Solver Guide for the MATLAB bulk-solid pulse propagation

Yi-Hao Chen
Applied and Engineering Physics, Cornell University

December 27, 2024


CORNELL UNIVERSITY
FOUNDED A.D. 1865

Contents

1 Overview **5**
1.1 Mathematical background 5
1.2 High-level understanding of this package 6

2 Before I go deeply into details **7**
2.1 Introduction 7

3 Input arguments **9**
3.1 fiber 9
3.2 initial_condition 10
3.3 sim 10
3.3.1 Polarization modes 11
3.3.2 Adaptive-step method 11
3.3.3 Algorithm to use 12

4 Output arguments **15**

5 load_default_UPPE3D_propagate() **17**

6 Diagram of the calling sequence **19**

7 Derivation **21**


CORNELL UNIVERSITY
FOUNDED A.D. 1865

Chapter 1

Overview

1.1 Mathematical background

This package aims to solve the 3D unidirectional pulse propagation equation (3D-UPPE):

$$\partial_z \vec{\mathbb{A}}(\vec{k}_\perp, z, \Omega) = i\left[K_z - (\beta_0 + \beta_1\Omega)\right]\vec{\mathbb{A}}(\vec{k}_\perp, z, \Omega) + i\frac{1}{2k_\omega}\mathfrak{F}_{k_\perp}\left[k_W^2(\vec{r}_\perp)\vec{\mathbb{A}}(\vec{r}_\perp, z, \Omega)\right] + i\frac{\omega^2}{2k_\omega c^2}\frac{\vec{P}(\vec{k}_\perp, z, \Omega)}{\epsilon_0}, \quad (1.1)$$

where $\vec{\mathbb{A}}$ is the field envelope (in $\sqrt{\mathrm{W/m^2}}$), with $\vec{\mathbb{A}}(\vec{k}_\perp) = \mathfrak{F}_{k_\perp}\left[\vec{\mathbb{A}}(\vec{r}_\perp)\right] = \mathfrak{F}_{k_x}\left[\mathfrak{F}_{k_y}\left[\vec{\mathbb{A}}(\vec{r}_\perp)\right]\right]$ representing the spatial frequency component after the spatial Fourier Transform $[\mathfrak{F}_{k_x}[A](k_x) = C_{\mathfrak{F}_{k_x}}\int_{-\infty}^{\infty} A(x)\mathrm{e}^{-ik_x x}\,\mathrm{d}x$ and $\vec{r}_\perp = (x, y)]$, and $\vec{\mathbb{A}}(\Omega) = \mathfrak{F}\left[\vec{\mathbb{A}}(T)\right]$ representing the spectral component after spectral Fourier Transform. Note that the spatial and spectral Fourier Transforms follow different conventions due to following $(kx - \omega t)$: the spatial Fourier Transform follows the same convention as in mathematics whereas the spectral Fourier Transform does not. The spectral Fourier Transform is applied with respect to $\Omega = \omega - \omega_0$, where ω_0 is the center frequency of the numerical frequency window. $K_z(\vec{k}_\perp, \Omega) = \sqrt{k_\omega^2(\omega) - \left|\vec{k}_\perp\right|^2}$ represents both the (spectral) dispersion (from k_ω), and the (spatial) diffraction [from $\vec{k}_\perp = (k_x, k_y)$] terms. $k_W^2(\vec{r}_\perp)$ determines the waveguide effect; it is zero if the medium is homogeneous. They are determined by splitting the material's $k^2(\vec{r}_\perp, \omega) = \omega^2\mu_0\epsilon_0\epsilon_r(\vec{r}_\perp, \omega)$ into its frequency-dependent part $k_\omega^2(\omega)$ and spatially-dependent (but frequency-independent) part $k_W^2(\vec{r}_\perp)$:

$$k^2(\vec{r}_\perp, \omega) = k_\omega^2(\omega) + k_W^2(\vec{r}_\perp). \quad (1.2)$$

β_0 and β_1 are to reduce the propagating global phase increment to facilitate simulations in which β_1 is the inverse group velocity of the moving reference frame that introduces the delayed time $T = t - \beta_1 z$. $k_\omega(\omega) = \sqrt{\epsilon_r(\omega)}k_0$ is the position-independent wave vector from the medium contribution that determines the (spectral) dispersion. $\vec{P}(\vec{r}_\perp, z, \Omega) = \mathfrak{F}\left[\vec{P}(\vec{r}_\perp, z, T)\right]$ is the

nonlinear term that follows

$$P^i(\vec{r}_\perp,z,t)=\frac{2\epsilon_0 n n_2}{3}\sum_{j=x,y}\Bigg\{(1-f_R)\left[\left(\mathbb{A}^j\right)^2\left(\mathbb{A}^i\right)^*+2\left|\mathbb{A}^j\right|^2\mathbb{A}^i\right] + f_R\Bigg\{f_a\left[3\mathbb{A}^i(t)\int_0^\infty h_a(\tau)\left|\mathbb{A}^j(t-\tau)\right|^2\mathrm{d}\tau\right] + f_b\left[\mathbb{A}^j(t)\int_0^\infty\frac{3}{2}h_b(\tau)\left(\mathbb{A}^i(\mathbb{A}^j)^*+(\mathbb{A}^i)^*\mathbb{A}^j\right)(t-\tau)\,\mathrm{d}\tau\right]\Bigg\}\Bigg\}, \tag{1.3}$$

(for $i=x,y$) which becomes

$$\begin{aligned}P(\vec{r}_\perp,z,t)&=2\epsilon_0 n n_2\Big[(1-f_R)\left|\mathbb{A}\right|^2\mathbb{A} + f_R\mathbb{A}(t)\int_0^\infty(f_a h_a+f_b h_b)(\tau)\left|\mathbb{A}(t-\tau)\right|^2\mathrm{d}\tau\Big]\\&=2\epsilon_0 n n_2\left\{(1-f_R)\left|\mathbb{A}\right|^2\mathbb{A}+f_R\left[\left[f_a h_a(t)+f_b h_b(t)\right]*\left|\mathbb{A}(t)\right|^2\right]\mathbb{A}(t)\right\}\end{aligned} \tag{1.4}$$

for scalar (singly-polarized) fields. n_2 is the material nonlinear refractive index (in m^2/W). f_R is the Raman fraction representing the contribution of the Raman response of all nonlinearities where f_a and f_b are Raman fractions of the total Raman response for isotropic and anisotropic Raman responses, respectively ($f_a+f_b=1$); $h_a(t)$ and $h_b(t)$ are isotropic and anisotropic Raman response functions [1, 2].

1.2 High-level understanding of this package

This package is designed to solve for nonlinear pulse propagation in bulk space (*e.g.*, crystals or free space). It's originally designed to solve for multimode fibers, but it can also be used for general bulk medium with a customized index profile. Not only scalar but also polarized fields can be simulated. The package exhibits an adaptive control of the step size. In addition to CPU, highly parallelized cuda computation with a Nvidia GPU is implemented. The package uses "RK4IP" (Runge-Kutta in the interaction picture) [3, 4].

The fastest way to learn how to use this code is to start with the example codes in the package.

Chapter 2

Before I go deeply into details

2.1 Introduction

This document describes how to use the UPPE3D_propagate() MATLAB function.

Below is how to call this function in general.

```
prop_output = UPPE3D_propagate(fiber, ...
                               initial_condition, ...
                               sim)
```

prop_output
: It contains the information of the output field after propagating through the medium, such as the field amplitudes and the positions of each saved field, etc.

fiber
: It contains the information of the fiber, such as the index profile.

initial_condition
: It contains the information of the input.

sim
: It contains a multitude of information about the simulation, such as the algorithm to use and the center wavelength, etc.

Chapter 3

Input arguments

Below, I use N_t as the number of time/frequency sampling points, N_x and N_y as the number of spatial sampling points in x and y dimensions, respectively. N_p is the number of polarizations. N_z is the number of saved fields after the propagation. Overall, the dimension of simulation arrays follows $N_t \times N_x \times N_y \times N_z \times N_p$.

I recommend to use the information below as a reference guide if you're confused. Start with an example script is always better than reading this first.

Some parameters are required only when you enable some settings. Below I labeled in blue the parameters required all the time.

3.1 fiber

fiber

This specifies the fiber type. It is used only with the index-creating function "calc_index_profile()," so in general, this parameter is not required depending on simulations.

n

This is the index profile of the medium. Its size is $N_t \times N_x \times N_y \times 1 \times N_p$.

It can be used with the "calc_index_profile()" function to generate the index profile of fibers. For bulk crystals, just use "repmat()" to create a homogeneous index profile. Below is the comment of "calc_index_profile():"

```
function [index_profile,dx] = calc_index_profile(fiber,Nx,spatial_window,f)
%CALC_INDEX_PROFILE It calculates the index profile of the fiber.
%
%   fiber: a structure with
%
%       Below you can use the fiber in the repository (in "fiber_collections.m")
%       or set all parameters yourself
%       (1) Set it yourself
%           fiber_type - a string with either 'step', 'Nufern', or 'GRIN'
%           material - fiber material; usually 'silica'
%           diameter - fiber core diameter (um)
%           NA - numerical aperture
%           extra_params - extra parameters for different fiber types
%
%       (2) Use fiber repository
%           fiber - the name of the fiber
%                   Currently I have '1060XP',
%                                    'YB1200-4_125',
%                                    'YB1200-6_125DC', 'YB1200-6_125DC-PM',
```

```
%                                       'YB1200-10_125DC', 'YB1200-10_125DC-PM',
%                                       'YB1200-20_400DC',
%                                       'YB1200-25_250DC', 'YB1200-25_250DC-PM',
%                                       'ER30-4_125', 'ER110-4_125',
%                                       'ER16-8_125', 'ER80-8_125',
%                                       'M5-980-125', 'M12-980-125',
%                                       'OM1','OM2','OM3','OM4',
%                                       'FUD-7005',
%                                       'PLMA-YDF-30_400-VIII'.
%
% ------------------------------------------------------------------------
% (1) step fiber:
%     extra_params = [];
% (2) GRIN fiber:
%     extra_params.alpha = 2.08; % Shape parameter
% ------------------------------------------------------------------------
%
%   Nx - number of spatial grid points
%   spatial_window - full spatial window size (um) (usually set to 100 um)
%   f - frequency points (from small to large) (THz)
```

n2

It's the nonlinear coefficient of the fiber. By default, GMMNLSE_propagate() uses $2.3 \times 10^{-20}\,\mathrm{m}^2/\mathrm{W}$ assuming we use a silica fiber around 1 µm if fiber.n2 is left empty.

L0

This is the fiber length. The unit is meter.

material

This specifies the fiber material. It's 'silica' by default. It's used only for specifying which Raman model to use. It's either 'silica', 'chalcogenide', or 'ZBLAN'.

3.2 initial_condition

dt

This is the time sampling step $\triangle t$ with a unit of ps.

dx

This is the spatial sampling step $\triangle x$ with a unit of m.

dy

This is the spatial sampling step $\triangle y$ with a unit of m.

fields

This is the input field temporal amplitude under the time domain ($\mathbb{A}(t)$ with the unit $\sqrt{\mathrm{W}/\mathrm{m}^2}$). Its size is $N_t \times N_x \times N_y \times 1 \times N_p$.

If its size is $N_t \times N_x \times N_y \times N_z \times N_p$, only the last N_z is taken as the input field.

3.3 sim

Below are the most basic parameters for a simualtion.

betas

In UPPE, we not only create a moving frame that follows the pulse with the inverse velocity $\beta_{(1)}$ but extract out the reference propagation constant $\beta_{(0)}$. The benefit of extracting $\beta_{(0)}$ is that it reduces the rate of global phase increment such that the simulation can run with a larger step. This is similar to the limitation of multimode simulations that different spatial modes have different propagation constants that generate beating. To resolve the multimode beating, the size of the z-step cannot be too large.

This "betas" is a 2×1 column vector.

$$\begin{bmatrix} \beta_{(0)} \\ \beta_{(1)} \end{bmatrix} \tag{3.1}$$

f0

The center frequency (THz). It's a scalar.

save_period

The length between saved fields (m). If it's zero, it's equivalent to save_period=fiber.L0 that saves only the input and output fields.

3.3.1 Polarization modes

Here are the parameters if the simulation includes polarization modes.

scalar

false (0)	includes polarization-mode coupling
true (1)	don't include polarization-mode coupling

ellipticity

The ellipticity of the polarization modes. Please refer to "Nonlinear Fiber Optics, eq (6.1.18) Agrawal" for the equations.

0	linear polarization	(+,-)=(x,y)
1	circular polarization	(+,-)=(right,left)

3.3.2 Adaptive-step method

Here are the parameters for the adaptive-step method which this code always uses. All parameters are contained within a "sim.adaptive_dz" structure. The user doesn't need to specify whether to use adaptive-step method or not; the code determines itself. With the adaptive-step method, the initial step size is set to a small 10^{-6} m.

adaptive_dz.threshold

The nonlinear threshold of the upper-level adaptive-step method (see Ch. 6). It controls the accuracy of the simulation and determines whether to increase or decrease the step size. I typically use 10^{-6}.

adaptive_dz.DW_threshold

The dispersion+diffraction threshold of the nested adaptive-step method (see Ch. 6). It controls the accuracy of the simulation and determines whether to increase or decrease the step size. I typically use 10^{-6}.

adaptive_dz.max_dz
The maximum z-step size (m) of the adaptive-step method. It's 1/10 the save_period by default.

3.3.3 Algorithm to use

gpu_yes
true (1) use GPU
false (0) don't use GPU

include_Raman
false(0) ignore Raman effect
true(1) Raman model including the anisotropic contribution ("Ch. 2.3, p. 43" and "Ch. 8.5, p. 340," Nonlinear Fiber Optics (5th), Agrawal)

Typically, only isotropic Raman is considered, which is based on a single vibrational Raman mode of molecules (Ch. 2.3, p.42, Nonlinear Fiber Optics (5th), Agrawal). Here, we include the anisotropic part if there is an existing model, such as the one in silica ("Ch. 2.3, p. 43" and "Ch. 8.5, p. 340," Nonlinear Fiber Optics (5th), Agrawal).

For more details about anisotropic Raman, please read "Raman response function for silica fibers," by Q. Lin and Govind P. Agrawal (2006). Besides silica, chalcogenide and ZBLAN are also included.

pulse_centering
Because the pulse will evolve in the fiber, it's hard to have the moving frame always move with the same speed as the pulse. As a result, the pulse will go out of the time window and come back from the other side due to the use of periodic assumption of discrete Fourier Transform. The shift in time is saved in "prop_output.t_delay" so that you don't lose the information

When enabling pulse_centering, the pulse will be centered to the center of the time window based on the moment of the field intensity ($|A|^2$).

true (1) center the pulse according to the time window
false (0) don't center the pulse

cuda_dir_path
The path to the cuda directory into which ptx files will be compiled and stored. This is "/GMMNLSE/cuda/."

gpuDevice.Index
The GPU to use. It's typically 1 if the computer has only one GPU. MATLAB starts the index with 1.

Here are the parameters for the progress bar used in the simulation. It's useful in general to see how a simulation progresses.

progress_bar
true (1) show progress bar
false (0) don't show progress bar

progress_bar_name

The name of the GMMNLSE shown on the progress bar. If not set (no "sim.progress_bar_name"), it uses a default empty string, ".

Chapter 4

Output arguments

fields
The $N_t \times N_x \times N_y \times N_z \times N_p$ output fields.

dt
This is the time sampling step $\triangle t$ with a unit of ps.

dx
This is the spatial sampling step $\triangle x$ with a unit of m.

dy
This is the spatial sampling step $\triangle y$ with a unit of m.

z
This is the positions of each saved field.

dz
The z-step size (m).

This contains the step size at each saved point. You can see how the step size evolves through the propagation with this parameter.

betas
The "sim.betas,"$[\beta_{(0)}; \beta_{(1)}]$, used in this propagation.

t_delay
The time delay of the pulse at each saved point due to pulse centering.

seconds
The time spent for this simulation.

Chapter 5

load_default_UPPE3D_propagate()

Because of the overwhelming parameters of input arguments, I've created a function that loads the default value for each parameter. If a user has specified the value already, the user's value precedes over the default one.

Here is a typical way of calling this function.

```
[fiber,sim] = load_default_UPPE3D_propagate(input_fiber,...
                                            input_sim])
```

input_fiber and input_sim are user-defined parameters. Below are some examples.

```
% User-defined parameters
fiber.L0 = 3; % m

% Incorporate default settings
[fiber,sim] = load_default_UPPE3D_propagate(fiber,[]);

% If there are "sim" settings
sim.gpu_yes = false;
[fiber,sim] =  load_default_UPPE3D_propagate(fiber,sim);

% Use only user-defined "sim", not "fiber"
[fiber,sim] = load_default_UPPE3D_propagate([],sim);
```

Besides loading the default values, this function gives a user more options to obtain several parameters. This function transforms them into the allowed parameters of UPPE3D_propagate(). I list them below. If both equivalence are specified unfortunately, the allowed UPPE3D_propagate() input has the higher priority.

Description	Allowed UPPE3D_propagate()'s input	Equivalent input arguments for this function
center frequency/wavelength	sim.f0 (THz)	sim.lambda0 (m)

Below is the process flow of this "load_default_UPPE3D_propagate()" function. Read this if you're not sure whether your input will be used or overwritten. Because user-defined parameters take precedence, overwritten should happen only for (f0,lambda0) mentioned above.

```
% <-- Uncorrelated parameters are loaded directly -->

sim.f0 - depend on input f0 or lambda0
```

```
4           If no input f0 or lambda0, f0=3e5/1030e-9 (THz)
5
6 % If there's a user-defined one, use user's instead for the parameters below.
7 % Below I list the default values -->
8 fiber.material = 'silica';
9 fiber.n2 = 2.3e-20;
10
11 sim.save_period = 0;
12
13 sim.ellipticity = 0; % linear polarization
14 sim.scalar = true;
15
16 sim.adaptive_dz.DW_threshold = 1e-6;
17 sim.adaptive_dz.threshold = 1e-6;
18
19 sim.gpu_yes = true;
20 sim.pulse_centering = true;
21 sim.include_Raman = true;
22 sim.gpuDevice.Index = 1;
23 sim.progress_bar = true;
24 sim.progress_bar_name = '';
25 sim.cuda_dir_path = 'UPPE3D/cuda';
```

Chapter 6

Diagram of the calling sequence

It's not necessary to know how or when each function is called. I keep it here for documentation or in case someone wants to modify the code.

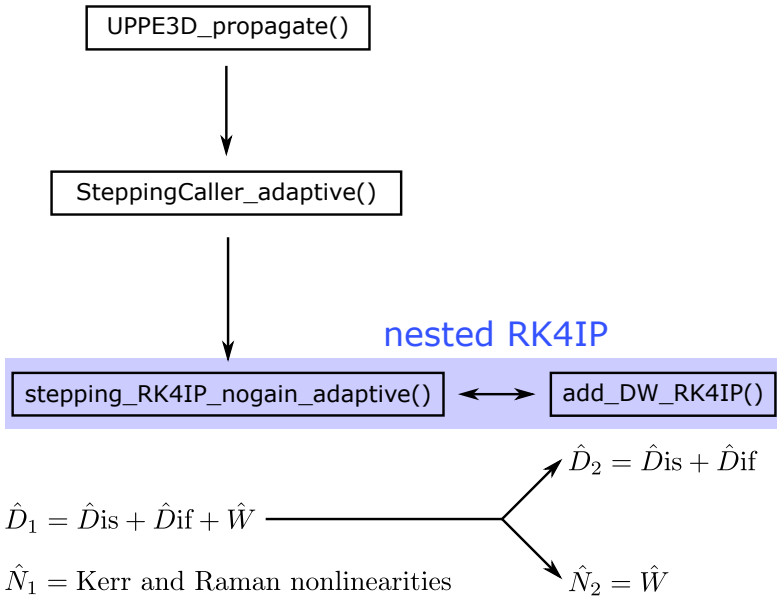


Figure 6.1: Diagram of the calling sequence.

The code uses "RK4IP" (Runge-Kutta in the interaction picture) [3, 4] with an adaptive step-size control, whose dispersion operation $\hat{D}_1$ contains dispersion $\hat{D}$is, diffraction $\hat{D}$if, and waveguide $\hat{W}$ effects, and nonlinear operation $\hat{N}_1$ contains Kerr and Raman nonlinearities. However, due to the incommutableness of the dispersion+diffraction $\hat{D}\text{is} + \hat{D}\text{if} \sim K_z$ and the waveguide $\hat{W} \sim k_W^2$ operators, we apply a second RK4IP for them whenever we need to apply $\hat{D}_1$. As typical dispersion and nonlinear operators that are incommutable, the nested RK4IP has a dispersion operator $\hat{D}_2 = \hat{D}\text{is} + \hat{D}\text{if}$ and a nonlinear operator $\hat{N}_2 = \hat{W}$. Because of two adaptive-step RK4IP, there are two threshold parameters to control each behavior, which by default are both set to 10^{-6}.

Chapter 7

Derivation

In this chapter, I show the derivation of 3D-UPPE [Eq. (1.1)]. Although it has been shown in the paper [5], I realized that there are a few mistakes in equations. Filing an errata might be a future option, but it's good for me to have one document that I can easily edit and update.

In fiber optics, light propagates in a dielectric medium, or gas if it is a hollow-core fiber, with its waveguide boundary conditions. Since there are no charge and current sources, such electromagnetic fields are governed by the following Maxwell's equations:

$$\nabla \times \vec{\mathbb{E}}(\vec{x},t) = -\mu_0 \partial_t \vec{\mathbb{H}}(\vec{x},t) \tag{7.1a}$$

$$\nabla \times \vec{\mathbb{H}}(\vec{x},t) = \epsilon_0 \partial_t \left[\epsilon_r(\vec{x},t) * \vec{\mathbb{E}}(\vec{x},t)\right] + \partial_t \vec{\mathbb{P}}(\vec{x},t), \tag{7.1b}$$

where $\vec{\mathbb{E}}$ and $\vec{\mathbb{H}}$ are electric and magnetic fields, respectively; $\vec{\mathbb{P}}$ is the induced nonlinear polarization; ϵ_r is the relative dielectric constant of the medium; ϵ_0 and μ_0 are permittivity and permeability in vacuum. We can do spectral Fourier Transform to the equations and consider only their positive-frequency components (analytic signal).

$$\mu_0 i\omega \vec{\mathcal{H}} = \nabla \times \vec{\mathcal{E}} \tag{7.2a}$$

$$\vec{\mathcal{J}} - i\omega\vec{\mathcal{P}} - i\omega\epsilon_0\epsilon_r\vec{\mathcal{E}} = \nabla \times \vec{\mathcal{H}}, \tag{7.2b}$$

which leads to

$$\begin{aligned} \nabla \times \nabla \times \vec{\mathcal{E}} &= \nabla\left(\nabla \cdot \vec{\mathcal{E}}\right) - \nabla^2\vec{\mathcal{E}} \\ &= \mu_0 i\omega\vec{\mathcal{J}} + \mu_0\omega^2\vec{\mathcal{P}} + \omega^2\mu_0\epsilon_0\epsilon_r\vec{\mathcal{E}} \end{aligned} \tag{7.3}$$

$\vec{\mathcal{E}}$, $\vec{\mathcal{H}}$, $\vec{\mathcal{J}}$, and $\vec{\mathcal{P}}$ are the complex-valued analytic signal of their corresponding real-valued counterparts. We assume that the electric field and the medium response (current $\vec{\mathcal{J}}$, nonlinear polarization $\vec{\mathcal{P}}$) are transverse, *i.e.*, perpendicular to the propagation direction determined by the wave number $\vec{k}$. This standard assumption in propagation of electromagnetic fields means that the term $\nabla\left(\nabla \cdot \vec{\mathcal{E}}\right)$ can be neglected. This remains valid as long as beams are not too strongly focused. When beam numerical aperture exceeds a few percent, a small longitudinal component $\vec{\mathcal{E}}$ may develop close to the focus and makes this approximation invalid. In addition, we also assume that there is no current $\vec{\mathcal{J}} = 0$. After these assumptions, we obtain

$$\nabla^2\vec{\mathcal{E}}(\vec{r}_\perp,z,\omega) + k^2(\vec{r}_\perp,\omega)\vec{\mathcal{E}}(\vec{r}_\perp,z,\omega) = -\mu_0\omega^2\vec{\mathcal{P}}(\vec{r}_\perp,z,\omega), \tag{7.4}$$

where $k^2 = \omega^2\mu_0\epsilon_0\epsilon_r$ and $\vec{r}_\perp = (x, y)$. Note that $\epsilon_r = \epsilon_r(\vec{r}_\perp, \omega)$ depends on x and y and is independent of z. This accounts for the variation in refractive index of the cross-sectional profile. We can split k^2 into its frequency-dependent part $k_\omega^2(\omega)$ and spatially-dependent (but frequency-independent) part $k_W^2(\vec{r}_\perp)$:

$$k^2(\vec{r}_\perp, \omega) = k_\omega^2(\omega) + k_W^2(\vec{r}_\perp). \tag{7.5}$$

$k_W^2(\vec{r}_\perp)$ determines the waveguide effect. It is zero if the medium is homogeneous.

By transforming Eq. (7.4) into the cross-sectional $k_\perp$-space, we obtain

$$\left[\partial_z^2 + \left(k_\omega^2(\omega) - \left|\vec{k}_\perp\right|^2\right)\right]\vec{\mathcal{E}}(\vec{k}_\perp, z, \omega) + \mathfrak{F}_{k_\perp}\left[k_W^2(\vec{r}_\perp)\vec{\mathcal{E}}(\vec{r}_\perp, z, \omega)\right] = -\mu_0\omega^2\vec{\mathcal{P}}(\vec{k}_\perp, z, \omega). \tag{7.6}$$

Its $\left[\partial_z^2 + \left(k_\omega^2(\omega) - \left|\vec{k}_\perp\right|^2\right)\right]$ can be rewritten into $\left[\left(\partial_z + iK_z(\vec{k}_\perp, \omega)\right)\left(\partial_z - iK_z(\vec{k}_\perp, \omega)\right)\right]$, where $K_z(\vec{k}_\perp, \omega) = \sqrt{k_\omega^2(\omega) - \left|\vec{k}_\perp\right|^2}$. Assume that the forward propagating component is larger than the backward propagating one: $\partial_z \to iK_z$. The above equation becomes

$$\partial_z\vec{\mathcal{E}}(\vec{k}_\perp, z, \omega) = iK_z\vec{\mathcal{E}}(\vec{k}_\perp, z, \omega) + i\frac{1}{2K_z}\mathfrak{F}_{k_\perp}\left[k_W^2(\vec{r}_\perp)\vec{\mathcal{E}}(\vec{r}_\perp, z, \omega)\right] + i\frac{\omega^2}{2K_z c^2}\frac{\vec{\mathcal{P}}(\vec{k}_\perp, z, \omega)}{\epsilon_0}. \tag{7.7}$$

By introducing the envelope of the analytic signal:

$$\begin{aligned}
\vec{\mathbb{E}}(\vec{x}, t) &= \frac{1}{2}\left[\vec{\mathcal{E}}(\vec{x}, t) + \text{c.c.}\right] \quad , \vec{\mathcal{E}} \text{ is the analytic signal of } \vec{\mathbb{E}} \\
&= \frac{1}{2}\left[\vec{E}(t)\mathrm{e}^{i\left(\beta_{(0)}z - \omega_0 t\right)} + \text{c.c.}\right] && \text{(7.8a)} \\
\vec{\mathbb{P}}(\vec{x}, t) &= \frac{1}{2}\left[\vec{\mathcal{P}}(\vec{x}, t) + \text{c.c.}\right] \quad , \vec{\mathcal{P}} \text{ is the analytic signal of } \vec{\mathbb{P}} \\
&= \frac{1}{2}\left[\vec{P}(t)\mathrm{e}^{i\left(\beta_{(0)}z - \omega_0 t\right)} + \text{c.c.}\right], && \text{(7.8b)}
\end{aligned}$$

Eq. (7.7) becomes

$$\partial_z\vec{E}(\vec{k}_\perp, z, \omega) = iK_z\vec{E}(\vec{k}_\perp, z, \omega) + i\frac{1}{2K_z}\mathfrak{F}_{k_\perp}\left[k_W^2(\vec{r}_\perp)\vec{E}(\vec{r}_\perp, z, \omega)\right] + i\frac{\omega^2}{2K_z c^2}\frac{\vec{P}(\vec{k}_\perp, z, \omega)}{\epsilon_0}. \tag{7.9}$$

Kerr and Raman nonlinearity

Note that, in general, the nonlinear polarization has the following form [2]:

$$\vec{\mathbb{P}}(t) = \int_{-\infty}^{\infty}\chi^{(3)}(t_1, t_2, t_3) \vdots \vec{\mathbb{E}}(t - t_1)\vec{\mathbb{E}}(t - t_2)\vec{\mathbb{E}}(t - t_3)\,\mathrm{d}t_1\,\mathrm{d}t_2\,\mathrm{d}t_3, \tag{7.10}$$

where

$$\chi^{(3)}(t_1, t_2, t_3) = \delta(t_1)\delta(t_2 - t_3)\hat{R}^{ijk\ell}(t_3) \tag{7.11a}$$

$$\hat{R}^{ijk\ell}(t) = \epsilon_0\chi^{(3)}_{\text{electronic}}\frac{\delta^{ij}\delta^{k\ell} + \delta^{ik}\delta^{j\ell} + \delta^{i\ell}\delta^{jk}}{3}\delta(t) + \mathtt{R}_a(t)\delta^{ij}\delta^{k\ell} + \mathtt{R}_b(t)\frac{\delta^{ik}\delta^{j\ell} + \delta^{i\ell}\delta^{jk}}{2} \tag{7.11b}$$

includes the instantaneous Kerr nonlinearity and the delayed isotropic ($\mathtt{R}_a$) and anisotropic ($\mathtt{R}_b$) Raman nonlinearities; therefore,

$$\mathbb{P}^i(t) = \sum_{jk\ell} \mathbb{E}^j(t) \int_{-\infty}^{\infty} \hat{R}^{ijk\ell}(\tau)\mathbb{E}^k(t-\tau)\mathbb{E}^\ell(t-\tau)\,\mathrm{d}\tau, \qquad \text{by applying } (t_3 \to \tau). \tag{7.12}$$

First, we solve for the Kerr nonlinearity:

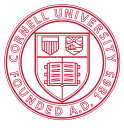
$$\begin{aligned}
\mathbb{P}^i(\vec{r},t) &= \sum_{jkl} \epsilon_0 \frac{1}{2}\left[E^j(t)\mathrm{e}^{i\left(\beta_{(0)}z-\omega_0 t\right)} + \text{c.c.}\right] \chi^{(3)}_{\text{electronic}} \frac{\delta^{ij}\delta^{k\ell} + \delta^{ik}\delta^{j\ell} + \delta^{i\ell}\delta^{jk}}{3} \\
&\qquad \times \frac{1}{2}\left[E^k(t)\mathrm{e}^{i\left(\beta_{(0)}z-\omega_0 t\right)} + \text{c.c.}\right] \frac{1}{2}\left[E^\ell(t)\mathrm{e}^{i\left(\beta_{(0)}z-\omega_0 t\right)} + \text{c.c.}\right] \\
&= \frac{\epsilon_0 \chi^{(3)}_{\text{electronic}}}{8} \sum_k \Bigg\{ \left(E^i(E^k)^2\mathrm{e}^{3i\left(\beta_{(0)}z-\omega_0 t\right)} + \text{c.c.}\right) \\
&\qquad + \left[\left((E^i)^*(E^k)^2 + 2E^i\left|E^k\right|^2\right)\mathrm{e}^{i\left(\beta_{(0)}z-\omega_0 t\right)} + \text{c.c.}\right]\Bigg\}.
\end{aligned} \tag{7.13}$$

Next, we solve for the isotropic Raman term:

$$\begin{aligned}
\mathbb{P}^i(\vec{r},t) &= \frac{1}{2}\left[E^i(t)\mathrm{e}^{i\left(\beta_{(0)}z-\omega_0 t\right)} + \text{c.c.}\right] \int \mathtt{R}_a(\tau) \sum_k \left\{\frac{1}{2}\left[E^k(t-\tau)\mathrm{e}^{i\left[\beta_{(0)}z-\omega_0(t-\tau)\right]} + \text{c.c.}\right]\right\}^2 \\
&= \frac{1}{8}\sum_k \Bigg\{ \left[E^i(t)\mathrm{e}^{3i\left(\beta_{(0)}z-\omega_0 t\right)} \int \mathtt{R}_a(\tau)\left(E^k(t-\tau)\right)^2 \mathrm{e}^{2i\omega_0\tau}\,\mathrm{d}\tau + \text{c.c.}\right] \\
&\qquad + \left[2E^i(t)\mathrm{e}^{i\left(\beta_{(0)}z-\omega_0 t\right)} \int \mathtt{R}_a(\tau)\left|E^k(t-\tau)\right|^2 \mathrm{d}\tau + \text{c.c.}\right] \\
&\qquad + \left[\left(E^i(t)\right)^* \mathrm{e}^{i\left(\beta_{(0)}z-\omega_0 t\right)} \int \mathtt{R}_a(\tau)\left(E^k(t-\tau)\right)^2 \mathrm{e}^{2i\omega_0\tau}\,\mathrm{d}\tau + \text{c.c.}\right]\Bigg\},
\end{aligned} \tag{7.14}$$

and the anisotropic term:

$$\begin{aligned}
\mathbb{P}^i(\vec{r},t) &= \sum_j \mathbb{E}^j(t) \int \mathtt{R}_b(\tau)\mathbb{E}^i(t-\tau)\mathbb{E}^j(t-\tau)\,\mathrm{d}\tau \\
&= \sum_j \frac{1}{2}\left[E^j(t)\mathrm{e}^{i\left(\beta_{(0)}z-\omega_0 t\right)} + \text{c.c.}\right] \\
&\qquad \int \mathtt{R}_b(\tau)\frac{1}{2}\left\{E^i(t-\tau)\mathrm{e}^{i\left[\beta_{(0)}z-\omega_0(t-\tau)\right]} + \text{c.c.}\right\}\frac{1}{2}\left\{E^j(t-\tau)\mathrm{e}^{i\left[\beta_{(0)}z-\omega_0(t-\tau)\right]} + \text{c.c.}\right\}\mathrm{d}\tau \\
&= \frac{1}{8}\sum_j \Bigg\{ \left[E^j(t)\mathrm{e}^{3i\left(\beta_{(0)}z-\omega_0 t\right)} \int \mathtt{R}_b(\tau)E^i(t-\tau)E^j(t-\tau)\mathrm{e}^{2i\omega_0\tau} + \text{c.c.}\right] \\
&\qquad + \left[E^j(t)\mathrm{e}^{i\left(\beta_{(0)}z-\omega_0 t\right)} \int \mathtt{R}_b(\tau)\left(E^i(E^j)^* + (E^i)^*E^j\right)(t-\tau)\,\mathrm{d}\tau + \text{c.c.}\right]
\end{aligned}$$



Cornell University

$$+\left[(E^j)^*\mathrm{e}^{i\left(\beta_{(0)}z-\omega_0 t\right)}\int \mathtt{R}_b(\tau)E^i(t-\tau)E^j(t-\tau)\mathrm{e}^{2i\omega_0\tau}\,\mathrm{d}\tau+\mathrm{c.c.}\right]\Bigg\}. \tag{7.15}$$

By ignoring the 3rd harmonic terms, which are terms with $\mathrm{e}^{3i\left(\beta_{(0)}z-\omega_0 t\right)}$, the polarization can be expressed, in terms of its envelope, as

$$\begin{aligned}P^i(\vec{r},t)=\frac{1}{4}\sum_j\Bigg\{&\epsilon_0\chi^{(3)}_{\text{electronic}}\left[(E^i)^*(E^j)^2+2E^i\left|E^j\right|^2\right]\\&+2E^i(t)\int \mathtt{R}_a(\tau)\left|E^j(t-\tau)\right|^2\mathrm{d}\tau+(E^i)^*(t)\int \mathtt{R}_a(\tau)(E^j)^2(t-\tau)\mathrm{e}^{2i\omega_0\tau}\,\mathrm{d}\tau\\&+E^j(t)\int \mathtt{R}_b(\tau)\left(E^i(E^j)^*+(E^i)^*E^j\right)(t-\tau)\,\mathrm{d}\tau\\&+(E^j(t))^*\int \mathtt{R}_b(\tau)\left(E^iE^j\right)(t-\tau)\mathrm{e}^{2i\omega_0\tau}\,\mathrm{d}\tau\Bigg\}.\end{aligned} \tag{7.16}$$

We further ignore the delay terms, and obtain

$$\begin{aligned}P^i(\vec{r},t)=\frac{1}{4}\sum_j\Bigg\{&\epsilon_0\chi^{(3)}_{\text{electronic}}\left[(E^i)^*(E^j)^2+2E^i\left|E^j\right|^2\right]\\&+2E^i(t)\int \mathtt{R}_a(\tau)\left|E^j(t-\tau)\right|^2\mathrm{d}\tau\\&+E^j(t)\int \mathtt{R}_b(\tau)\left(E^i(E^j)^*+(E^i)^*E^j\right)(t-\tau)\,\mathrm{d}\tau\Bigg\}.\end{aligned} \tag{7.17}$$

Since $\vec{E}$ above has the unit of V/m, we transform it into $\sqrt{\mathrm{W/m^2}}$, a more commonly used physical quantity such that $\left|\vec{\mathbb{A}}\right|^2$ directly corresponds to the intensity (in $\mathrm{W/m^2}$). The transformation relies on $\left|\vec{\mathbb{A}}\right|^2=\frac{\epsilon_0 n_{\text{eff}}c}{2}\left|\vec{E}\right|^2$, where $\vec{\mathbb{A}}$ has a unit of $\sqrt{\mathrm{W/m^2}}$. The transformed equation looks the same as Eq. (7.9) by replacing $\vec{E}$ with $\vec{\mathbb{A}}$, but the nonlinear polarization $\vec{P}$ becomes

$$\begin{aligned}P^i(\vec{r},t)=\frac{1}{2\epsilon_0 n_{\text{eff}}c}\sum_j\Bigg\{&\epsilon_0\chi^{(3)}_{\text{electronic}}\left[(\mathbb{A}^i)^*(\mathbb{A}^j)^2+2\mathbb{A}^i\left|\mathbb{A}^j\right|^2\right]\\&+2\mathbb{A}^i(t)\int \mathtt{R}_a(\tau)\left|\mathbb{A}^j(t-\tau)\right|^2\mathrm{d}\tau\\&+\mathbb{A}^j(t)\int \mathtt{R}_b(\tau)\left(\mathbb{A}^i(\mathbb{A}^j)^*+(\mathbb{A}^i)^*\mathbb{A}^j\right)(t-\tau)\,\mathrm{d}\tau\Bigg\}.\end{aligned} \tag{7.18}$$

Consider the moving frame:

$$T=t-\beta_{(1)}z \quad\Rightarrow\quad \partial_z\mathbb{A}(\vec{r},t)=\partial_z\mathbb{A}(\vec{r},T)-\beta_{(1)}\partial_T\mathbb{A}(\vec{r},T), \tag{7.19}$$

where the ∂_z on the left-hand side is actually a total derivative. Also, we apply the Fourier Transform with respect to the offset frequency $\Omega = \omega - \omega_0$.

$$\partial_z \vec{\mathbb{A}}(\vec{k}_\perp, z, \Omega) = i\left[K_z - (\beta_0 + \beta_1\Omega)\right]\vec{\mathbb{A}}(\vec{k}_\perp, z, \Omega) + i\frac{1}{2K_z}\mathfrak{F}_{k_\perp}\left[k_W^2(\vec{r}_\perp)\vec{\mathbb{A}}(\vec{r}_\perp, z, \Omega)\right] + i\frac{\omega^2}{2K_z c^2}\frac{\vec{P}(\vec{k}_\perp, z, \Omega)}{\epsilon_0}. \quad (7.20)$$

If we apply the model commonly used in solid-core silica fiber, where

$$\epsilon_0\chi^{(3)}_{\text{electronic}} = (1 - f_R)\,\epsilon_0\chi^{(3)}_{xxxx} = (1 - f_R)\,\frac{4\epsilon_0^2 n_{\text{eff}}^2 c}{3}n_2 \quad (7.21a)$$

$$\mathtt{R}_a = f_R f_a \epsilon_0 \chi^{(3)}_{xxxx}\frac{3}{2}h_a = 2 f_R f_a \epsilon_0^2 n_{\text{eff}}^2 c n_2 h_a(t) \quad (7.21b)$$

$$\mathtt{R}_b = f_R f_b \epsilon_0 \chi^{(3)}_{xxxx}\frac{3}{2}h_b = 2 f_R f_b \epsilon_0^2 n_{\text{eff}}^2 c n_2 h_b(t), \quad (7.21c)$$

we obtain Eq. (1.3).

Because K_z can be close to 0 as $|k_\perp|^2$ is huge, which can easily happen under a decently-huge $k_\perp$-space (with fine spatial sampling), the computation of the nonlinear term will easily blow up due to the $1/K_z$ term, artificially producing unreasonable results. Therefore, to simplify the simulation process, K_z in the denominator of the nonlinear term is replaced with k_ω. This assumption fails when the light has high-$\vec{k}_\perp$ components, such as tightly-focusing. However, during the derivation, we have already assumed that all the field and the responses are transverse to make $\nabla\left(\nabla\cdot\vec{\mathcal{E}}\right) = 0$, so the assumption for $K_z \to k_\omega$ makes sense. After all, we achieve deriving the final 3D-UPPE in Eq. (1.1).


CORNELL UNIVERSITY
FOUNDED A.D. 1865

Bibliography

[1] R. H. Stolen, J. P. Gordon, W. J. Tomlinson, and H. A. Haus, "Raman response function of silica-core fibers", J. Opt. Soc. Am. B **6**, 1159–1166 (1989).

[2] Q. Lin and G. P. Agrawal, "Raman response function for silica fibers", Opt. Lett. **31**, 3086–3088 (2006).

[3] A. M. Heidt, "Efficient Adaptive Step Size Method for the Simulation of Supercontinuum Generation in Optical Fibers", J. Light. Technol. **27**, 3984–3991 (2009).

[4] S. Balac and F. Mahé, "Embedded Runge-Kutta scheme for step-size control in the interaction picture method", Comput. Phys. Commun. **184**, 1211–1219 (2013).

[5] W. Wang, Y. Eisenberg, Y.-H. Chen, C. Xu, and F. Wise, "Efficient temporal compression of 10-μJ pulses in periodic layered Kerr media", Opt. Lett. **49**, 5787–5790 (2024).