulse through a medium. On the left hand side of Equation 2.7, the second term depicts the loss in the medium, while the third and fourth terms depict the effects of the second- and third-order dispersion. All the terms on the right hand side of Equation 2.7 summarize the nonlinear effects on the pulse propagation. Depending on the initial pulse width T_0 [2] and the pulse peak power P_0, dispersive or nonlinear effects may dominate. Hence, it is insightful to express the length scales over which each phenomenon is dominant. The dispersive effects are dominant at the *dispersion length*, L_D given by,

$$L_D = \frac{T_0^2}{|\beta_2|} \tag{2.11}$$

while the nonlinear effects are dominant at the *nonlinear length*, L_{NL}, given by,

$$L_{NL} = \frac{1}{\gamma P_0} \tag{2.12}$$

[2] T_0 is the half-width at the $1/e$ intensity point and is related to the FWHM pulse width by the following factors:

sech^2	$T_{fwhm} \approx 1.763 T_0$
Gaussian	$T_{fwhm} \approx 1.665 T_0$

In the given system, with the specifications that were outlined in Section 1.2, the dispersion length can be calculated to be,

$$L_D \approx \frac{(0.535\,\mathrm{ps}/1.763)^2}{|20\,\mathrm{ps}^2/\,\mathrm{km}|} = 4.6\,\mathrm{m} \tag{2.13}$$

where a sech^2 pulse is assumed, and the β_2 of single mode fiber at $\lambda = 1.55\,\mu\mathrm{m}$ is taken to be $\approx 20\,\mathrm{ps}^2/\,\mathrm{km}$ [2]. Similarly, the nonlinear length can also be calculated to be,

$$L_{NL} \approx \frac{1}{(2\,\mathrm{W}^{-1}\,\mathrm{km}^{-1})(9.88\times 10^4\,\mathrm{W})} = 5.1\,\mathrm{mm} \tag{2.14}$$

where a 30 nJ sech^2 pulse is assumed to calculated the peak power, and the γ of single mode fiber at $\lambda = 1.55\,\mu\mathrm{m}$ is taken to be $\approx 2\,\mathrm{W}^{-1}\,\mathrm{km}^{-1}$ [2]. From the calculated values for L_D and L_{NL}, it is evident that if the pulse were to be passed directly (i.e., without stretching) through a couple of EDFAs, which can contain upto 30 m of fiber each, the pulses would accumulate both dispersion and nonlinearity. Therefore, it is *essential* to stretch the pulse before amplification.

2.4 The Simulation

The entire setup can be simulated given the input pulse specifications, the fiber parameters and the dispersion of the stretcher-compressor system. The simulation has been coded in MATLAB so that the NLSE solver function given in Listing A.1 can be easily called. The entire simulation can be split into 6 stages as follows:

I Pre-Amplification

II Propagation to Stretcher

III Pulse Stretcher

IV Amplification

V Propagation to Compressor

VI Pulse Compressor

The simulation code for each of these stages is discussed below along with the results of the simulations. This code, writen by Ishan Sharma, is a presented in its entirety as Listing B.1 Appendix B.

2.4.1 Initial Pulse Parameters

Since the core of this program is a numerical NLSE solver, the initial pulse is defined in terms of discrete points in the time domain. First, a suitable time window is chosen to maintain a good balance between the steadfastness of the numerical methods and the use of the computer's memory or computation time. After several tries, a time window of 1400 ps was found to be acceptable. Similarly, a suitable number of grid points in chosen to maintain a good balance between resolution and the use of the computer's memory or computation time - 2^{19} was found to be a good number. The time grid, in the units of picoseconds, is then created of the defined time length with the specified resolution. Since the NLSE assumes a *retarded time*, i.e., a time frame travelling with the pulse, the time grid is centered about 0 with an equal number of negative and positive time grid values. The frequency grid, in the units of radians, is created by using the relation $\omega = 2\pi/T$. The MATLAB code for the time and frequency grids is given in Listing 2.1 below.

Listing 2.1: MATLAB code for creating the Time and Frequency Grids

```
% ==========================================
% Definitions of the time and frequency grids
% ==========================================

cpt = cputime;                                  % time since MATLAB started (initial reference time)
n = 2^19;                                       % number of grid points
twidth = 1400;                                  % width of time window [ps]
c = 299792458*1e9/1e12;                         % speed of light [nm/ps]
wavelength = 1550;                              % reference wavelength [nm]
w0 = (2.0*pi*c)/wavelength;                     % reference frequency [rads/ps]
dT = twidth/n;                                  % time interval
T = (-n/2:n/2-1)*dT;                            % time grid (NB: better than T = linspace(-twidth/2, twidth/2, n), since there is now a point at T=0)
V = 2*pi*(-n/2:n/2-1)/(n*dT);                   % frequency grid, ' means transpose
Vabs = V + w0;                                  % absolute frequency grid
WL = 2*pi*c./Vabs;                              % wavelength grid
```

The initial pulse in the simulation is based on the spectrum trace of the Calmar Laser Seed Monitoring Output measured by an Optical Spectrum Analyzer (OSA). The specifications are coded as given in Listing 2.2. The user has the option of choosing a sech^2 or a Gaussian shaped intensity pulse. Note that only the slowly-varying envelope of the electric field is defined here, since that forms the input for the NLSE solver. The *fftshift* and *fft* functions of MATLAB are used to calculate the Fourier Transform of $E(t)$; and the *ifftshift* and *ifft* functions of MATLAB are used to calculate the Inverse Fourier Transform of $\bar{E}(\omega)$.

Listing 2.2: MATLAB code for the Initial Pulse

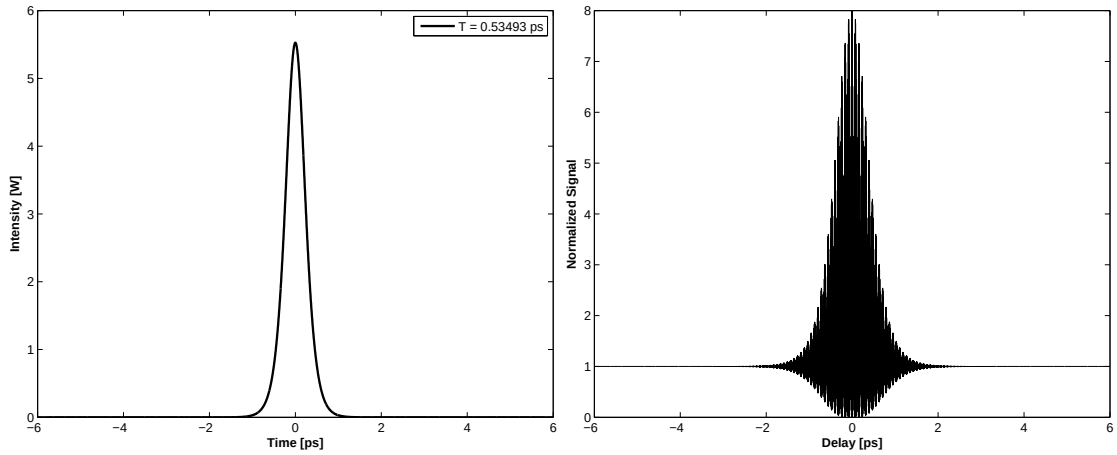
```
% ==========================================
% Definition of the Input Pulse - Stage 0
% ==========================================

reprate = 20*10^6;                              % repetition rate [Hz]
FWHM = 0.3127;                                  % FWHM [ps]
```

```
power = 0.88*3.36/FWHM;                                % peak power of input [W] = 0.88*E_pulse/T_fwhm (NB: E_pulse is in pJ)
t0 = FWHM/1.763;                                       % sech duration of input [ps]
E = sqrt(power)*sech(T/t0);                            % sech input field [W(1/2)]
% t0 = FWHM/(2*log(2))^0.5;                            % guassian duration of input [ps]
% E = sqrt(power)*exp(-T.^2/(2*t0^2));                 % gaussian input field [W(1/2)]
cx0 = fftshift(fft(E));                                % FT of input field
E0 = ifft(ifftshift(cx0));                             % inverse_FT of FT of input field (sanity check)
%plot(T,E,T,E0)                                        % plots should overlap (sanity check)

% ====== Figures of Merit ======%
E0_energy = sum(abs(E0).^2*dT);                        % should match E_pulse entered above (sanity check) [pJ]
I0 = abs(E0).^2;                                       % pulse intensity [W]
t_FWHM0 = fwhm(T,I0);                                  % fwhm pulse width [ps]
spec0 = abs(cx0).^2;                                   % fwhm pulse width [ps]
w_FWHM0 = fwhm(V,spec0);                               % pulse spectrum
lambda_FWHM0 = (w_FWHM0/(2*pi))*wavelength^2/c;        % should match the delta_lambda from OSA [nm]
```

After each stage, a few useful figures of merit are calculated. These include the pulse energy (to check energy conservation), the pulse intensity profile, and the pulse spectral profile. The Full-Width-at-Half-Maximum temporal and spectral pulse widths are calculated using the *fwhm* function obtained from MATLAB Central File Exchange [4] and presented in its entirety in Appendix C. The intensity pulse and the autocorrelation trace for this stage are plotted and presented in Figure 2.1. The function used to generate the autocorrelation trace is given in its entirety in Appendix D



((a)) ((b))

Figure 2.1: The simulated intensity profile and autocorrelation trace of the initial pulse

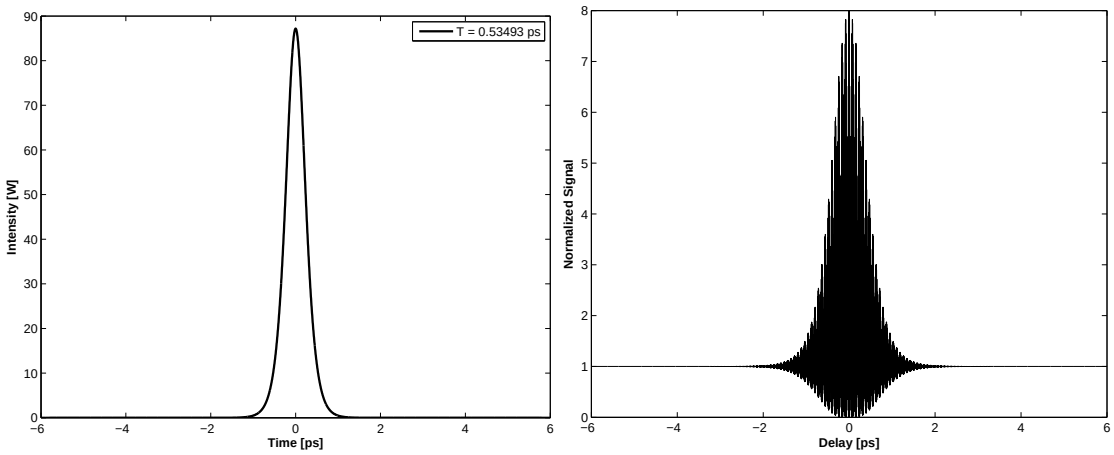
2.4.2 Stage I: Pre-Amplification

The pulse pre-amplification is represented by a gain factor that is multiplied to the electric field. The pulse is amplified to an average power of $\sim 1\,\mathrm{mW}$. As expected, the pulse peak power increases without any change in the pulse width (Figure 2.2(a)), while the autocorrelation trace remains the same (Figure 2.2(b)).

Listing 2.3: MATLAB code for the Stage I

```
% ================================================
% Pre-Amplifier (EDFA/EDWA) - Stage 1
% ================================================

avgpower_preamp = 1.06*10^-3;                    % Average power ↩
    after EDFA [W]
E_pulse_preamp = 10^12*avgpower_preamp/reprate; % Calculate the ↩
    pulse energy after EDFA [pJ]
P_peak_preamp = 0.88*E_pulse_preamp/t_FWHM0;     % Calculate peak ↩
    power [W]
gain_preamp = sqrt(P_peak_preamp)/sqrt(power);  % gain for preamp
E1 = E0.*gain_preamp;
cx1 = fftshift(fft(E1));
```



((a)) ((b))

Figure 2.2: The simulated intensity profile and autocorrelation trace of the pulse after pre-amplification (Stage I)

2.4.3 Stage II: Propagation through Fiber between the Preamp and the Stretcher

Since fiber amplifiers usually have 10-30 m of fiber through which the pulse propagates after amplification, the NLSE solver is used to simulate the effect of this propagation on the pulse. The output pulse from Stage I is used as input for this stage. A fiber length of 30 m is used as shown in the code presented as Listing 2.4. The output intensity curve and autocorrelation trace is presented in Figure 2.3.

Listing 2.4: MATLAB code for the Stage II

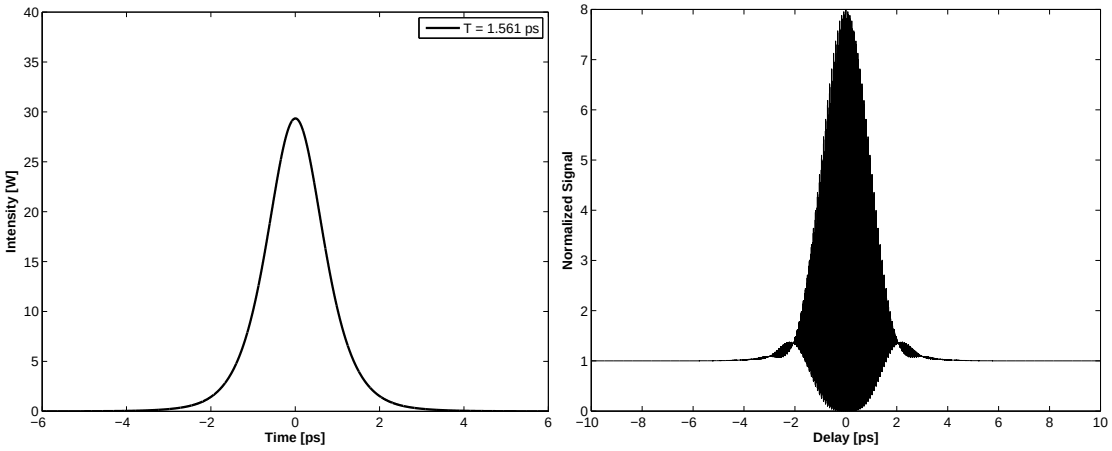
```
% ================================================
% Propagation through fiber between preamp and stretcher - Stage 2
% ================================================

flength1 = 30;                                  % fibre length [m]

% ===== Simulation Parameters ===== %
nsaves = 20;                                    % number of length ↩
    steps to save field at

% ===== Propagate Field ===== %
```

```
[Z, AT2, AW2, W] = gnlse(T, E1, w0, gamma, betas, loss, fr, RT, ↩
    flength1, nsaves);
E2 = AT2(nsaves,:);
cx2 = AW2(nsaves,:);
```



((a)) ((b))

Figure 2.3: The simulated intensity profile and autocorrelation trace of the pulse after propagation through fiber between the preamp and the stretcher (Stage II)

As can be seen from Figure 2.3, the pulse is stretched; however, experimentally, it was found that the pulse width is smaller that the input pulse as shown in Figure 4.3. A more accurate simulation can be obtained by incorporating the effects of the EDFA more carefully, perhaps based on Section 4.2.6 of Agrawal [2]. To ensure that the simulation matches the experiment, a sech^2 intensity pulse with the pulse width obtained from the experiment (Figure 4.3) is used for the subsequent stages instead of using the output of Stage II (Figure 2.3).

2.4.4 Stage III: The Pulse Stretcher

For the pulse stretcher, the function *gratingpair* given as Listing 3.1 is called with the desired grating separation and angle of incidence as the input parameters.

The group-velocity dispersion and the third-order dispersion are the outputs. As mentioned above, a sech^2 intensity pulse with a pulse width of 0.313 ps is used instead of the output pulse from Stage II. The stretcher parameters are as follows: 1000 lines/mm gratings, 30 cm focal length lenses, 16 cm of separation between the gratings and the lenses, and an incidence angle that is 9° away from the Littrow angle. These values were chosen to stretch the pulse to $\sim 136\,\text{ps}$, which is the same pulse width obtained by Howe et al [5] in a similar experimental setup [3]. The output intensity curve and autocorrelation trace for this stage is presented in Figure 2.4.

Listing 2.5: MATLAB code for the Stage III

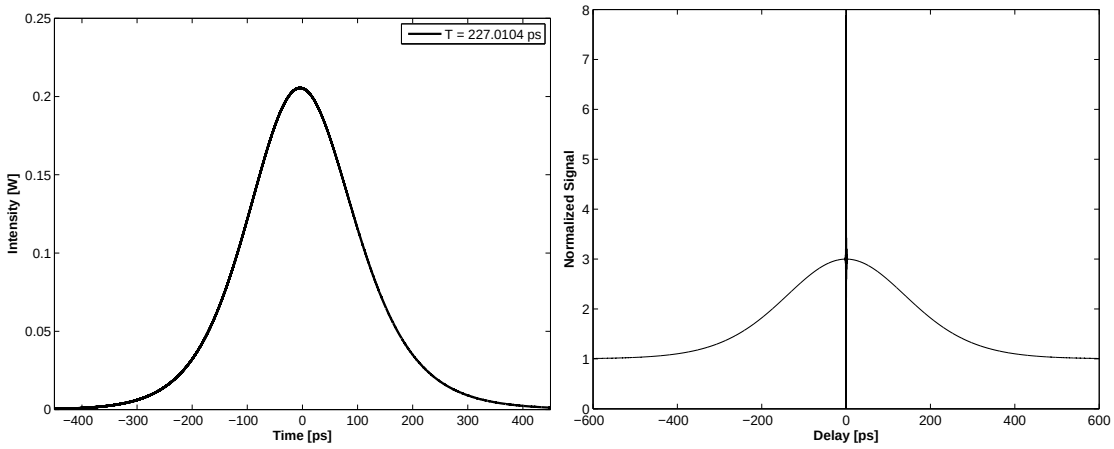
```
% ================================================
% Grating-pair stretcher - Stage 3
% ================================================

[GVD1, TOD1] = gratingpair(1, wavelength, 1.5*lambda_FWHM1, 1000, 0,↩
    30, 16, 16, 50.8-9)
cx3=exp(((1i*(1/2)*GVD1).*V.^2)+((1i*(1/6)*TOD1).*V.^3)).*cx1;
E3 = ifft(ifftshift(cx3));
```

2.4.5 Stage IV: The Pulse Amplifier

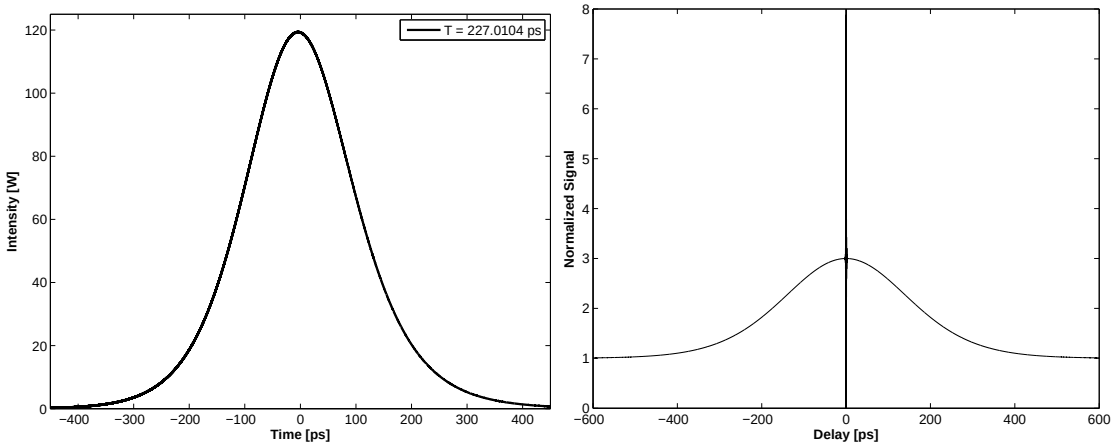
To get the desired pulse energy of 30 nJ, a gain of ~ 24.1 is required from this stage. The code is similar to that of the preamp given in Listing 2.3. The output intensity curve and autocorrelation trace for this stage is presented in Figure 2.5.

[3] Upon talking to the author of [5], it was realized that the experimental stretch factor is usually less than the simulated stretch factor. Therefore, the pulse is over-stretched to ensure that the desired experimental pulse width is obtained.



((a)) ((b))

Figure 2.4: The simulated intensity profile and autocorrelation trace of the pulse after the pulse stretcher (Stage III)



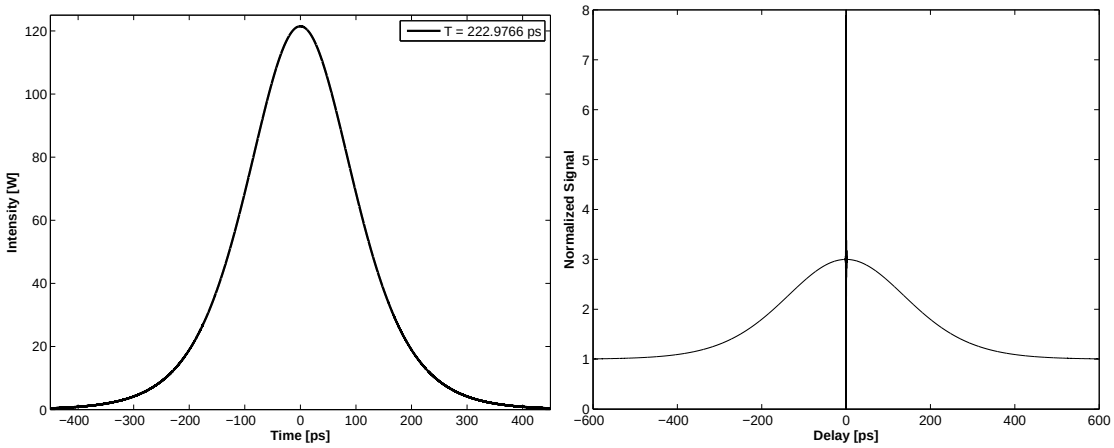
((a)) ((b))

Figure 2.5: The simulated intensity profile and autocorrelation trace of the pulse after the amplifier (Stage IV)

2.4.6 Stage V: Propagation through Fiber between the Amplifier and the Compressor

The pulse is propagated through 30 m of fiber (typical fiber length in an EDFA) to get a sense of how the pulse is affected. The code is similar to that of the

pulse propagation between the preamp and the stretcher given in Listing 2.4. The output intensity curve and autocorrelation trace for the pulse propagation between the amplifier and the compressor (Stage V) is presented in Figure 2.6, which shows that the FWHM pulse width after propagation does not change by a significant amount, as desired.



((a)) ((b))

Figure 2.6: The simulated intensity profile and autocorrelation trace of the pulse after propagation between amplifier and compressor (Stage V)

2.4.7 Stage VI: The Pulse Compressor

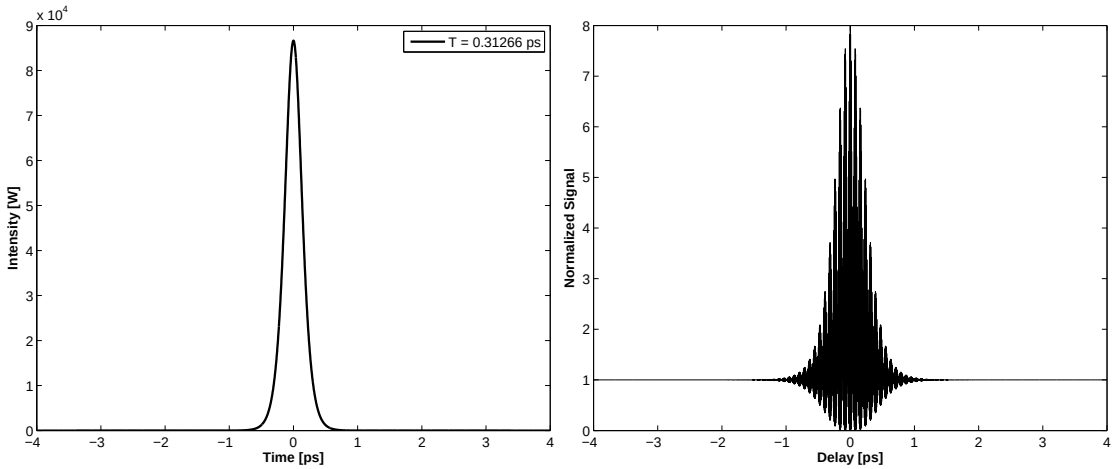
In this stage, the pulse is compressed back to femtosecond pulse-width. The parameters for the grating-pair were calculated to ensure that the Group Velocity Dispersion of the compressor canceled that of the stretcher. Since the propagation through fiber in Stage V does not affect the pulse significantly, the output from Stage IV is used as input to the compressor. This also ensures that the stretcher and the compressor are matched (small adjustments can be made to compensate

for miscellaneous GVD contributions). As can be seen from the output intensity curve and autocorrelation trace presented in Figure 2.7, the original pulse width is recovered with a 30 nJ pulse energy (*Note:* The distance between the gratings can be adjusted to achieve the desired 400 fs pulse width).

Listing 2.6: MATLAB code for the Stage VI

```
% ================================================
% Grating-pair compressor - Stage 6
% ================================================

[GVD2, TOD2] = gratingpair(0, wavelength, lambda_FWHM4, 1200, ↩
    1.636877881749541e+001, 0, 0,0, 74.4)
cx6=exp(((1i*(1/2)*GVD2).*V.^2)+((1i*(1/6)*TOD2).*V.^3)).*cx4;
E6 = ifft(ifftshift(cx6));
```



((a)) ((b))

Figure 2.7: The simulated intensity profile and autocorrelation trace of the pulse after the compressor (Stage VI)

CHAPTER 3

CPA SYSTEM DESIGN AND FABRICATING PROCEDURE

3.1 The Grating-Pair System

There are many ways of stretching a pulse by introducing dispersion, such as propagation through material (fiber), a prism pair or a grating pair; however, the grating-pair is the cleanest method that also allows the design of a matched compressor to get back the initial pulse width. It was realized by Martinez that by placing a telescope between a grating pair, the dispersion is controlled by the effective distance between the second grating and the image of the first grating. When this distance is optically made to be negative, the arrangement has exactly the opposite dispersion of a grating compressor. This forms the basis of a perfectly matched stretcher-compressor pair [6] [7].

3.1.1 The Equations for the Grating-Pair System

When light at a particular wavelength, λ, is incident on a grating at an angle of incidence, γ, from the normal to the surface of the grating, the different wavelengths in the beam get diffracted by the diffracted angle, θ, as governed by Equation 3.1. Therefore, the pulse will become negatively chirped (short wavelengths precede longer wavelengths).

$$\sin\gamma + \sin\theta = \frac{\lambda}{d} \tag{3.1}$$

As calculated by Backus et al, the phase contribution due to a double pass through the grating-pair system, where a pulse travels through the grating-pair once, gets

reflected by a retro-reflector, and passes again through the grating-pair system, is given by Equation 3.2 [8]:

$$\phi(\omega) = \frac{4\omega L_{eff}}{c}\left[1 - \left(\frac{2\pi c}{\omega d} - \sin\gamma\right)^2\right]^{1/2} \tag{3.2}$$

where L_{eff} is the effective grating separation, and d is the groove spacing of the grating in the unit of length. To find the group velocity dispersion (GVD) and the third order dispersion (TOD), Equation 3.2 can be differentiated w.r.t. ω twice and thrice, respectively. This gives the expression for GVD in Equation 3.3 and the expression for TOD in Equation 3.4:

$$\frac{\partial^2\phi(\omega)}{\partial\omega^2} = -\frac{2\lambda^3 L_{eff}}{2\pi c^2 d^2}\left[1 - \left(\frac{\lambda}{d} - \sin\gamma\right)^2\right]^{-3/2} \tag{3.3}$$

$$\frac{\partial^3\phi(\omega)}{\partial\omega^3} = -\frac{3}{2\pi}\frac{\lambda}{c}\frac{\partial^2\phi(\omega)}{\partial\omega^2}\left(\frac{1 + \frac{\lambda}{d}\sin\gamma - \sin^2\gamma}{\left[1 - \left(\frac{\lambda}{d} - \sin\gamma\right)^2\right]}\right) \tag{3.4}$$

Equations 3.2, 3.3 and 3.4 are valid for both the stretcher and the compressor with one major difference: while the L_{eff} is the distance between the grating surfaces in the case of the compressor, the L_{eff} for the stretcher is given by $-2(f - s)$, where f is the focal length of the lenses and s is the distance between the lens and the grating.

3.1.2 The MATLAB Function *gratingpair*

As part of this project, the function *gratingpair* was coded in MATLAB by Ishan Sharma to take the grating-pair setup parameters as inputs and output the GVD

and TOD. This function also calculates the spatial chirp on the pulse after a single pass through the grating-pair system based on the input bandwidth; the calculated spatial chirp can help in the system design to ensure that the beam is not clipped. The code for this function is included below as Listing 3.1. To calculate the grating-pair parameters required to achieve a desired GVD, a program is given as Listing E.1 in Appendix E.

Listing 3.1: MATLAB code for the function *gratingpair*

```
function [GVD_ps, TOD_ps] = gratingpair(stretcher_bool, lambda_nm, ↩
    delta_lambda_nm, sgroove, Lg_cm, f_cm, s1_cm, s2_cm, thetai_deg)
%GRATINGPAIR Calculation of the GVD, TOD and FOD for a grating-pair
%
% Coded by Ishan Sharma, 2011.
%
% Calculation of the GVD, TOD and FOD for a grating-pair-telescope ↩
    stretcher
% based on "Backus et al. Rev.Sci.Instrum., Vol.69, No.3, March ↩
    1998.
%
% Implementation is as follows:
% [GVD_ps, TOD_ps] = gratingpair(stretcher_bool, lambda_nm, sgroove,↩
     Lg_cm, f_cm, s_cm)


% ================================================
% User input variables
% ================================================

stretcher = stretcher_bool;          % enter 1 if stretcher (with ↩
    telescope), else 0
lambda = lambda_nm*10^-9;            % center wavelength [m]
delta_lambda = delta_lambda_nm*10^-9; % wavelength spread [m]
f = f_cm*10^-2;                      % focal length of lenses [m]
s1 = s1_cm*10^-2;                    % distance between lens and ↩
    grating 1 [m]
s2 = s2_cm*10^-2;                    % distance between lens and ↩
    grating 2 [m]
Lg = Lg_cm*10^-2;                    % distance between gratings (↩
    grating separation) [m]
sgroove;                             % groove spacing [lines/mm]


% ================================================
% Conversion of input to relevant variables
% ================================================

c = 2.99792458*10^8;                 % speed of light [m/s]
if stretcher == 1
```

```
33      Leff = -((f-s1)+(f-s2));          % effective grating separation [m]
34      Lg = f+f+s1+s2;                   % Actual grating separation [m]
35  else
36      Leff = Lg
37  end
38  d = 1/(sgroove*10^3);                 % line-width [m]
39  thetalittrow = asin(lambda/(2*d));    % calculation of the littrow angle [rads]
40  if thetai_deg == 0
41      thetai = thetalittrow;            % working at the Littrow angle [rads]
42  else
43      thetai = thetai_deg*(pi/180);     % working at user-input incidence angle [rads]
44  end
45  thetai_deg = thetai/(pi/180)
46  thetad = asin((lambda/d)-sin(thetai));
47  thetad_deg = thetad/(pi/180)
48  lambda_min = lambda - delta_lambda/2;
49  lambda_max = lambda + delta_lambda/2;
50
51
52  % ================================================
53  % Calculation of GVD, TOD and FOD
54  % ================================================
55
56  GVD = -(2*lambda^3*Leff/(2*pi*c^2*d^2))*(1-((lambda/d)-sin(thetai))^2)^-(3/2);
57  TOD = -(3/(2*pi)*lambda/c)*GVD*((1+(lambda*sin(thetai)/d)-(sin(thetai))^2)/(1-((lambda/d)-sin(thetai))^2));
58  FOD = (6*d^2/c^2)*GVD*(((80*lambda^2/d^2)+20-(48*lambda^2*cos(thetai)/d^2)+(16*cos(2*thetai))-(4*cos(4*thetai))+(32*lambda*sin(thetai)/d)+(32*lambda*sin(3*thetai)/d))/(((-8*lambda/d)+(4*d/lambda)+(4*d*cos(2*thetai)/lambda)+(32*sin(thetai)))^2-(TOD*(3*2*pi*lambda/c)*((1+(lambda*sin(thetai)/d)-(sin(thetai))^2)/(1-((lambda/d)-sin(thetai))^2)))));
59  GVD_ps = GVD*(10^12)^2;
60  TOD_ps = TOD*(10^12)^3;
61  FOD_ps = FOD*(10^12)^4;
62
63
64  % ================================================
65  % Calculation of spatial chirp
66  % ================================================
67  % Assume that the initial beam spot is small (which it is). So thetai is
68  % the same for blue and red.
69
70  thetad_min = asin((lambda_min/d)-sin(thetai));
71  thetad_max = asin((lambda_max/d)-sin(thetai));
72  spatialspread = (tan(thetad_max-thetad)*Lg)+(tan(thetad-thetad_min)*Lg) % in [m]
73
74  end
```

3.2 CPA Design Schematics

The schematic design for the entire CPA system is given below as Figure 3.1.

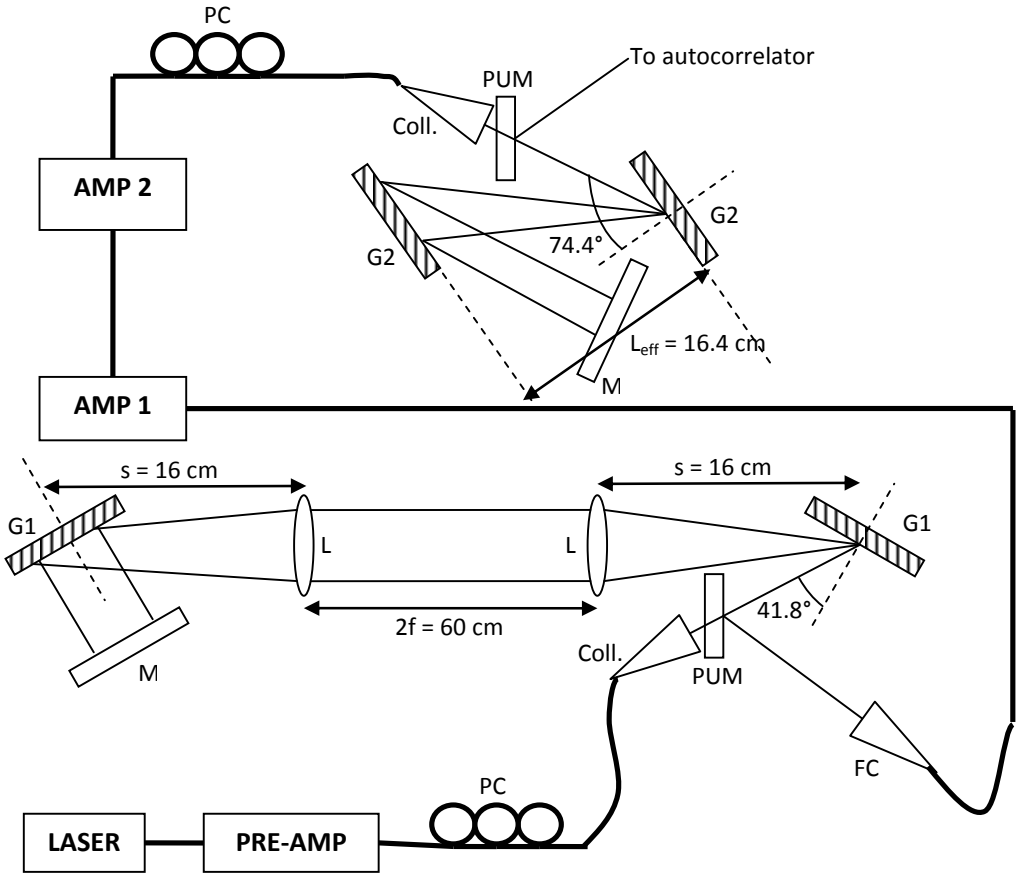


Figure 3.1: Design Schematics of the CPA system. PC, polarization controller; Coll., fiber collimater; G1, 2-inch^2 grating (1000 lines/mm); L, 2-inch double convex lens (30 cm focal length); M, mirror that offsets beam vertically; PUM, pick-up mirror vertically offset; FC, fiber collimator; G2, 2-inch^2 grating (1200 lines/mm)

LASER: Calmar Laser Model: FLCPA-01C

PRE-AMP: Amonics Model: AEDFA-C-23I-B-FA with Diode 1 at 25 mA and

Diode 2 at 25 mA

AMP1: AFC Model: RS-232

AMP2: IPG Photonics Model: EAR-1K-C-W

Since the grating efficiency is polarization sensitive, the fiber-based paddle polarization controllers are used to maximize the power of the diffracted beam. The entire stretcher is first aligned using irises and continuous wave (CW) light. Pulsed light is then used to ensure that the spatially chirped beam is not being clipped. The return beam in the stretcher is displaced slightly above the incident beam. The upward displacement switches to downward displacement through the telescope and the output is collected by the pickup-mirror that is placed below the incident beam. For the fiber collimator, a 0.18 N.A. aspheric lens is used in conjunction with a fiber tip on a Nanomax translation stage. The second grating might need to be tilted to reduce spatial chirp in the output beam. Spatial chirp in the output beam causes decreased bandwidth in the light coupled into fiber. A similar procedure is followed in aligning the compressor.

CHAPTER 4

EXPERIMENTAL DATA

4.1 Spectrum Measurements and Autocorrelation Traces or Pulse-width Measurements

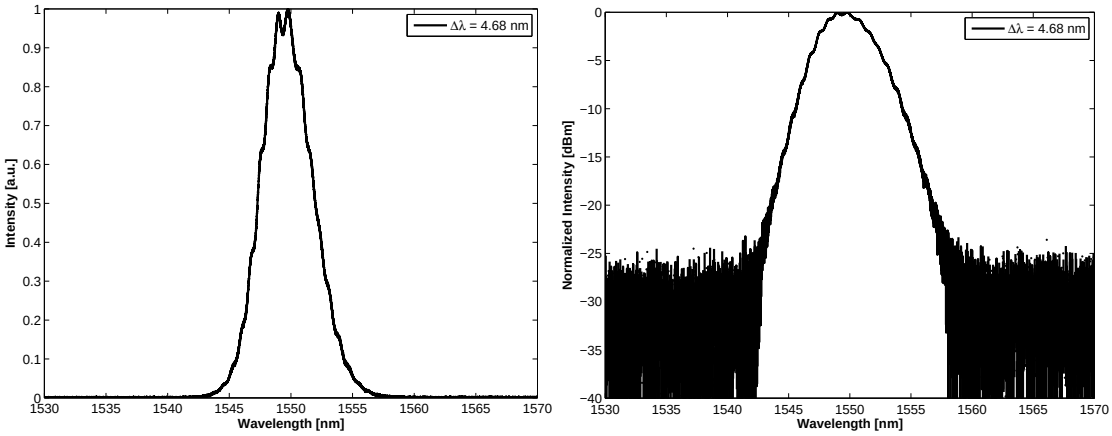
The spectrum measurements obtained from the Optical Spectrum Analyser (ANDO Model: AQ6317) for each stage are given below in linear and logarithmic scales. To gain information about the pulse width, autocorrelation traces and calculated intensity pulse plots are given. In the cases where the pulse width was too big for the scanning range of the motor available or the pulse peak-power was too low to trigger a two-photon response from the detector, the pulse width traces from the Sampling Oscilloscope (AGILENT Model: 86100A) are given.

4.1.1 Initial Pulse

Since the power from the laser is quite low, an autocorrelation trace could not be done; therefore, only the spectrum measurements are given in Figure 4.1. The average power is measured to be 67.10 μW.

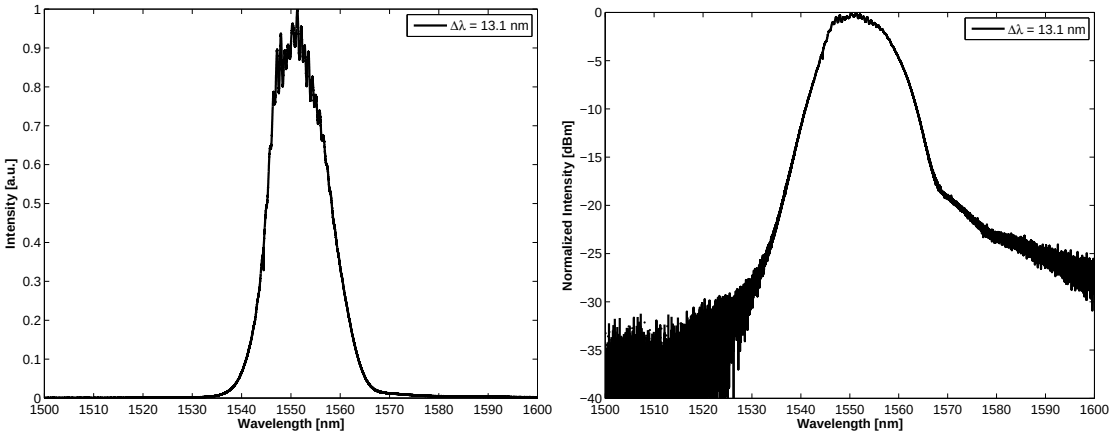
4.1.2 Pre-amplifier

Both the spectrum measurements and the autocorrelation measurements are given for this stage in Figures 4.2 and 4.3. The average power is measured to be 1.06 mW.



((a)) ((b))

Figure 4.1: The measured spectrum of the initial pulse in 4.1(a) linear scale and 4.1(b) logarithmic scale

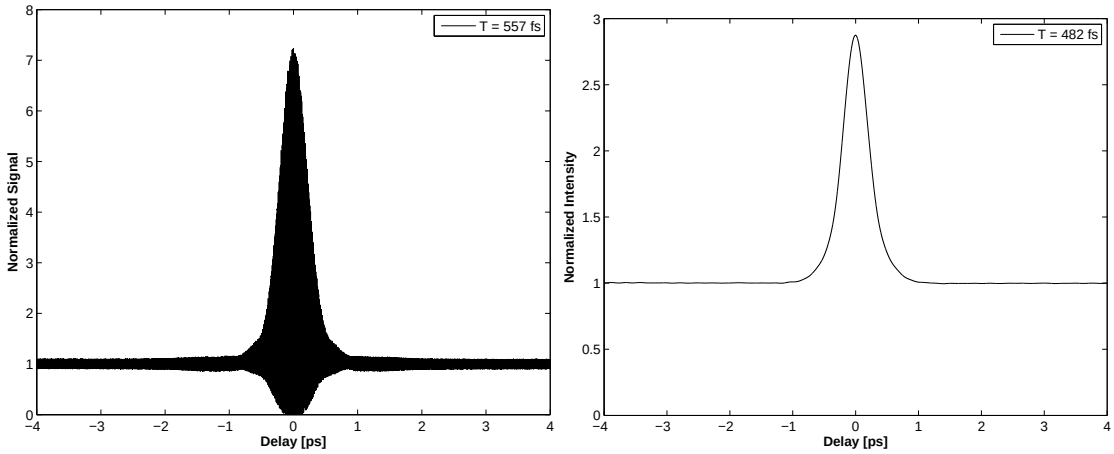


((a)) ((b))

Figure 4.2: The measured spectrum of the pulse after the pre-amplifier in 4.2(a) linear scale and 4.2(b) logarithmic scale

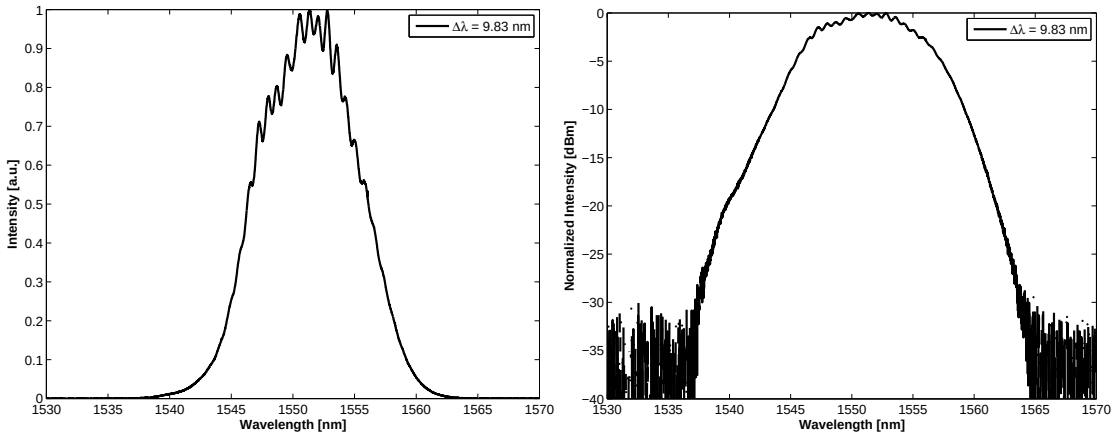
4.1.3 Pulse Stretcher

The spectrum measurements are presented in Figure 4.4. It was difficult to obtain an autocorrelation trace, so the Sampling Oscilloscope is used to measure the pulse width, given in Figure 4.5.



((a)) ((b))

Figure 4.3: The measured autocorrelation trace of the pulse after the pre-amplifier



((a)) ((b))

Figure 4.4: The measured spectrum of the pulse after the pulse stretcher in 4.4(a) linear scale and 4.4(b) logarithmic scale

4.1.4 Pulse Compressor

In order to ensure that the pulse compressor works, it is connected directly to the pre-amplifier (thus by-passing the stretcher altogether). If the compressor is designed correctly, then it should have the same stretcher factor as the stretcher,

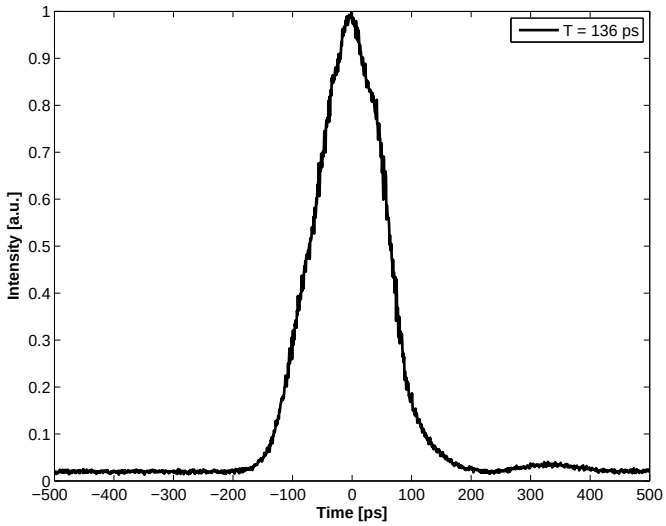
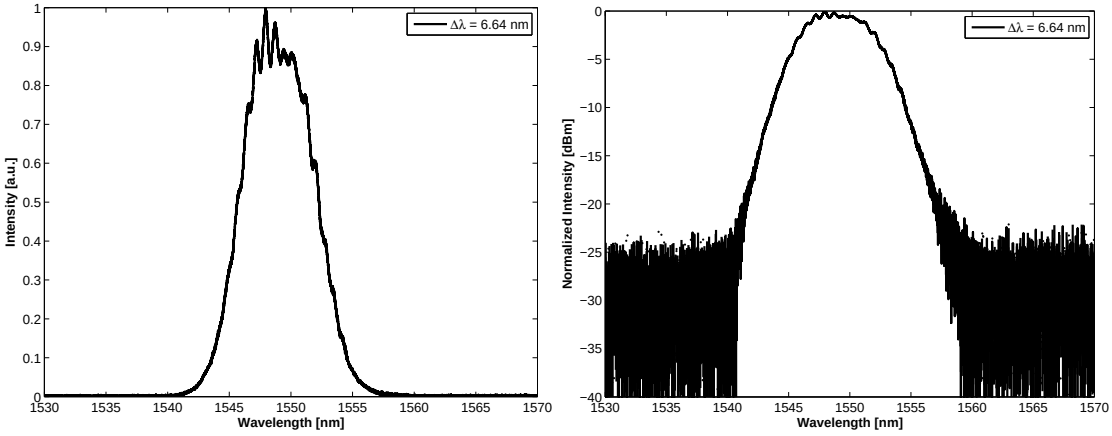


Figure 4.5: The measured pulse width after the pulse stretcher

thereby giving a pulse width close to that of the stretcher. Once again the spectrum was measured (Figure 4.6) along with the pulse width (Figure 4.7) using the Sampling Oscilloscope.



((a)) ((b))

Figure 4.6: The measured spectrum of the pulse after the pulse compressor (bypassing the stretcher) in 4.6(a) linear scale and 4.6(b) logarithmic scale

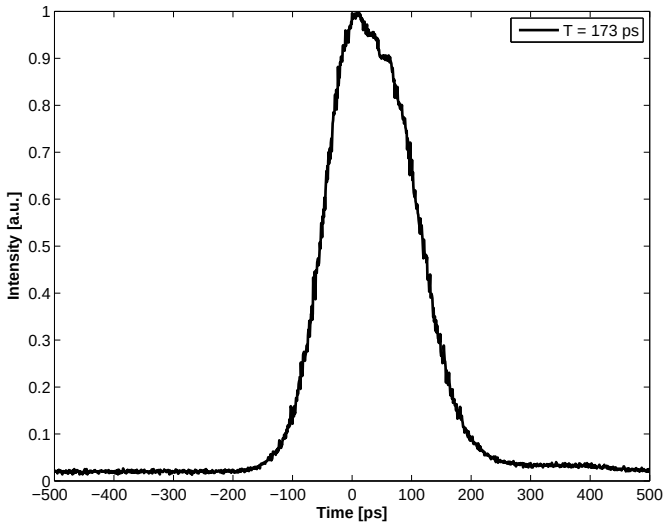


Figure 4.7: The measured pulse width after the pulse compressor (bypassing the stretcher)

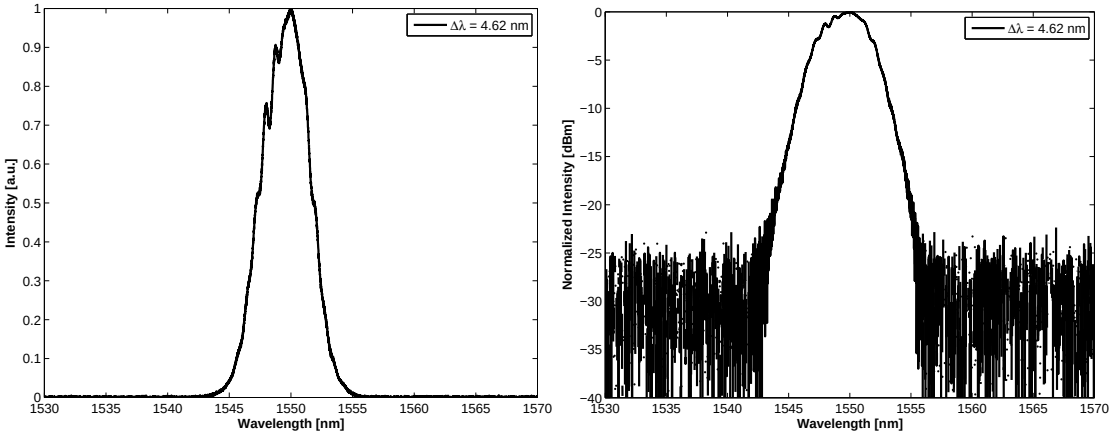
4.1.5 Output of the Entire Setup

Upon testing the stretcher and the compressor individually, the entire setup is tested. The optical circuit shown in Figure 3.1 is built; but instead of the *IPG Photonics Model: EAR-1K-C-W* for AMP2, the *MPB Communications Model: EFA-P22F* fiber amplifier is used as AMP2. This amplifier can easily be switched out for the higher power amplifier initially suggested. The rest of the setup is kept the same.

The average power out of the pre-amp is 1.06 mW. After coupling the stretched pulse into fiber, the average power is measured to be 0.06 mW. After AMP1, the average power is 8.2 dBm or 6.6 mW and after AMP2, the average power is 21 dBm or 126 mW.

Perhaps due to the traffic near the optical table, the stretcher was found to be out of alignment when performing this measurement. It was difficult to reach the

same optical bandwidth (thus the same stretch factor) as presented previously in Figure 4.4. The spectrum for the output pulse from the stretcher is presented in Figure 4.8. The FWHM pulse width of the output from the stretcher is measured to be 79.3 ps as shown in Figure 4.9.



((a)) ((b))

Figure 4.8: The measured spectrum of the pulse after the pulse stretcher for the final trial in 4.8(a) linear scale and 4.8(b) logarithmic scale

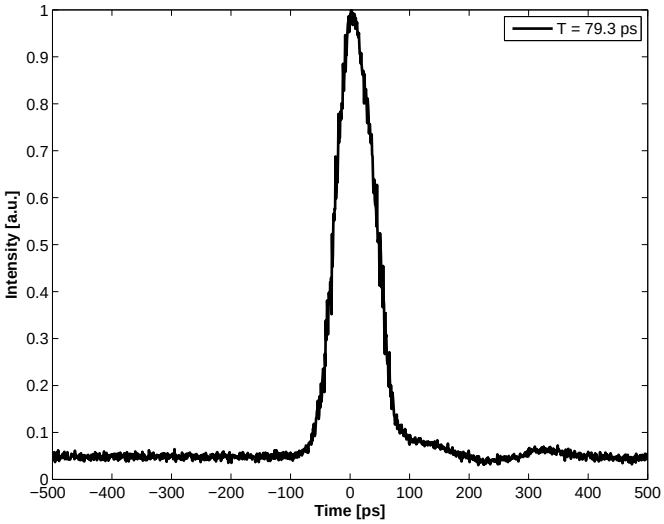
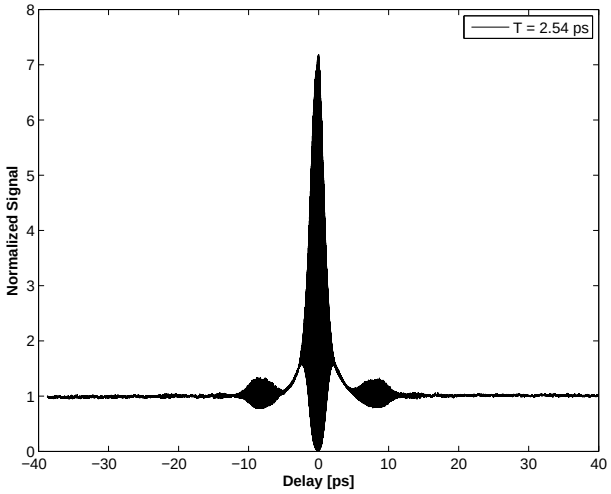


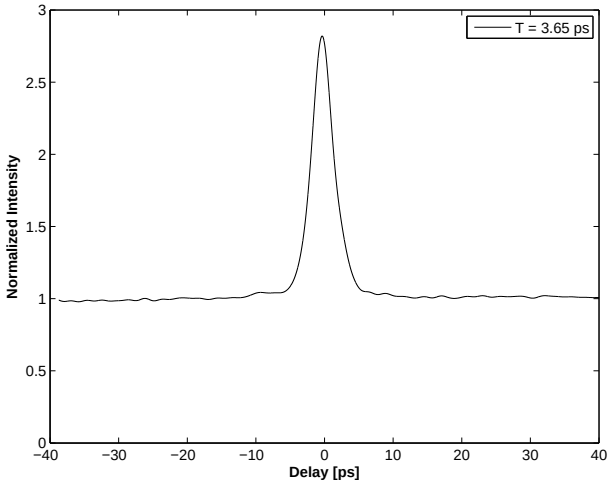
Figure 4.9: The measured pulse width after the pulse stretcher for the final trial

The pulse from AMP2 is passed through the compressor. The distance between

the gratings in the compressor (and hence the GVD) is adjusted until the desired pulse width is obtained[1]. Upon passing the pulse through the compressor, the autocorrelation trace presented in Figure 4.10 is obtained. The intensity pulse-width, after taking the de-convolution factor for a sech^2 pulse (1.543) into account, is: $T_{fwhm} = 3.65/1.543 = 2.37\,\text{ps}$



((a))



((b))

Figure 4.10: The measured autocorrelation trace of the output pulse from the CPA

[1]Due to time constraints, this adjustment was stopped before the final pulse-width objective was achieved.

CHAPTER 5

CONCLUSION

The goal of this project was to amplify an ultra-short 20 MHz pulse with a pulse-width of $T_{fwhm} = 535$ ps (assuming a transform-limited sech^2-intensity pulse-shape) and a pulse-energy of $E_p = 3.36$ pJ to an output pulse of width 400 fs and energy $30 - 50$ nJ. In order to achieve a high pulse quality, it was important to reduce nonlinearities accumulated by the pulse while propagating through fiber. To achieve this objective, a technique called *Chirped Pulse Amplification*, where the pulse is stretched, amplified and compressed, is used to ensure that the pulse peak-power is kept low. It was decided that a grating-pair would be used to stretch and compress the pulse. A program was written in MATLAB to simulate both the propagation of the pulse through the stretcher-compressor system and the fiber. The optical circuit for the CPA was then designed as given in Figure 3.1. The spectrum measurements and autocorrelation or pulse-width traces were obtained for each stage in the optical circuit. Finally, the output pulse autocorrelation, showing an intensity FWHM pulse width of 2.37 ps, was presented.

The output pulse still has some chirp, which can easily be removed by adjusting the distance between the gratings in the compressor. The two-photon current produced by the output beam on a detector can then be observed to see the relative change in the final pulse width (Two-photon current $\propto 1/\tau$), i.e. as the pulse width gets shorter, the two-photon current will increase. Moreover, time can also be spent on removing the spatial chirp on the output of the stretcher and increasing the efficiency of coupling into fiber.

In conclusion, the reader is presented with a well-documented and thorough simulation, detailed design schematics, experimental results and observations of the

grating-pair Chirped Pulse Amplification system. It is the hope that upon reading this thesis, one will be able to replicate this setup or use the simulation code to design a new CPA system to suit his/her input and output pulse requirements.

APPENDIX A

MATLAB CODE FOR THE NONLINEAR SCHRÖDINGER EQUATION SOLVER

The code for the Nonlinear Schrödinger Equation Solver as presented by Travers et al [3] is given below.

Listing A.1: MATLAB code for the NLSE Solver [3]

```
function [Z, AT, AW, W] = gnlse(T, A, w0, gamma, betas, ...
    loss, fr, RT, flength, nsaves)
% Propagate an optical field using the generalised NLSE
% This code integrates Eqs. A.1, A.4 and A.5.
% For usage see the exampe of test Dudley.m (below)
% Written by J.C. Travers, M.H. Frosz and J.M. Dudley (2009)
% Please cite this chapter in any publication using this code.
% Updates to this code are available at www.scgbook.info
n = length(T); dT = T(2)-T(1); % grid parameters
V = 2*pi*(-n/2:n/2-1)'/(n*dT); % frequency grid
B = 0;
for i = 1:length(betas) % Taylor expansion of betas
    B = B + betas(i)/factorial(i+1).*V.^(i+1);
end

if (loss==0)
    alpha = loss/10/log10(exp(1));
else
    alphadb=polyval(loss,V);
    alpha = alphadb/10/log10(exp(1)); % attenuation coefficient
end

% plot(V,alphadb,'b');

L = 1i*B - alpha/2; % linear operator (one*i)
if abs(w0) > eps % if w0>0 then include shock
    gamma = gamma/w0;
    W = V + w0; % for shock W is true freq
else
    W = 1; % set W to 1 when no shock
end

RW = n*ifft(fftshift(RT.')); % frequency domain Raman
L = fftshift(L); W = fftshift(W); % shift to fft space
% === define function to return the RHS of Eq. A.1
    function R = rhs(z, AW)          %AW=F(AT).*exp(-Lz), or A'(z,w) ↩
        in Eq.1.8
        AT = fft(AW.*exp(L*z)); % time domain field, see comments ↩
            above
        IT = abs(AT).^2; % time domain intensity
        if (length(RT)==1)||(abs(fr)<eps)
```

```
40              M=ifft(AT.*IT);
41          else
42              RS = dT*fr*fft(ifft(IT).*RW); % Raman convolution
43              M = ifft(AT.*((1-fr).*IT + RS));% response function
44          end
45          R = 1i*gamma*W.*M.*exp(-L*z); % full RHS of Eq. A.1
46      end
47  % === define function to print ODE integrator status
48      function status = report(z, y, flag) %
49          status = 0;
50          if isempty(flag)
51              fprintf('%05.1f %% complete\n', z/flength*100);
52          end
53      end
54  % === setup and run the ODE integrator
55  Z = linspace(0, flength, nsaves); % select output z points
56  % === set error control options
57  options = odeset('RelTol', 1e-5, 'AbsTol', 1e-12, ...
58      'NormControl', 'on', ...
59      'OutputFcn', @report);
60  [Z, AW] = ode45(@rhs, Z, ifft(A), options); % run integrator, ↩
        integrate region is Z, initial value of AW=ifft(A) at Z=0
61  % === process output of integrator
62  AT = zeros(size(AW(1,:)));
63  for i = 1:length(AW(:,1))
64      AW(i,:) = AW(i,:).*exp(L.'*Z(i)); % change variables
65      AT(i,:) = fft(AW(i,:)); % time domain output
66      AW(i,:) = fftshift(AW(i,:))./dT; % scale
67  end
68  W = V + w0; % the absolute frequency grid
69  end
```

APPENDIX B

MATLAB CODE FOR THE CHIRPED PULSE AMPLIFICATION SYSTEM

The MATLAB code discussed in the previous chapters is given in its entirety below.

The code was written for this Master's Thesis by Ishan Sharma, Cornell University.

Listing B.1: MATLAB code for the Chirped Pulse Amplification System

```
% Simulation for the CPA system using grating-pair stretcher and
% grating-pair compressor. Coded by Ishan Sharma, 2011.

clear all

% ================================================
% Definitions of the time and frequency grids
% ================================================

cpt = cputime;                                  % time since MATLAB started (initial reference time)
n = 2^19;                                       % number of grid points
twidth = 1400;                                  % width of time window [ps]
c = 299792458*1e9/1e12;                         % speed of light [nm/ps]
wavelength = 1550;                              % reference wavelength [nm]
w0 = (2.0*pi*c)/wavelength;                     % reference frequency [rads/ps]
dT = twidth/n;                                  % time interval
T = (-n/2:n/2-1)*dT;                            % time grid (NB: better than T = linspace(-twidth/2, twidth/2, n), since there is now a point at T=0)
V = 2*pi*(-n/2:n/2-1)/(n*dT);                   % frequency grid, ' means transpose
Vabs = V + w0;                                  % absolute frequency grid
WL = 2*pi*c./Vabs;                              % wavelength grid

% ================================================
% Definition of the Input Pulse - Stage 0
% ================================================

reprate = 20*10^6;                              % repetition rate [Hz]
FWHM = 0.3127;                                  % FWHM [ps]
power = 0.88*3.36/FWHM;                         % peak power of input [W] = 0.88*E_pulse/T_fwhm (NB: E_pulse is in pJ)
```

```
29 t0 = FWHM/1.763;                                    % sech duration of ↩
       input [ps]
30 E = sqrt(power)*sech(T/t0);                         % sech input field [↩
       W(1/2)]
31 % t0 = FWHM/(2*log(2))^0.5;                         % guassian ↩
       duration of input [ps]
32 % E = sqrt(power)*exp(-T.^2/(2*t0^2));              % gaussian input ↩
       field [W(1/2)]
33 cx0 = fftshift(fft(E));                             % FT of input field
34 E0 = ifft(ifftshift(cx0));                          % inverse_FT of FT ↩
       of input field (sanity check)
35 %plot(T,E,T,E0)                                     % plots should ↩
       overlap (sanity check)
36
37 % ====== Figures of Merit ====== %
38 E0_energy = sum(abs(E0).^2*dT);                     % should match ↩
       E_pulse entered above (sanity check) [pJ]
39 I0 = abs(E0).^2;                                    % pulse intensity [W↩
       ]
40 t_FWHM0 = fwhm(T,I0);                               % fwhm pulse width [↩
       ps]
41 spec0 = abs(cx0).^2;                                % fwhm pulse width [↩
       ps]
42 w_FWHM0 = fwhm(V,spec0);                            % pulse spectrum
43 lambda_FWHM0 = (w_FWHM0/(2*pi))*wavelength^2/c;     % should match the ↩
       delta_lambda from OSA [nm]
44
45 % ================================================
46 % Definitions of fiber parameters for use with GNLSE code
47 % ================================================
48
49 % ====== Definitions for SMF-28 ====== %
50 radius_mode = 0.5*10.5*10^-6;                       % Mode Field Radius ↩
       @ 1550nm for SMF-28 (Ref: Thorlabs.com) [m]
51 Aeff = pi*radius_mode^2;                            % Effective mode ↩
       area (make this more accurate (read 2.3.1 of Agrawal)) [m^2]
52 n2 = 2.6*10^-20;                                    % nonlinear index ↩
       coefficient, n2, for silica [m^2/W]
53 gamma = 2*pi*n2/(Aeff*wavelength*10^-9);            % nonlinear ↩
       coefficient [1/W/m]
54 loss = 0;                                           % loss [dB/m]
55 [betas] = dispersion_smf(c,w0);                     % call the function ↩
       to extract betas for SMF
56
57 % ====== Raman Response for Silica ====== %
58 fr = 0.18;                                          % fractional Raman ↩
       contribution (for Silica)
59 tau1 = 0.0122; tau2 = 0.032;
60 RT = (tau1^2+tau2^2)/tau1/tau2^2*exp(-T/tau2).*sin(T/tau1);
61 RT(T<0) = 0;                                        % heaviside step ↩
       function
62 RT = RT/trapz(T,RT);                                % normalise RT to ↩
       unit integral
63
64
65 % ================================================
```

```
% Pre-Amplifier (EDFA/EDWA) - Stage 1
% ================================================

avgpower_preamp = 1.06*10^-3;                    % Average power after EDFA [W]
E_pulse_preamp = 10^12*avgpower_preamp/reprate; % Calculate the pulse energy after EDFA [pJ]
P_peak_preamp = 0.88*E_pulse_preamp/t_FWHM0;     % Calculate peak power [W]
gain_preamp = sqrt(P_peak_preamp)/sqrt(power);  % gain for preamp
E1 = E0.*gain_preamp;
cx1 = fftshift(fft(E1));

% ====== Figures of Merit ======%
E1_energy = sum(abs(E1).^2*dT);                  % pulse energy after preamp [pJ]
I1 = abs(E1).^2;                                 % pulse intensity [W]
t_FWHM1 = fwhm(T,I1);                            % fwhm pulse width [ps]
spec1 = abs(cx1).^2;                             % pulse spectrum
w_FWHM1 = fwhm(V,spec1);                         % fwhm spectral width
lambda_FWHM1 = (w_FWHM1/(2*pi))*wavelength^2/c; % delta_lambda [nm]


% ================================================
% Propagation through fiber between preamp and stretcher - Stage 2
% ================================================

flength1 = 30;                                   % fibre length [m]

% ====== Simulation Parameters ======%
nsaves = 20;                                     % number of length steps to save field at

% ====== Propagate Field ======%
[Z, AT2, AW2, W] = gnlse(T, E1, w0, gamma, betas, loss, fr, RT, flength1, nsaves);
E2 = AT2(nsaves,:);
cx2 = AW2(nsaves,:);

% ====== Figures of Merit ======%
E2_energy = sum(abs(E2).^2*dT);                  % pulse energy after propagation through Fiber_1 [pJ]
I2 = abs(E2).^2;                                 % pulse intensity [W]
t_FWHM2 = fwhm(T,I2);                            % fwhm pulse width [ps]
spec2 = abs(cx2).^2;                             % pulse spectrum
w_FWHM2 = fwhm(V,spec2);                         % fwhm spectral width
lambda_FWHM2 = (w_FWHM2/(2*pi))*wavelength^2/c; % delta_lambda [nm]


% ================================================
% Grating-pair stretcher - Stage 3
```

```
110 % ================================================
111
112 [GVD1, TOD1] = gratingpair(1, wavelength, 1.5*lambda_FWHM1, 1000, 0,↩
        30, 16, 16, 50.8-9)
113 cx3=exp(((1i*(1/2)*GVD1).*V.^2)+((1i*(1/6)*TOD1).*V.^3)).*cx1;
114 E3 = ifft(ifftshift(cx3));
115
116 % ====== Figures of Merit ====== %
117 E3_energy = sum(abs(E3).^2*dT);                  % pulse energy after↩
        stretcher [pJ]
118 I3 = abs(E3).^2;                                 % pulse intensity [W↩
       ]
119 t_FWHM3 = fwhm(T,I3);                           % fwhm pulse width [↩
       ps]
120 spec3 = abs(cx3).^2;                            % pulse spectrum
121 w_FWHM3 = fwhm(V,spec3);                        % fwhm spectral ↩
       width
122 lambda_FWHM3 = (w_FWHM3/(2*pi))*wavelength^2/c; % delta_lambda [nm]
123
124
125 % ================================================
126 % Amplifier (Cascaded EDFAs/EDWAs) - Stage 4
127 % ================================================
128
129 E_pulse_amp = 30*10^3;                           % Desired pulse ↩
       energy after EDFA [pJ]
130 P_peak_amp = 0.88*E_pulse_amp/t_FWHM3;           % Calculate peak ↩
       power [W]
131 gain_amp = 24.1;                                % gain for amp
132 E4 = E3.*gain_amp;
133 cx4 = fftshift(fft(E4));
134
135 % ====== Figures of Merit ====== %
136 E4_energy = sum(abs(E4).^2*dT);                  % pulse energy after↩
        amplifier [pJ]
137 I4 = abs(E4).^2;                                 % pulse intensity [W↩
       ]
138 t_FWHM4 = fwhm(T,I4);                           % fwhm pulse width [↩
       ps]
139 spec4 = abs(cx4).^2;                            % pulse spectrum
140 w_FWHM4 = fwhm(V,spec4);                        % fwhm spectral ↩
       width
141 lambda_FWHM4 = (w_FWHM4/(2*pi))*wavelength^2/c; % delta_lambda [nm]
142
143
144 % ================================================
145 % Propagation through fiber between EDFA and Compressor - Stage 5
146 % ================================================
147
148 flength5 = 30;                                   % fibre length [m]
149
150 % ====== Simulation Parameters ====== %
151 nsaves = 20;                                     % number of length ↩
       steps to save field at
152
153 % ====== Propagate Field ====== %
```

```
154 [Z, AT5, AW5, W] = gnlse(T, E4, w0, gamma, betas, loss, fr, RT, ↩
      flength5, nsaves);
155 E5 = AT5(nsaves,:);
156 cx5 = AW5(nsaves,:);
157
158 %====== Figures of Merit ======%
159 E5_energy = sum(abs(E5).^2*dT);                    % pulse energy after↩
      propagation through Fiber_2 [pJ]
160 I5 = abs(E5).^2;
161 t_FWHM5 = fwhm(T,I5);
162 spec5 = abs(cx5).^2;
163 w_FWHM5 = fwhm(V,spec5);
164 lambda_FWHM5 = (w_FWHM5/(2*pi))*wavelength^2/c;
165
166
167 %================================================
168 % Grating-pair compressor - Stage 6
169 %================================================
170
171 [GVD2, TOD2] = gratingpair(0, wavelength, lambda_FWHM4, 1200, ↩
      1.636877881749541e+001, 0, 0,0, 74.4)
172 cx6=exp(((1i*(1/2)*GVD2).*V.^2)+((1i*(1/6)*TOD2).*V.^3)).*cx4;
173 E6 = ifft(ifftshift(cx6));
174
175 %====== Figures of Merit ======%
176 E6_energy = sum(abs(E6).^2*dT);                    % pulse energy after↩
      compressor [pJ]
177 I6 = abs(E6).^2;
178 t_FWHM6 = fwhm(T,I6);
179 spec6 = abs(cx6).^2;
180 w_FWHM6 = fwhm(V,spec6);
181 lambda_FWHM6 = (w_FWHM6/(2*pi))*wavelength^2/c;
```

APPENDIX C

MATLAB CODE FOR THE *FWHM* FUNCTION

The MATLAB code for the function *fwhm*, called several times by the CPA program given in Appendix B, is given below. The source code for this function, authored by Patrick Egan [4], was obtained from the MATLAB Central File Exchange server and is presented below as per the instructions in the BSD License.

Listing C.1: MATLAB code for the *fwhm* Function

```
function width = fwhm(x,y)

% function width = fwhm(x,y)
%
% Full-Width at Half-Maximum (FWHM) of the waveform y(x)
% and its polarity.
% The FWHM result in 'width' will be in units of 'x'
%
%
% Rev 1.2, April 2006 (Patrick Egan)


y = y / max(y);
N = length(y);
lev50 = 0.5;
if y(1) < lev50                 % find index of center (max or min) of pulse
    [garbage,centerindex]=max(y);
    Pol = +1;
    %disp('Pulse Polarity = Positive')
else
    [garbage,centerindex]=min(y);
    Pol = -1;
    %disp('Pulse Polarity = Negative')
end
i = 2;
while sign(y(i)-lev50) == sign(y(i-1)-lev50)
    i = i+1;
end                                 %first crossing is between v(i-1) & v(i)
interp = (lev50-y(i-1)) / (y(i)-y(i-1));
tlead = x(i-1) + interp*(x(i)-x(i-1));
i = centerindex+1;                  %start search for next crossing at center
while ((sign(y(i)-lev50) == sign(y(i-1)-lev50)) & (i <= N-1))
    i = i+1;
end
if i ~= N
    Ptype = 1;
    %disp('Pulse is Impulse or Rectangular with 2 edges')
```

```
38      interp = (lev50-y(i-1)) / (y(i)-y(i-1));
39      ttrail = x(i-1) + interp*(x(i)-x(i-1));
40      width = ttrail - tlead;
41  else
42      Ptype = 2;
43      %disp('Step-Like Pulse, no second edge')
44      ttrail = NaN;
45      width = NaN;
46  end
47
48
49  % Copyright (c) 2009, Patrick Egan All rights reserved.
50  %
51  % Redistribution and use in source and binary forms, with or without
52  % modification, are permitted provided that the following conditions
53  % are met:
54  %
55  %     * Redistributions of source code must retain the above
56  %       copyright notice, this list of conditions and the
57  %       following disclaimer.
58  %     * Redistributions in binary form must reproduce the above
59  %       copyright notice, this list of conditions and the following
60  %       disclaimer in the documentation and/or other materials
61  %       provided with the distribution.
62  %
63  % THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND ↩
      CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, ↩
      INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF ↩
      MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE ↩
      DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR ↩
      CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, ↩
      SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT↩
       LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS ↩
      OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER ↩
      CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, ↩
      STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ↩
      ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ↩
      ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

APPENDIX D

MATLAB CODE FOR AN AUTOCORRELATION FUNCTION

The MATLAB code for a function to obtain the autocorrelation trace with the time-domain electric field as the input is given below. This code was authored by Ji Cheng, Cornell University.

Listing D.1: MATLAB code for an Autocorrelation Function

```
function [output]=StrictAutocorrelation(Eft,Ntime,time)

clight=3e5;                                    %unit nm/ps

t=1:Ntime;
dt=time/Ntime;
T=(t-Ntime/2-1)*dt;

w=1:Ntime;
W=2*pi*(w-Ntime/2-1)/time;


input_l=1550;                                   % unit nm
input_w=2*pi*clight/input_l;                    % unit ps-1
lambdas=2*pi*clight./(W+input_w);
Eft=Eft(t).*exp(-i*input_w*T);
%figure; plot(T,Eft.*conj(Eft));

Iat=Eft.*conj(Eft);
%figure;plot(T,Iat);
c201=xcorr(Eft.^2);
c20=c201+conj(c201);
c211=xcorr(Iat.*Eft, Eft);
c21=c211+conj(c211);
c221=xcorr(conj(Eft), Iat.*conj(Eft));
c22=c221+conj(c221);
c23=xcorr(Iat);
c2=2*sum(Iat.*Iat)+2*c21+2*c22+4*c23+c20;

C1=xcorr(Iat);
C11=sum(Iat.^2);
Cint=2*C11+4*C1;

output=c2;

Start=Ntime*1/4;
End=Ntime*7/4-1;
indext=1:End-Start+1;

%figure; plot(indext*dt,3*Cint(Start:End)/max(Cint(Start:End)),'k');
% figure; plot(indext*dt,8*c2(Start:End)/max(c2(Start:End)),'k');
```

APPENDIX E

MATLAB CODE TO CALCULATE GRATING-PAIR PARAMETERS GIVEN A GVD VALUE

The MATLAB code for a program to calculate the grating-pair parameters including the grating separation and angle of incidence is given below. This code was authored by Ishan Sharma, Cornell University.

Listing E.1: MATLAB code to Calculate Grating-pair Parameters given GVD

```
% Coded by Ishan Sharma, 2011

lambda = 1550*10^-9;            % center wavelength [m]
sgroove = 1200;                          % groove spacing [lines/mm]
c = 2.99792458*10^8;                % speed of light [m/s]
d = 1/(sgroove*10^3);               % line-width [m]
thetalittrow = asin(lambda/(2*d)); % calculation of the littrow angle [rads]
thetalittrow_deg = thetalittrow*180/pi
% thetai = thetalittrow;           % working at the Littrow angle [rads]
% thetai_deg = 50:0.001:90;
thetai_deg = 68.4+6
thetai = thetai_deg.*(pi/180);   % working at user-input incidence angle [rads]
thetad = asin((lambda/d)-sin(thetai));
thetad_deg = thetad/(pi/180)
GVD = -3.591320891731296e+001/(10^12)^2;

L_eff_cm = 100./(-(1/GVD)*(2*lambda^3/(2*pi*c^2*d^2))*(1-((lambda/d)-sin(thetai)).^2).^-(3/2))
length_cm = L_eff_cm/cos(thetad)
collimator_cm = 5/tan(thetai-thetad)
% plot(thetai_deg,L_eff)
% hold on
% plot(thetad_deg,L_eff)
% hold off
```

BIBLIOGRAPHY

[1] A. Galvanauskas, "Mode-scalable fiber-based chirped pulse amplification systems," *IEEE Journal on Selected Topics in Quantum Electronics*, vol. 7, p. 504, July/August 2001.

[2] G. P. Agrawal, *Nonlinear Fiber Optics*. Academic Press, fourth ed., 1865.

[3] C. Travers, M. H. Frosz, and J. M. Dudley, "Nonlinear fiber optics overview," in *Supercontinuum Generation in Optical Fibers*, ch. 3, Cambridge University Press, 2010.

[4] P. Egan, "fwhm - to calculate the full-width at half-maximum of an input," April 2006.

[5] J. van Howe, G. Zhu, and C. Xu, "Compensation of self-phase modulation in fiber-based chirped-pulse amplification systems," *Optics Letters*, vol. 31, p. 1756, June 2006.

[6] O. Martinez, "3000 times grating compressor with positive group velocity dispersion: Application to fiber compensation in 1.3-1.6 μm region," *Quantum Electronics, IEEE Journal of*, vol. 23, p. 59, January 1987.

[7] O. Martinez, "Design of high-power ultrashort pulse amplifiers by expansion and recompression," *Quantum Electronics, IEEE Journal of*, vol. 23, p. 1385, August 1987.

[8] S. Backus, C. G. Durfee III, M. M. Murnane, and H. C. Kapteyn, "High power ultrafast lasers," *Review of Scientific Instruments*, vol. 69, p. 1207, March 1998.