Module 3 – Numerical Pulse Propagation in Fibers

Dr. E. M. Wright
Professor, College of Optics,
University of Arizona

Dr. E. M. Wright is a Professor of Optical Sciences and Physic sin the University of Arizona. Research activity within Dr. Wright's group is centered on the theory and simulation of propagation in nonlinear optical media including optical fibers, integrated optics, and bulk media. Active areas of current research include nonlinear propagation of light strings in the atmosphere, and the propagation of novel laser fields for applications in optical manipulation and trapping of particles, and quantum nonlinear optics.

Email: ewan.wright@optics.arizona.edu

Introduction

Module 2 drew together the material needed to describe linear pulse propagation in a fiber in a formulation that will allow extension to nonlinear fiber propagation in modules 4 and 5. In the current module 3 we shall develop a numerical approach to linear pulse propagation in fibers. This will serve as both a simulation tool and an aid to understanding. For this module it is assumed you have a working knowledge of Matlab and are familiar with the basic commands and plotting, and also have access to a copy of Matlab.

The learning outcomes for this module include

- The student will be conversant with the Fourier representation for a pulse propagating in a fiber and the idea of the pulse spectrum.
- The student will be conversant with the algorithm for performing linear pulse propagation in a fiber using Fourier transforms.
- The student will be able to perform discrete Fourier transforms used in the numerical evaluation of Fourier transforms.
- The student will be able to convert the propagation algorithm into discrete form suitable for numerical evaluation in Matlab.
- The student will be conversant with the Matlab code for performing linear pulse propagation in fibers including the examples of Gaussian, super-Gaussian, and hyperbolic-secant pulses

1 The authors would like to acknowledge support from the National Science Foundation through CIAN NSF ERC under grant #EEC-0812072

3.1 Fourier representation of pulses and pulse spectra

So far we have described the evolution of the pulse in space (z) and time (t or T) using the *temporal field envelope* $A(z,T)=\sqrt{P_0}U(z,T)$ and the slowly varying envelope **Equation 2.24** or **2.26**. In the realm of linear optics the distinction between the formulations in terms of A or U is irrelevant since the propagation is independent of the initial power P_0 and we can always choose units such that $P_0=1$ in which case the two are identical. Here we shall concentrate on $A(z,T)$ and the envelope **Equation. 2.26** in the retarded frame for zero absorption which we write in the form

$$\frac{\partial A}{\partial z}+\frac{i\beta_2}{2}\frac{\partial^2 A}{\partial T^2}=0\,.$$

(Equation 3.1)

We wish to solve **Equation. 3.1** subject to the initial condition $A(0,T)$ at $z=0$ as the input pulse to the fiber. In order to solve **Equation. 3.1** numerically it proves useful to introduce the Fourier representation of the pulse envelope

$$A(z,T)=\int_{-\infty}^{\infty}d\Omega\,\tilde{A}(z,\Omega)e^{-i\Omega T}.$$

(Equation 3.2)

Physically a pulse may be represented as a wavepacket of monochromatic or single frequency fields with temporal variation $\exp(-i\Omega t)$, and the *inverse Fourier transform* in **Equation. 3.2** is a means to synthesize the field envelope into its constituent frequencies Ω, each frequency being weighted by the *spectral field envelope* $\tilde{A}(z,\Omega)$. Recalling that the vector electric field in **Equation. 2.16** involves the product of the field envelope times the carrier plane-wave $e^{i(\beta_0 z-\omega_0 t)}$, including the carrier frequency ω_0 the Fourier transform synthesizes the pulse into monochromatic fields with frequencies $\omega=\omega_0+\Omega$. We thus identify $\Omega=\omega-\omega_0$ as the frequency relative to the carrier frequency, and $\tilde{A}(z,\Omega)$ represents the pulse envelope in space (z) and frequency (Ω).

The *Fourier transform* can be used to obtain the *spectral field envelope* using the integral

$$\tilde{A}(z,\Omega)=\frac{1}{2\pi}\int_{-\infty}^{\infty}dT\,A(z,T)e^{i\Omega T}\,,$$

(Equation 3.3)

and together $A(z,T)$ and $\tilde{A}(z,\Omega)$ form a *Fourier transform pair*. Physically $|A(z,T)|^2$ represents the temporal variation of power within the pulse for a given propagation distance z, whereas $|\tilde{A}(z,\Omega)|^2$ is the *power spectrum* that represents the distribution of power amongst the

The authors would like to acknowledge support from the National Science Foundation through CIAN NSF ERC under grant #EEC-0812072

various frequency components that make up the pulse for a given propagation distance z. The power spectrum or simply spectrum will play an important role in the remaining modules.

For additional reading on the material covered in this section see Sec. 11.2 of Ref. [1] and Sec. 12.0 of Ref. [2].

3.2 Linear pulse propagation using Fourier methods

Our next goal is to obtain a solution of **Equation. 3.1** that we can use for numerical simulations. As a first attempt we may write **Equation. 3.1** in the operator form

$$\frac{\partial A(z,T)}{\partial z} = \hat{D}A(z,T) \ ,$$

(Equation 3.4)

where the *dispersion operator* is $\hat{D} = -\frac{i\beta_2}{2}\frac{\partial^2}{\partial t^2}$, with solution

$$A(z,T) = e^{\hat{D}z}A(0,T) \ .$$

(Equation 3.5)

This is a formal solution but is not of utility for numerical evaluation. In this formal solution the operator $e^{\hat{D}z}$ acts or operates on the initial pulse $A(0,T)$ to produce the pulse $A(z,T)$ after a propagation distance z. We shall utilize this operator notation in modules 4 and 5.
To find a useful solution we substitute the Fourier integral **Equation. 3.2** into the slowly varying envelope **Equation. 3.1** and rearrange to obtain

$$\int_{-\infty}^{\infty} d\Omega e^{-i\Omega T}\left[\frac{\partial \tilde{A}(z,\Omega)}{\partial z} - \frac{i\beta_2\Omega^2}{2}\tilde{A}(z,\Omega)\right] = 0 \ .$$

(Equation 3.6)

In order for this to be satisfied the term in square brackets will have to vanish yielding the equation

$$\frac{\partial \tilde{A}(z,\Omega)}{\partial z} = \frac{i\beta_2\Omega^2}{2}\tilde{A}(z,\Omega) \ ,$$

(Equation 3.7)

with the *exact solution*

$$\tilde{A}(z,\Omega) = e^{i\beta_2\Omega^2 z/2}\tilde{A}(0,\Omega) \ .$$

(Equation 3.8)

The authors would like to acknowledge support from the National Science Foundation through CIAN NSF ERC under grant #EEC-0812072

The propagation of the spectral field envelope thus involves a simple multiplication by a phase factor, and this is the reason to work in the frequency domain

We now have the basis for an algorithm for performing pulse propagation as follows

1. Calculate the spectral field envelope $\tilde{A}(0,\Omega)$ from the initial pulse $A(0,T)$ at $z=0$ using the Fourier transform **Equation. 3.3**

$$\tilde{A}(0,\Omega)=\frac{1}{2\pi}\int_{-\infty}^{\infty}dT\,A(0,T)e^{i\Omega T}.$$

(Equation 3.9)

2. Propagate the spectral field envelope $A(z,\Omega)$ for a desired propagation distance z using **Equation. (3.8)**

$$\tilde{A}(z,\Omega)=e^{i\beta_2\Omega^2 z/2}\tilde{A}(0,\Omega)\ .$$

(Equation 3.10)

3. Calculate the temporal field envelope $A(z,T)$ from $\tilde{A}(z,\Omega)$ using the inverse Fourier transform in **Equation. 3.1**

$$A(z,T)=\int_{-\infty}^{\infty}d\Omega\,\tilde{A}(z,\Omega)e^{-i\Omega T}=\int_{-\infty}^{\infty}d\Omega\,e^{i\beta_2\Omega^2 z/2}\tilde{A}(0,\Omega)e^{-i\Omega T}.$$

(Equation 3.11)

We now make some comments regarding this algorithm:

- This algorithm requires one Fourier transform and one inverse Fourier transform per calculation in **Equations. 3.9 and 3.11** in steps 1 and 3, and as we shall discuss these can be done very rapidly using the fast Fourier transform in Matlab.
- The operation in **Equation. 3.10** in step 2 is very rapid as it is a simple multiplication.
- Note that the algorithm is the same and will take the same computational time irrespective of the propagation distance z, so it is the same for short and long propagation distances.
- The propagation algorithm is identical in content to the formal solution given in **Equation. 3.5** but provides a means of implementing the formal solution.

We also note that using **Equation. 3.8** we obtain

$$|\tilde{A}(z,\Omega)|^2=|\tilde{A}(0,\Omega)|^2\ ,$$

(Equation 3.12)

so the pulse spectrum is invariant under linear propagation. That is, although the power profile $|A(z,T)|^2$ may greatly reshape upon propagation through a fiber the corresponding pulse

The authors would like to acknowledge support from the National Science Foundation through CIAN NSF ERC under grant #EEC-0812072

spectrum $|\tilde{A}(z,\Omega)|^2$ will remain the same (it may decay due to linear absorption but the spectral profile will remain the same).

In the next two sections we shall describe the numerical implementation of the above algorithm in Matlab followed by some numerical examples of fiber propagation.

3.3 Numerical propagation using MATLAB

In order to numerically implement the algorithm in the last section we need to translate the equations involved into a form suitable for numerical calculation. As a first step we need to set up a time grid to represent the continuous time variable T. To do this we consider a time window of temporal width T_{max} and assign N discrete time values

$$T_j = -\frac{T_{max}}{2} + (j-1)\Delta T, \quad j = 1,2,\ldots N$$

(Equation 3.13)

where $\Delta T = T_{max}/(N-1)$. The time window is thus centered on $T = 0$ and varies from $T_1 = -T_{max}/2$ to $T_N = T_{max}/2$. For each point on the grid we assign the corresponding value of the temporal field envelope value

$$A_j(z) \equiv A(z,T_j), \quad j = 1,2,\ldots N$$

(Equation 3.14)

so each point in the time window is labeled by the subscript j for a given value of z. The hope is that the time window T_{max} and the number of points N can be chosen such that the continuous field envelope may be well approximated by the discrete formulation of the problem on the grid.

Using the above discretization scheme the Fourier transform in **Equation. 3.3** may be approximated as a sum in the form

$$\tilde{A}(z,\Omega) = \left(\frac{\Delta T}{2\pi}\right)\sum_{j=1}^{N} A_j(z)e^{i\Omega T_j},$$

(Equation 3.15)

which is an example of a *discrete Fourier transform* (DFT). In **Equation. 3.15** we note that the frequency variable is still continuous and can be evaluated for any value of Ω.

To precede it is necessary to introduce a frequency grid to represent the continuous frequency variable Ω. The evaluation of the sum in **Equation. 3.15** is greatly facilitated by a particular choice of frequency grid given by

The authors would like to acknowledge support from the National Science Foundation through CIAN NSF ERC under grant #EEC-0812072

$$\Omega_m = -\frac{\Omega_{max}}{2} + (m-1)\Delta\Omega, \quad m = 1, 2, \ldots N,$$

(Equation 3.16)

where $\Omega_{max} = 2\pi / \Delta T$, and $\Delta\Omega = \Omega_{max} / (N-1) = 2\pi / T_{max}$. This choice of frequency grid is in keeping with the *Nyquist sampling theorem* according to which for a time grid of spacing ΔT the highest frequency that can be resolved is $\Omega_{max} / 2 = \pi / \Delta T$. The frequency window so constructed is centered on $\Omega = 0$ and varies from $\Omega_1 = -\Omega_{max} / 2$ to $\Omega_N = \Omega_{max} / 2$. Note that the parameters of the frequency grid Ω_{max} and $\Delta\Omega$ are determined by the parameters chosen for the temporal grid T_{max} and N, so the time and frequency grids are intimately related. For each point on the frequency grid we assign the corresponding value of the spectral field envelope

$$\tilde{A}_m(z) \equiv \tilde{A}(z, \Omega_m), \quad m = 1, 2, \ldots N \ ,$$

(Equation 3.17)

so that each point in the frequency window is labeled by the subscript m for a given value of z. With this choice of frequency grid the Fourier transform in **Equation. 3.15** becomes

$$\tilde{A}_m(z) = \left(\frac{\Delta T}{2\pi}\right) \sum_{j=1}^{N} A_j(z) e^{i\Omega_m T_j},$$

(Equation 3.18)

and it is left as an exercise for you to verify that the inverse Fourier transform in **Equation. 3.2** becomes

$$A_j(z) = \Delta\Omega \sum_{m=1}^{N} \tilde{A}_m(z) e^{-i\Omega_m T_j} \ .$$

(Equation 3.19)

Equations 3.18 and 3.19 are the discrete approximation to the Fourier transform pair in **Eqs. 3.2 and 3.3.** We note that if we perform a Fourier transform followed by an inverse Fourier transform the result will involve the product $(\Delta\Omega\Delta T) / 2\pi$ of the prefactors appearing in **Eqs. 3.18 and 3.19.** Using the relations $\Delta\Omega = 2\pi / T_{max} = 2\pi / (N\Delta T)$ we find $(\Delta\Omega\Delta T) / 2\pi = 1 / N$. The overall factor $1 / N$ can then be lumped into one sums rather than have prefactors for both sums, and we shall do this below.

To appreciate why the above choice of temporal and frequency grids is special consider the product $\Omega_m T_j$ appearing in the exponentials in Eqs. 3.18 and 3.19 with $T_j = (j-1)\Delta T$, $\Omega_m = (m-1)\Delta\Omega$. (Here we have dropped the constant terms $-T_{max} / 2$ and $-\Omega_{max} / 2$ for simplicity in notation: The $-T_{max} / 2$ term is a simple shift in the T-axis, and the $-\Omega_{max} / 2$ term can be seen as a simple shift in the carrier frequency $\omega_0 \to \omega_0 + \Omega_{max} / 2$). Then we find

The authors would like to acknowledge support from the National Science Foundation through CIAN NSF ERC under grant #EEC-0812072

$$\Omega_m T_j = (m-1)(j-1)\Delta\Omega\Delta T = \frac{2\pi}{N}(m-1)(j-1)\ ,$$

(Equation 3.20)

where we used $\Delta\Omega = 2\pi / T_{max} = 2\pi / (N\Delta T)$ to find $\Delta\Omega\Delta T = 2\pi / N$. Substituting **Equation. 3.20 into Eqs. 3.18 and 3.19** yields the *discrete Fourier transform pair*

$$\tilde{A}_m(z) = \sum_{j=1}^{N} A_j(z) e^{2\pi i(j-1)(m-1)/N},$$

(Equation 3.21)

$$A_j(z) = \frac{1}{N}\sum_{m=1}^{N} \tilde{A}_m(z) e^{-2\pi i(j-1)(m-1)/N}\ ,$$

(Equation 3.22)

where we note the prefactor $1/N$ in **Equation. 3.22** instead of the two individual prefactors. Sums such as these can be carried out very rapidly using the fast Fourier transform (FFT) algorithm in Matlab.

Equations 3.21 and 3.22 will allow us to numerically evaluate the Fourier transform and inverse transform needed for steps 1 and 3 of the propagation algorithm. We also need perform the phase transformation in **Equation. 3.10** of step 2. The discrete form of **Equation. 3.10** is

$$\tilde{A}_m(z) = e^{i\beta_2 \Omega_m{}^2 z/2} \tilde{A}_m(0)\ ,$$

(Equation 3.23)

which can be applied at each point labeled m on the frequency grid.

For concreteness let us repeat the propagation algorithm in discrete form:

1. Calculate the spectral field envelope $\tilde{A}_m(0)$ on the frequency grid from the initial pulse $A_j(0)$ on the time grid at $z = 0$ using the discrete Fourier transform in **Equation. 3.21**

$$\tilde{A}_m(0) = \sum_{j=1}^{N} A_j(0) e^{2\pi i(j-1)(m-1)/N},$$

(Equation 3.24)

2. Propagate the spectral field envelope $\tilde{A}_m(z)$ for a desired propagation distance z using **Equation. (3.23)**

$$\tilde{A}_m(z) = e^{i\beta_2 \Omega_m{}^2 z/2} \tilde{A}_m(0)\ ,$$

(Equation 3.25)

The authors would like to acknowledge support from the National Science Foundation through CIAN NSF ERC under grant #EEC-0812072

3. Calculate the field envelope $A_j(z)$ from $\tilde{A}(z,\Omega)$ using the inverse Fourier transform in **Equation. 3.22**

$$A_j(z) = \frac{1}{N}\sum_{m=1}^{N} \tilde{A}_m(z) e^{-2\pi i(j-1)(m-1)/N} \quad .$$

(Equation 3.26)

The short Matlab code LinearPulseProp.m on the next page performs linear pulse propagation according to this algorithm. The goal is for you to back engineer how this code is constructed to gain an appreciation of how numerical pulse propagation works, though how to choose numerical parameters such as T_{max} and N takes a bit more time and experience. If you have a lot of Matlab knowledge you may want to rewrite the code to your liking once you have figured it out, or if you are a beginner you may use this code to learn some Matlab.

We now turn to a description of how the code is constructed. After the first line specifying the function name LinearPulseProp the code starts at the top with specification of the input parameters $T_{max}, N, T_0, \beta_2, z$ and C in the units indicated. For example, T0 $\equiv T_0$ is in ps, and beta2 $\equiv \beta_2$ is in ps^2/m, and N is normally taken as a power of N=2^p with p a positive integer, eg. 128, 256, 512, etc. The code as written is specifically for propagation of a Gaussian pulse of pulse width T_0 and chirp parameter C in an optical fiber with GVD parameter β_2 and length z in km.

Moving down the code the next task is to set up the time and frequency grids. Here the function linspace(0,N-1,N) is used to set up a one-dimensional array v=[0 1 …. N-1] of length N. You may find documentation for each Matlab function by using the help command in the Matlab command window, eg. >help linspace. This is next used to set up the one-dimensional array T containing the time grid. You should check for yourself that T set up in this way does correspond to the time grid. The frequency grid is a bit more involved since for reasons of computational efficiency Matlab does not store the array of frequencies precisely as specified in **Equation. 3.16**. In particular, zero frequency $\Omega_m = 0$ corresponds to $m = 1$, positive frequencies $0 < \Omega_m < \Omega_{max}/2$ correspond to $2 \le m \le N/2$, while negative frequencies $-\Omega_{max}/2 < \Omega_m < 0$ correspond to $(N/2+2) \le m \le N$, the case $m = (N/2+1)$ corresponding to *both* $\Omega_{m=N/2} = \pm\Omega_{max}/2$. You should check for yourself that the three lines in the code that are related to setting up the frequency grid do so in the correct manner, and for this it is best to take a small number, say N=8, and make sure it works.

Once you have the time and frequency grids worked out the rest is relatively easy. There is a line marked 'Gaussian input pulse' and that is used to set up the initial pulse as an array A on the time grid. You should compare this line against the Gaussian beam formula 2.32 including chirp. STEP 1 of the propagation algorithm is then implemented using the single line of code At0=fft(A) to obtain the Fourier transform At0 of the input pulse A (you may get documentation on fft.m using the help command in Matlab). The phase factor is applied in frequency space to

The authors would like to acknowledge support from the National Science Foundation through CIAN NSF ERC under grant #EEC-0812072

At0 to yield Atilde in STEP 2 using **Equation. 3.25** as you should verify in the indicated line of the code.

```
function LinearPulseProp
%
%       Input parameters
%
Tmax = 20.00;          % temporal grid size in ps
N = 0128;              % number of grid points
T0 = 1.0;              % pulse duration in ps
beta2 = +25;           % GVD parameter in ps^2/km
z = 0.07;              % propagation distance in km
C = +0.0;              % chirp parameter
%
%      Set up time and frequency grids
%
v = linspace(0,N-1,N);
dT = Tmax/N;
T = -Tmax/2 + v*dT;              % Time grid
dOmega = 2*pi/Tmax;
%
p = find(v > floor(N/2));      % These three lines set up the frequency
v(p) = v(p)-N;                 % grid in the storage form employed by
Omega = v*dOmega;              % Matlab.
%
A = exp(-((1+1i*C)/2)*(T/T0).^2);         % Gaussian input pulse
A0 = A;                                   % copy the initial Gaussian
%
At0 = fft(A);                              % STEP 1 - FT initial pulse
Atilde = At0.*exp(1i*beta2*Omega.^2*z/2);  % STEP 2 - phase factor
A = ifft(Atilde);                          % STEP 3 - inverse FT
%
%            Plot the output pulse
%
plot(T,abs(A).^2,T,abs(A0).^2,'--');
set(gca,'FontSize',15);
xlabel('T (ps)');
ylabel('|A(z,T)|^2');
title('Initial pulse (dash) final pulse (solid)');
```

To run this Matlab code one simple types

>LinearPulseProp;

at the Matlab commend line and presses return.

Finally, the inverse Fourier transform in STEP 3 is implemented using the single line of code A=ifft(Atilde) to yield the propagated field A at distance z.

The authors would like to acknowledge support from the National Science Foundation through CIAN NSF ERC under grant #EEC-0812072

The last part of the code involves plotting the propagated pulse power profile $|A(z,T)|^2$ versus T (solid line) along with the initial pulse power profile (dashed line).

Once you have back engineered this code you will have the basic idea of how to perform linear pulse propagation in optical fibers. It is possible to extend the basic code to incorporate many generalizations including linear absorption and higher-order dispersion. The idea here is to expose you to the basic ideas of numerical pulse propagation using Matlab so that we can use such codes as a learning aid. We shall build upon this basic code in the coming modules.

For additional reading on the material covered in this section see 12.1 and 12.2. of Reference [2].

3.4 Examples

In this section we shall explore three examples of linear pulse propagation in an optical fiber using the code LinearPulseProp.m, namely Gaussian, super-Gaussian, and hyperbolic-secant pulses. We start with the example of a Gaussian input pulse for which we have the exact solution from module 2 section 2.6, and you may check the code against this exact solution. (Note that for linear pulse propagation with $P_0 = 1$ then $U(z,T) \equiv A(z,T)$, so we may use the solution from Sec. 2.6 here). The super-Gaussian and hyperbolic-secant pulses do not allow analytic solutions and are therefore good examples of where the numerical approach is indispensable.

Let us start with the case of the Gaussian pulse as this will allow us to discuss the choice of numerical parameters by example. In particular we consider an initial chirped Gaussian pulse

$$U(0,T) = \exp\left(-\frac{(1+iC)}{2}\frac{T^2}{T_0^2}\right) = \exp\left(-\frac{T^2}{2T_0^2} + i\varphi(T)\right),$$

(Equation 3.27)

with Gaussian pulse width T_0 and chirp parameter C. In the code LinearPulseProp.m the specific parameters are $T0 \equiv T_0 = 1$ ps and $C = 0$, $\beta_2 = 25$ ps^2/km, yielding a dispersion length $L_D = T_0^2 / |\beta_2| = 0.04$ km, and we shall use these values throughout unless otherwise stated. Clearly we need the time window $T_{max} > T_0$, and as a rule of thumb one typically takes $T_{max} = 10T_f$ as a starting point, where T_f is the feature one is trying to model. Here the feature we are modeling is the Gaussian pulse peak $T_f \approx 2T_0$, so $T_{max} = 20T_0 = 20$ ps is a good starting point. We also require $\Delta T = T_{max} / N << T_f$ so that there are many grid point over the width of the feature so that it is well sampled, giving $N >> T_{max} / T_f$, or $N >> 10$. Here we chose $N = 128 = 2^7$ as a starting point.
The code is initially set up for a propagation distance $z = \sqrt{3}L_D = 0.07$ km. If you run the code you will produce the figure below

The authors would like to acknowledge support from the National Science Foundation through CIAN NSF ERC under grant #EEC-0812072

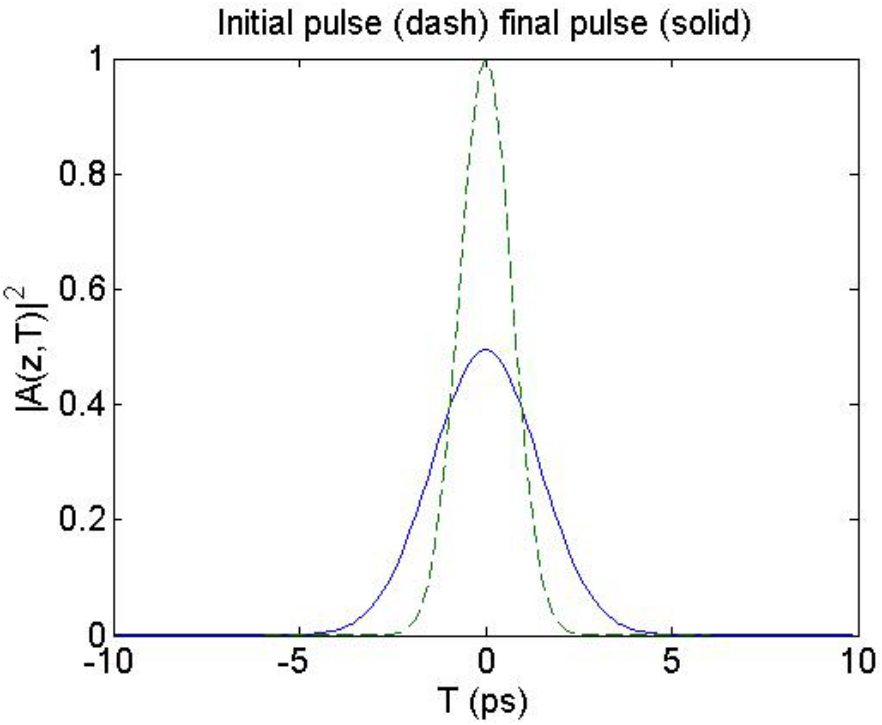

Figure.3.1 *Pulse power versus time* T *for* $z = 0.07$ *km*.

For this example there is no initial chirp so the figure shows *dispersion-induced pulse broadening* as discussed in Sec. 2.6, the initial pulse being shown as the dash line and the final pulse as the solid line. There are two noteworthy features of this plot: The peak power $|A(z,0)|^2$ at $T = 0$ has reduced from P_0 to $P_0/2$ under propagation (recall that $P_0 = 1$), and the pulse width $T_1(z)$ has increased by a factor of 2 (if you read of the FWHM of the input and final pulses from **Figure. 3.1** you will see an increase of a factor of 2.). That the peak power in the pulses decreases inversely as the pulse width is a direct physical consequence of energy conservation. Furthermore, the increase in a factor of 2 in pulse width follows from the exact result for the pulse width

$$\frac{T_1(z)}{T_0} = \left[\left(1+\frac{C\beta_2 z}{T_0^2}\right)^2 + \left(\frac{z}{L_D}\right)^2\right]^{1/2}, \qquad \textit{(Equation 3.28)}$$

which for $C = 0, z = \sqrt{3}L_D$ gives $T_1(z)/T_0 = 2$. **Figure 3.1** therefore provides validation of the propagation algorithm and code.

Although the propagation algorithm and code can in principle be used for any propagation distance there is a limit due to the finite time grid width T_{max}. In particular, one must make sure

The authors would like to acknowledge support from the National Science Foundation through CIAN NSF ERC under grant #EEC-0812072

that $T_1(z)$ remains much less that T_{max} since in the discrete Fourier transform it is assumed that $A(z,T)$ is actually periodic in time with period T_{max}. If $T_1(z) \approx T_{max}$ then the assumed periodicity becomes apparent as the pulse fills the whole time window leading to a phenomenon called aliasing. An example of aliasing is shown in **Figure. 3.2** for $z = \sqrt{24} L_D = 0.196$ km and $T_1(z) = 5T_0 = 5$ ps, other parameters being the same as **Figure. 3.1**.

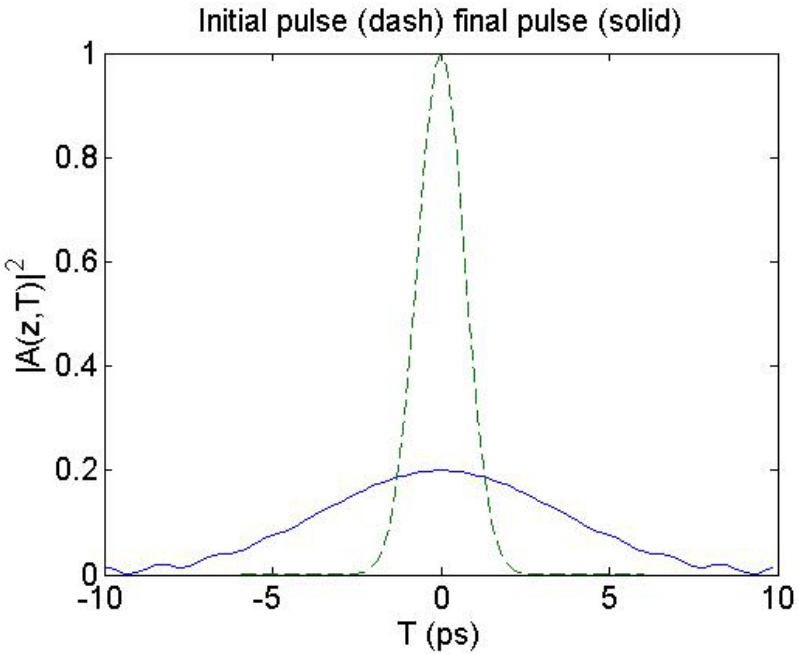

Figure.3.2 *Example of aliasing for* $z = 0.196$ *km*

You should take case to always avoid aliasing in your numerical simulations and look for signs that the field is starting to broaden over the whole time grid. For the example in **Figure. 3.2** we can obtain a reliable simulation by increasing the time grid size to $T_{max} = 40$. You should run the code for this case and verify that the peak power reduces by a factor 1/5 in keeping with the fact that the pulse width has increased by a factor of 5.

To finish with the Gaussian solution we consider a more complicated case with the same parameters as **Figure. 3.2** but with chirp parameter $C = -2$. **Figure. 3.3** shows a plot of the broadening factor $T_1(z)/T_0$ versus z according to **Equation. 3.28** and illustrates that in this case the chirp can cause the initial pulse to compress, though GVD will always eventually cause the pulse to broaden for large enough z.

The authors would like to acknowledge support from the National Science Foundation through CIAN NSF ERC under grant #EEC-0812072

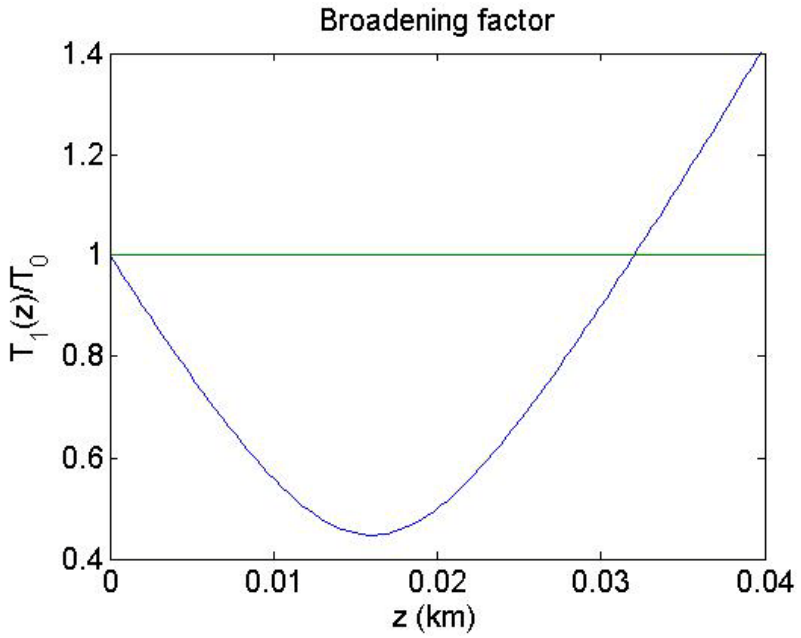

Figure. 3.3 *Broadening factor versus propagation for a chirp parameter* $C = -2$.

It is useful to plot the pulse propagation in this case by plotting $|A(z,T)|^2$ in the z-T plane by using color coding of the power values as shown in the figure below, the color bar to the right showing the color coding

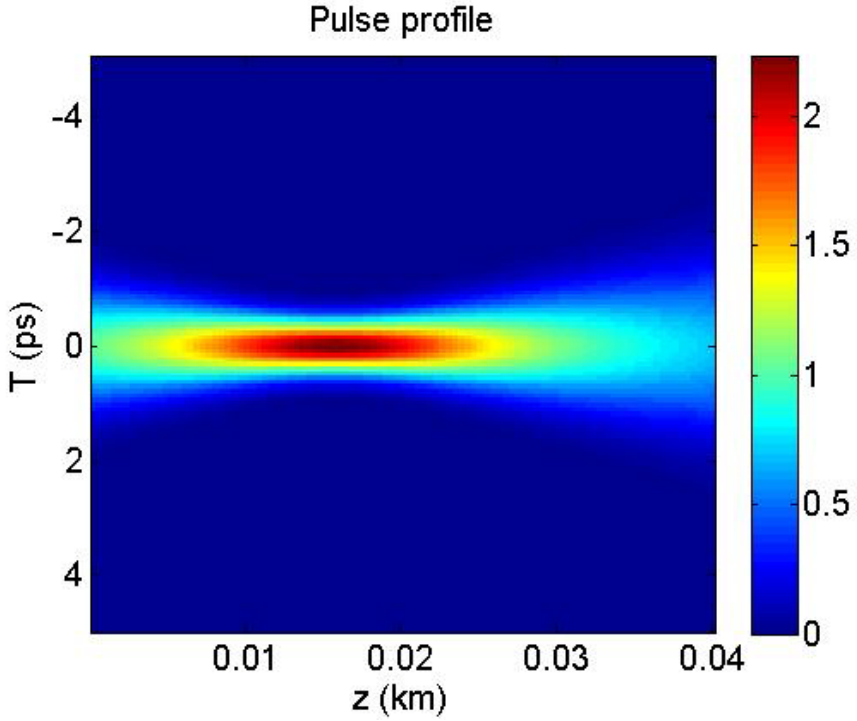

Figure. 3.4 *Pulse profile versus propagation distance z and time T for a chirped Gaussian pulse.*

The authors would like to acknowledge support from the National Science Foundation through CIAN NSF ERC under grant #EEC-0812072

Plots such as **Figure. 3.4** are of great utility in visualizing the dynamics of pulse propagation and provide a great aid to understanding. For the current example of pulse propagation with chirp the initial pulse is at z=0 to the left. With increasing pulse propagation we see the pulse compression as the profile becomes redder around $z = 0.016$ km and the pulse width reaches its minimum value, see **Figure. 3.3**, but eventually the pulse broadens again for z>0.016 km.

We shall employ figures such as **Figure. 3.4** to visualize a host of pulse dynamics in the following modules. The code PulseVisual.m on the next page was used to generate this pulse profile: it is essentially the same as LinearPulseProp.m but the propagation of length z is broken down into N_z small steps of length z / N_z using a for loop to run through the small steps, and the pulse profile at all the steps is plotted as in **Figure. 3.4.** We shall build upon this code in modules 4 and 5.
Our second example is that of a super-Gaussian pulse of the form

$$U(0,T) = \exp\left(-\frac{(1+iC)}{2}\left(\frac{T}{T_0}\right)^{2m} \right),$$

(Equation 3.29)

with $m \geq 1$ a positive integer. The code PulseVisual.m allows for super-Gaussian propagation via the input parameter m. The virtue of this example is that for m>1 there is no analytic solution, and for m>>1 this initial condition tends towards a top hat pulse shape. **Figure 3.5** shows the pulse profile for m=5 and no chirp and there is clearly a lot of dynamics. In spite of the fact that there is no initial chirp the pulse appears to compress as evidenced by the red region around z=0.01 km, but the pulse ultimately spreads.

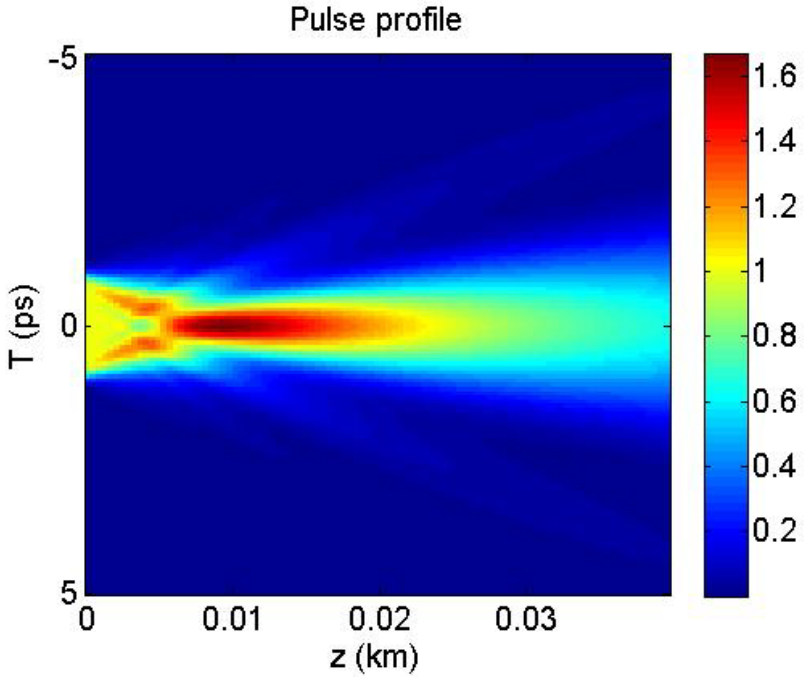

Figure. 3.5 *Pulse profile for a super-Gaussian pulse with m=5.*

The authors would like to acknowledge support from the National Science Foundation through CIAN NSF ERC under grant #EEC-0812072

```
function PulseVisual;
close all
%
%       Input parameters
%
Tmax = 40.00;          % temporal grid size in ps
N = 512;               % number of grid points
T0 = 1.0;              % pulse duration in ps
m = 5;                 % super-Gaussian order
                       % m < 0 hyperbolic-secand pulse
beta2 = +25;           % GVD parameter in ps^2/km
z = 0.04;              % porpagation distance in km
Nz = 100;              % number of points along z
C = -0.0;              % chirp parameter
%
%      Set up time and frequency grids
%
v = linspace(0,N-1,N);
dT = Tmax/N;
T = -Tmax/2 + v*dT;            % Time grid
dOmega = 2*pi/Tmax;
p = find(v > floor(N/2));      % These three lines set up the frequency
v(p) = v(p)-N;                 % grid in the storage form employed by
Omega = v*dOmega;              % Matlab.
%
A = exp(-((1+1i*C)/2)*(T/T0).^(2*m));         % Gaussian input pulse
A0 = A;
%
LD = T0^2/abs(beta2);
zval = linspace(0,z,Nz);
Bfac = sqrt((1+C*zval/LD).^2+(zval/LD).^2);   % Broadening factor
dz = z/Nz;
for iz = 1:Nz
At0 = fft(A);                                 % STEP 1 - FT initial pulse
Atilde = At0.*exp(1i*beta2*Omega.^2*dz/2);    % STEP 2 - phase factor
A = ifft(Atilde);                             % STEP 3 - inverse FT
Iprof(:,iz) = abs(A).^2;
end
%
%            Plot the output pulse
%
figure
plot(zval,Bfac,zval,ones(Nz,1));
set(gca,'FontSize',15);
xlabel('z (km)');
ylabel('T_1(z)/T_0');
title('Broadening factor');
%
figure
imagesc(zval,T,Iprof)
colorbar
set(gca,'FontSize',15);
xlabel('z (km)');
ylabel('T (ps)');
title('Pulse profile');
%
figure
%
plot(T,abs(A).^2,T,abs(A0).^2,'--');
set(gca,'FontSize',15);
xlabel('T (ps)');
ylabel('|A(z,T)|^2');
```


The authors would like to acknowledge support from the National Science Foundation through CIAN NSF ERC under grant #EEC-0812072

```
title('Initial pulse (dash) final pulse (solid)');
```

Figure 3.6 provides a movie or animation of how the pulse power profile evolves with increasing propagation distance z corresponding to the super-Gaussian pulse propagation in **Figure. 3.5.** You should run the movie to see how the pulse evolves. We shall use animations such as these as a learning aid in later modules.

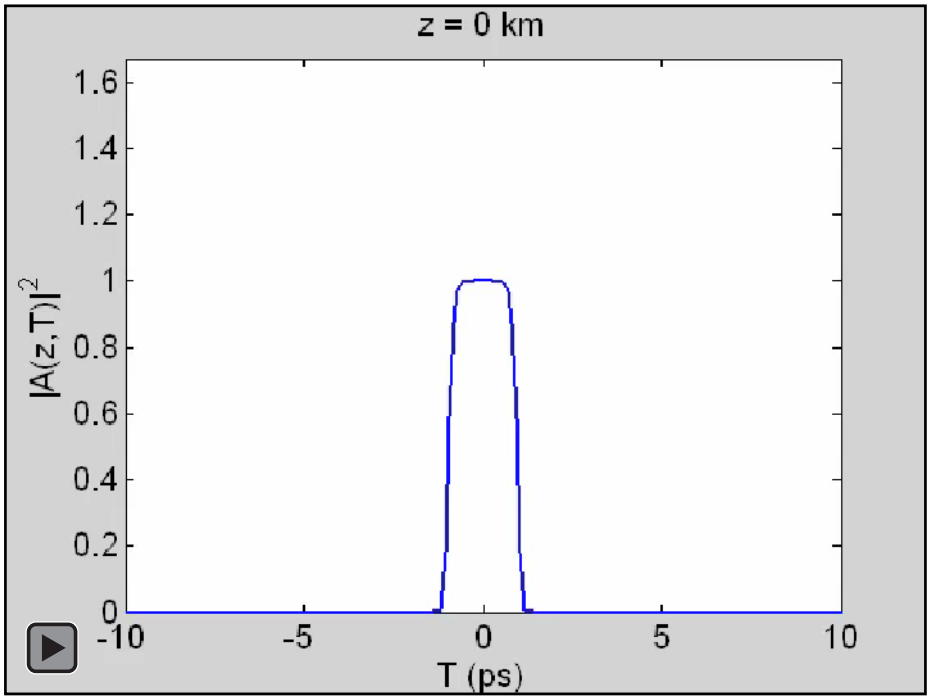

Figure. 3.6 *Animation of pulse propagation for a super-Gaussian with m=5.*

The first frame of the animation shows the initial profile and the final frame shows the pulse power profile for $z = 0.04$ km. Here we note that even for m=5 the initial pulse profile is quite close to a top hat shape, and the final pulse in the final frame has developed pronounced oscillations. In fact the power profile of the final pulse is quite close to a sinc-function squared where sinc(a)=sin(a)/a. One way to think of the pulse dynamics for a super-Gaussian for m>>5 is to exploit the analogy between propagation of a top hat pulse and diffraction of a plane-wave by a narrow slit in diffraction optics. It is well known that in the far field or Fraunhofer diffraction region beyond the slit the field intensity profile will be proportional to the Fourier transform of the initial field, the Fourier transform of a top hat being a sinc-function. In our problem the role of diffraction in the spatial domain is played by GVD in the temporal, and for large enough propagation distances an initial top hat pulse will approach a sinc-function in the time domain, and this is what is seen in the animation. For intermediate propagation distances 0<z<0.04 km the pulse dynamics is analogous to Fresnel diffraction with accompanying strong dynamics in the pulse profile as evidenced in **Figure. 3.5** and in the animation. For a discussion of diffraction by a slit see Sec. 10.2 of Ref. [1].

The authors would like to acknowledge support from the National Science Foundation through CIAN NSF ERC under grant #EEC-0812072

Our final example involves a hyperbolic-secant pulse of the form

$$A(z,T) = sech(T/T_0)e^{-iCT^2/2T_0^2},$$

(Equation 3.30)

where $sech(a) = 1/\cosh(a) = 2/(\exp(a)+\exp(-a))$. The main reason we choose to introduce the hyperbolic-secant pulse at this point is that the *temporal optical solitons* we shall study in module 5 have this pulse profile. In many respects hyperbolic-secant pulses are very similar qualitatively to Gaussian pulses in that they are bell shaped. **Figure 3.7** shows the pulse profile for a chirped hyperbolic pulse with $C = -2$ and the same parameters as for **Figure. 3.4.**

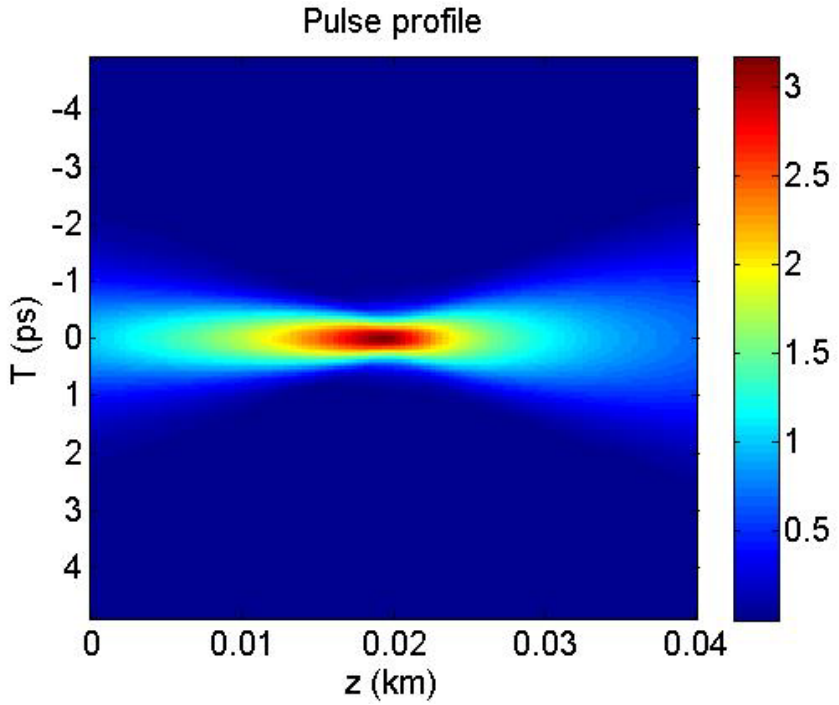

Figure. 3.7 *Pulse profile versus propagation distance z and time T for a chirped hyperbolic-secant pulse* with $C = -2$.

Comparing with **Figure. 3.4** for the Gaussian pulse we see that the hyperbolic-secant pulse corresponds to a broader initial condition for a given pulse width T_0. This follows since for $T >> T_0$ the hyperbolic-secant decays as $\exp(-|T|/T_0)$ whereas the Gaussian decays more rapidly as $\exp(-T^2/T_0^2)$. Given this qualitative difference the Gaussian and hyperbolic-secant pulse results in **Figures. 3.4 and 3.7** are qualitatively the same, with the hyperbolic-secant reaching the minimum pulse width a bit later at z=0.02 in comparison to the Gaussian for which the minimum pulse width occurs at z=0.016 km.

The authors would like to acknowledge support from the National Science Foundation through CIAN NSF ERC under grant #EEC-0812072

The goal of this module was to introduce the ideas of numerical linear pulse propagation in fibers in a way that we shall extend to the nonlinear case in modules 4 and 5. The utility of the codes presented has been illustrated with the examples of Gaussian, super-Gaussian, and hyperbolic-secant pulses. We shall build upon these basic codes to simulate and understand nonlinear propagation in optical fibers in modules 4 and 5.

For additional reading on the material covered in this section see Chap. 3.2 of Ref. [3].

Some contemporary engineering problems that require knowledge of the material taught in this module are

- Numerical evaluation of Fourier transforms from numerical or experimental data for signal processing applications.
- Simulation of pulse propagation and distortion between repeaters in a telecommunication link.
- Simulation and design of linear fiber optics systems.

References

1. E. Hecht, Optics, 4th Ed. (Addison Wesley, San Francisco, 2002).
2. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, 3rd Ed. (Cambridge University Press, 2007).
3. G. P. Agrawal, *Nonlinear Fiber Optics*, 3rd Ed. (Academic Press, San Diego 2001).

The authors would like to acknowledge support from the National Science Foundation through CIAN NSF ERC under grant #EEC-0812072