Solver Guide for the MATLAB gas-filled-fiber pulse propagation

Yi-Hao Chen
Applied and Engineering Physics, Cornell University

January 17, 2024


CORNELL UNIVERSITY
FOUNDED A.D. 1865

Contents


CORNELL UNIVERSITY
FOUNDED A.D. 1865

Chapter 1

Overview

This package contains all the code needed to simulate pulse propagation in a gas-filled hollow-core fiber. Pulse follows the multimode unidirectional pulse propagation equation (MM-UPPE) which is shown in the following:

$$\begin{aligned}\partial_z A_p(z,\Omega) &= i\left[\beta_p(\omega) - \left(\beta_{(0)} + \beta_{(1)}\Omega\right)\right] A_p(z,\Omega) \\ &+ \frac{i\omega}{4}\sum_{\ell mn}\left\{\left(\frac{3}{4}\epsilon_0\chi^{(3)}_{\text{electronic}}\right) Q^K_{p\ell mn}\mathfrak{F}\left[A_\ell A_m A_n^*\right]\right. \\ &\qquad \left. + \left\{Q^{R_a}_{p\ell mn}\mathfrak{F}\left[A_\ell\left[\left(\frac{1}{2}\mathtt{R}_a\right) * \left(A_m A_n^*\right)\right]\right] + Q^{R_b}_{p\ell mn}\mathfrak{F}\left[A_\ell\left[\left(\frac{1}{2}\mathtt{R}_b\right) * \left(A_m A_n^*\right)\right]\right]\right\}\right\} \\ &+ \frac{i\omega}{4}\sum_m Q^I_{pm}(\omega) P_{I,m}(z,\Omega),\end{aligned} \tag{1.1}$$

which includes dispersion, as well as instantaneous electronic and delayed Raman nonlinearities. $A_p(z,t)$ is the electric field ($\sqrt{\text{W}}$) of mode p, whose Fourier Transform is $A_p(z,\Omega) = \mathfrak{F}[A_p(z,t)]$. The Fourier Transform is applied with respect to angular frequency $\Omega = \omega - \omega_0$, where ω_0 is the center angular frequency of the numerical frequency window required to cover the investigated physical phenomena. β_p is the propagation constant of the mode p, obtained with the dispersion formula in [1] for anti-resonant fiber platform or in [2] for capillaries; $\beta_{(0)}$ is to reduce the propagating global-phase increment to facilitate simulations, $\beta_{(1)}$ is the inverse group velocity of the moving frame. $\chi^{(3)}_{\text{electronic}}(\omega)$ is the third-order nonlinear susceptibility of the electronic response (m^2/V^2); $\mathtt{R}_a(t)$ and $\mathtt{R}_b(t)$ are isotropic and anisotropic Raman response functions. $P_{I,m}(z,\Omega)$ is the photoionization-induced polarization. p, ℓ, m, and n are the eigenmode indices; $Q^K_{p\ell mn}$, $Q^{\mathtt{R}_a}_{p\ell mn}$, and $Q^{\mathtt{R}_b}_{p\ell mn}$ are overlap integrals of eigenmode fields $\vec{F}_j(\vec{r}_\perp)$ (1/m), where $\vec{r}_\perp = (x,y)$:

$$Q^K_{p\ell mn} = \frac{2}{3}Q^{R_a}_{p\ell mn} + \frac{1}{3}Q^k_{p\ell mn} \qquad , Q^k_{p\ell mn} = \frac{\int\left(\vec{F}_p^* \cdot \vec{F}_n^*\right)\left(\vec{F}_\ell \cdot \vec{F}_m\right)\mathrm{d}x\,\mathrm{d}y}{N_p N_\ell N_m N_n} \tag{1.2a}$$

$$Q^{R_a}_{p\ell mn} = \frac{\int\left(\vec{F}_p^* \cdot \vec{F}_\ell\right)\left(\vec{F}_m \cdot \vec{F}_n^*\right)\mathrm{d}x\,\mathrm{d}y}{N_p N_\ell N_m N_n} \tag{1.2b}$$

$$Q^{R_b}_{p\ell mn} = \frac{1}{2}\left[\frac{\int\left(\vec{F}^*_p\cdot\vec{F}_m\right)\left(\vec{F}_\ell\cdot\vec{F}^*_n\right)\mathrm{d}x\,\mathrm{d}y}{N_pN_\ell N_mN_n} + Q^k_{p\ell mn}\right] = \frac{1}{2}\left(Q^{r_b}_{p\ell mn} + Q^k_{p\ell mn}\right), \tag{1.2c}$$

where $\vec{F}_p$ is the pth spatial eigenmode ($\vec{F}_p \in \mathbb{R}$), with the normalization condition, $\int \vec{F}^*_m\vec{F}_n\,\mathrm{d}^2x = \delta_{mn}$ and $N_p = \sqrt{\frac{\epsilon_0 n_{\mathrm{eff},p}c}{2}}$. n_{eff} and $n_{i,\mathrm{eff}}$ in each N_i ($i \in \{p,\ell,m,n\}$) is assumed to follow

$$\frac{\epsilon_0^2 n_{\mathrm{eff}}^2 c^2}{N_pN_\ell N_mN_n} = 4. \tag{1.3}$$

The Raman response functions are

$$\frac{1}{2}\mathtt{R}_a = R_a = R^{\mathrm{vib}} - 2R^{\mathrm{rot}} \tag{1.4a}$$

$$\frac{1}{2}\mathtt{R}_b = R_b = 6R^{\mathrm{rot}}, \tag{1.4b}$$

where

$$R^{\mathrm{vib}} = \Theta(t)N_g\frac{\left(\frac{\mathrm{d}\alpha}{\mathrm{d}\mathbb{Q}}\right)^2_0}{4\mu}\mathrm{e}^{-\gamma_2^{\mathrm{vib}}t}\sum_J(2J+1)\,\rho_J^{(0)}\frac{1}{\omega_{1,J;0,J}}\sin\left(\omega_{1,J;0,J}t\right) \tag{1.5a}$$

$$R^{\mathrm{rot}} = \Theta(t)N_g\frac{(\triangle\alpha)^2}{60\hbar}\mathrm{e}^{-\gamma_2^{\mathrm{rot}}t}\sum_J\left(\rho_J^{(0)} - \rho_{J+2}^{(0)}\right)\frac{(J+1)\,(J+2)}{(2J+3)}\sin\left(\omega_{0,J+2;0,J}t\right). \tag{1.5b}$$

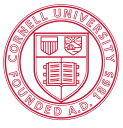
$\Theta(t)$ is the Heaviside step function, $\left(\frac{\mathrm{d}\alpha}{\mathrm{d}\mathbb{Q}}\right)_0$ is the polarizability derivative at equilibrium, μ is the reduced mass of a molecule. γ_2^{vib} and γ_2^{rot} are dephasing rates of vibrational and rotational SRS, respectively. $\omega_{\nu_2,J_2;\nu_1,J_1} = \omega_{\nu_2,J_2} - \omega_{\nu_1,J_1}$ is the transition frequency of state (ν_1,J_1) to (ν_2,J_2). ν and J are the vibrational and rotational quantum numbers, respectively.

It is worth noting that the field A_p is defined as

$$\begin{aligned}\vec{\mathbb{E}}(\vec{x},t) &= \frac{1}{2}\left[\vec{\mathcal{E}}(\vec{x},t) + \mathrm{c.c.}\right] \quad ,\vec{\mathcal{E}} \text{ is the analytic signal of } \vec{\mathbb{E}}\\ &= \sum_p\int\mathrm{d}\omega\frac{1}{2}\left\{\frac{\vec{F}_p(x,y,\omega)}{N_p(\omega)}A_p(z,\omega)\mathrm{e}^{i\left[\beta_p(\omega)z-\omega t\right]} + \mathrm{c.c.}\right\}.\end{aligned} \tag{1.6}$$

This makes MATLAB "ifft" become the Fourier Transform and "fft" become the *inverse* Fourier Transform. The convolution theorem also becomes different with different conventions. In our package, we follow this convention [Eq. (1.6)]. For example, to see the spectrum, please use

```
c = 299792.458; % nm/ps
wavelength = c./f(f>0); % nm
Nt = size(field,1);
dt = t(2)-t(1); % ps
factor_correct_unit = (Nt*dt)^2/1e3; % to make the spectrum of the correct unit "nJ/THz"
                                     % "/1e3" is to make pJ into nJ
spectrum = abs(fftshift(ifft(field),1)).^2*factor_correct_unit; % in frequency domain
```



Cornell University

Use "fftshift" to shift the spectrum from small frequency to large frequency. Note that it is not "ifftshift". They differ when the number of points is odd. To understand this, think about what the first data point is in "ifft(field):" it is the zero-frequency component, so we need to use "fftshift" with the frequency defining as

```
f = f0+(-Nt/2:Nt/2-1)'/(Nt*dt); % THz
```

About the photoionization term, we implement only the single-mode case at this moment.

$$\begin{aligned}\partial_z A(z,\Omega) = i\left[\beta(\omega) - \left(\beta_{(0)} + \beta_{(1)}\Omega\right)\right] A(z,\Omega) \\ + \frac{i\omega}{4\left[N(\omega)\right]^4 A_{\text{eff}}(\omega)}\left(\frac{3}{4}\epsilon_0 \chi^{(3)}_{\text{electronic}} \mathfrak{F}\left[|A|^2 A\right] + \mathfrak{F}\left[A\left(R * |A|^2\right)\right]\right) \\ - i\frac{e^2}{4\left[N(\omega)\right]^2 \omega m_e}\mathfrak{F}\left[n_e(T)A(T)\right] - \frac{E_b A_{\text{eff}}}{4}\mathfrak{F}\left[\frac{\partial n_e}{\partial T}\frac{A(T)}{|A(T)|^2}\right],\end{aligned} \tag{1.7}$$

where $e = 1.602\,176\,63 \times 10^{-19}\,\text{C}$ is the electron charge, $m_e = 9.109\,383\,7 \times 10^{-31}\,\text{kg}$ is the electron mass, and E_b is the ionization energy. n_e is the free-electron density that satisfies

$$\partial_t n_e = W\left(N_g - n_e\right) \approx W N_g \tag{1.8}$$

with W being the ionization rate and N_g being the total neutral-gas density, such that

$$n_e(t) \approx N_g \int_{-\infty}^{t} W(\tau)\,\mathrm{d}\tau \quad , n_e(t \to -\infty) = 0 \tag{1.9}$$

The ionization rate W can be calculated with the Perelomov-Popov-Terent'ev model.

For more details, please read our papers on soliton self-frequency shift [3], long-wave-infrared generation [4], and unified theory of Raman scattering. In particular, their supplements include all the details required for gas-based fiber simulations.

1.1 High-level understanding of this package

This package is designed for both single-spatial(transverse) mode or multi-spatial modes. Not only scalar but also polarized fields can be simulated, as well as Raman scattering. The package exhibits an adaptive control of the step size. In addition to CPU, highly parallelized cuda computation with a Nvidia GPU is implemented, which is strongly recommended for running with multimodes. In single-mode simulations, for sampling numbers less than approximately 2^{25}, they can still run faster with CPU than with GPU.

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

Chapter 2

Use

Some examples of how to use the this package to simulate pulses can be found in the "Examples" folder. Generally, the workflow is as follows:

1. **Determine the number of discrete points N_t, time spacing $\mathrm{d}t$, and center frequency f_0.**

   They determine the time window and the frequency window. Because oftentimes the simulation covers a huge spectral window, the center frequency of the window cannot be at the pulse center frequency; otherwise, the edge of the frequency window can become negative. UPPE is based on *analytic signal* that includes only signals with positive frequencies. Because simulating with frequency window into the negative frequencies involves the interaction of these negative-frequency spectral components, it can generate a wrong result. It is thus important to shift the center frequency to maintain positiveness of the frequency window.

2. **Setup simulation parameters (three structure variables): sim, fiber, and gas.**

   "sim" includes simulation parameters such as whether to use GPU or not, whether to center the pulse temporally during propagation (the delay is saved for users to recover), what the moving speed of a time window is, and so on.
   "fiber" includes the fiber parameters such as fiber length and propagation constants of the involving modes.
   "gas" includes gas parameters such as gas pressure, gas type, and all the other relevant parameters of hollow-core fibers.

3. **Setup the initial condition which corresponds to input pulses.**

4. **Run the UPPE_propagate() function to start the pulse propagation.**


CORNELL UNIVERSITY
FOUNDED A.D. 1865

Chapter 3

Before I go deeply into details

3.1 Introduction

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

Below is how to call this function in general.

```
prop_output = UPPE_propagate(fiber, ...
                             initial_condition, ...
                             sim, ...
                             gas)
```

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

fiber
: It contains the information of the fiber, such as β and S^R, etc.

initial_condition
: It contains the information of the input field.

Figure 3.1: Initial conditions

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

gas
: It contains the information required to run gas simulations, such as gas nonlinear refractive index and the information of its Raman response functions.


CORNELL UNIVERSITY
FOUNDED A.D. 1865

Chapter 4

Input arguments

Below, I use N_t as the number of time/frequency sampling points, N_m as the number of modes, and N_{sm} as the number of spatial modes. If there is no polarized mode, $N_m = N_{sm}$; otherwise, $N_m = 2N_{sm}$.

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 labelled in blue the parameters required all the time.

4.1 fiber

betas

betas are column vectors of $\beta(\omega)$, which follows

$$\begin{bmatrix} \beta_1 & \beta_2 & \cdots \end{bmatrix}, \tag{4.1}$$

where each β_m is a column vector of the propagation constant of the mth mode. It's a function of frequencies, ordered from small to large. Don't take "ifftshift" for this variable. If the simulation is run with N_t time/frequency points, this β_m should have the length of N_t as well.

SR

It's the overlap integral $S^R_{p\ell mn} = \int F_p F_\ell F_m F_n \, \mathrm{d}^2x$ in scalar UPPE. Its dimension is N^4_{sm}.

Since $N_p N_\ell N_m N_n = \frac{\epsilon_0^2 n_{\mathrm{eff}}^2 c^2}{4}$ is a constant value independent of modes, only the integral part needs to be calculated. Furthermore, all overlap integrals can be calculated from $S^R_{p\ell mn}$. For example, in scalar situations, $Q^K_{p\ell mn} = S^K / \left(N_p N_\ell N_m N_n\right) = S^R_{p\ell mn} / \left(N_p N_\ell N_m N_n\right)$ if the field is linearly polarized or $\frac{2}{3} S^R_{p\ell mn} / \left(N_p N_\ell N_m N_n\right)$ is it's circularly polarized. The unit is m^{-2}. It's more complicated with a polarized field, whose $Q^R_{p\ell mn}$ and $Q^K_{p\ell mn}$ can also be calculated from the $S^R_{p\ell mn}$. This will be done by UPPE_propagate() automatically if "sim.scalar=false." For details, check Chap.6.

L0

This is the fiber length. The unit is meter.

4.2 initial_condition

dt

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

fields

This is the input field amplitude under the time domain. It has the unit of $\sqrt{\mathrm{W}}$. Its size is $N_t \times N_m$.

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

4.3 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 beating, the size of the z-step cannot be too large.

This "betas" is a 2×1 column vector.

$$\begin{bmatrix} \beta_{(0)} \\ \beta_{(1)} \end{bmatrix} \tag{4.2}$$

The "betas" is found such that it minimizes the max $\left(\left| \beta_1(\omega) - \left(\beta_{(0)} + \beta(1)\omega \right) \right| \right)$ of the 1st mode in the frequency window if this "sim.betas" is left empty (sim.betas=[]).

f0

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

midx

The mode index. It's an integer array. It's 1 for scalar simulations.

X3

It's the electronic third-order susceptibility of the gas.

save_period

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

4.3.1 Mode info

Here are the information for mode profiles.

mode_profiles.mode_profiles
2D mode profiles of each mode.

mode_profiles.norms
This is the norms of the mode profiles.

mode_profiles.r

mode_profiles.dr

mode_profiles.dtheta

4.3.2 MPA

Here are the parameters if the simulation uses MPA step method. All parameters are contained within a "sim.MPA" structure.

MPA.M
This is the parallel extent for MPA. 1 is no parallelization. 5-20 is recommended; there are strongly diminishing returns after 5-10. 10 is recommended.

MPA.n_tot_max
The maximum number of iterations for MPA. This doesn't really matter because if the step size is too large, the algorithm will diverge after a few iterations. 20 is a typical number for this.

MPA.n_tot_min
The minimum number of iterations for MPA. Use 2 for this parameter.

MPA.tol
The tolerance of convergence for MPA, which is related to the values of the average NRMSE between consecutive itertaions in MPA at which the step is considered converged.

4.3.3 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

If the simulation is solved with "sim.scalar=true," the input field takes only the scalar fields, e.g.,

$$\begin{bmatrix}\text{mode 1} & \text{mode 2} & \text{mode 3} & \dots\end{bmatrix}.$$

Cornell University

Otherwise, the input field of each polarized mode needs to be specified in the order of

$$\left[\text{mode } 1_+ \quad \text{mode } 1_- \quad \text{mode } 2_+ \quad \text{mode } 2_- \quad \ldots\right],$$

where (+,-) can be (x,y), (right-handed circular, left-handed circular), or any orthogonally polarized modes.

Based on whether to include polarization-mode coupling, S^R and S^K are automatically calculated to its polarized version by UPPE_propagate().

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)

4.3.4 Adaptive-step method

Here are the parameters for adaptive-step method. All parameters are contained within a "sim.adaptive_deltaZ" structure.

adaptive_deltaZ.threshold

The threshold of the adaptive-step method. It controls the accuracy of the simulation and determines whether to increase or decrease the step size. I typically use 10^{-8}.

adaptive_deltaZ.max_deltaZ

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

4.3.5 Algorithm to use

gpu_yes

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

Raman_model

0	ignore Raman effect
1	use Raman model for Raman-active gases

photoionization_model

Photoionization model is implemented currently only in single-mode scenarios.

0	ignore the photoionization effect
1	include the photoionization effect

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

num_photon_noise_per_bin

This controls the number of photon noise to include per spectral discretization bin. It's default to 1.

$$P_{\text{noise}} = hf/\left(N\triangle t\right), \tag{4.3}$$

whose relation depends on the convention of Fourier Transform. For convention of the laser field, Fourier Transform is defined as MATLAB's "ifft," which leads to the relation here. With a different convention, the multiplication factor might be different from $1/\left(N\triangle t\right)$.

cuda_dir_path

The path to the cuda directory into which ptx files will be compiled and stored. This is "/UPPE/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 UPPE shown on the progress bar. If not set (no "sim.progress_bar_name"), it uses a default empty string, ''.

Cornell University

Chapter 5

Output arguments

fields
The $N_t \times N_m \times N_z$ output fields.

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

z
This is the positions of each saved field.

deltaZ
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.

delta_permittivity
Raman-induced permittivity change $\triangle\epsilon$. Its size is $N_t \times N_m \times N_R \times N_z$, where N_R is the number of Raman types. This is implemented only in scalar simulations.

We have implemented a few gases whose N_R dimension follows:

H_2, N_2, O_2	$\left[\triangle\epsilon^{\text{vib}} \quad \triangle\epsilon^{\text{rot}}\right]$
air	$\left[\triangle\epsilon^{\text{vib }(N_2)} \quad \triangle\epsilon^{\text{rot }(N_2)} \quad \triangle\epsilon^{\text{vib }(O_2)} \quad \triangle\epsilon^{\text{rot }(O_2)}\right]$
CH_4	$\left[\triangle\epsilon^{\text{vib}}\right]$

$\triangle\epsilon^{\text{vib}}$ is calculated from Eq. (1.5a) of each gas, while $\triangle\epsilon^{\text{rot}}$ is calculated from Eq. (1.5b) for circularly polarized fields but its 4 times for linearly polarized fields. See our paper of a unified theory for details.

relative_Ne

The ionized electron density ($1/m^3$).

Chapter 6

Polarization modes

If the simulation is solved with "sim.scalar=true", the input field takes only the scalar fields, e.g.,

$$\Big[\text{mode 1} \quad \text{mode 2} \quad \text{mode 3} \quad \ldots\Big].$$

Otherwise, the input field of each polarized mode needs to be specified in the order of

$$\Big[\text{mode } 1_+ \quad \text{mode } 1_- \quad \text{mode } 2_+ \quad \text{mode } 2_- \quad \ldots\Big],$$

where (+,-) can be (x,y), (right-handed circular, left-handed circular), or any orthogonally polarized modes.

If the input β has a dimension of only the number of spatial modes, N_{sm}, I assume there's no significant influence from birefringence; thus, it's expanded into $2N_{sm}$ dimension with each i and j (polarization) modes being degenerate by UPPE_propagate(). For polarized fields,

$$S^R_{plmn} = \frac{\int \mathrm{d}x\,\mathrm{d}y\Big[\mathbf{F}^*_p \cdot \mathbf{F}_l\Big]\Big[\mathbf{F}^*_m \cdot \mathbf{F}_n\Big]}{\left[\left(\int \mathrm{d}x\,\mathrm{d}y|\mathbf{F}_p|^2\right)\left(\int \mathrm{d}x\,\mathrm{d}y|\mathbf{F}_l|^2\right)\left(\int \mathrm{d}x\,\mathrm{d}y|\mathbf{F}_m|^2\right)\left(\int \mathrm{d}x\,\mathrm{d}y|\mathbf{F}_n|^2\right)\right]^{1/2}} \tag{6.1}$$

$$S^K_{plmn} = \frac{2}{3}S^R_{plmn} + \frac{1}{3}\frac{\int \mathrm{d}x\,\mathrm{d}y\Big[\mathbf{F}^*_p \cdot \mathbf{F}^*_n\Big]\Big[\mathbf{F}_m \cdot \mathbf{F}_l\Big]}{\left[\left(\int \mathrm{d}x\,\mathrm{d}y|\mathbf{F}_p|^2\right)\left(\int \mathrm{d}x\,\mathrm{d}y|\mathbf{F}_l|^2\right)\left(\int \mathrm{d}x\,\mathrm{d}y|\mathbf{F}_m|^2\right)\left(\int \mathrm{d}x\,\mathrm{d}y|\mathbf{F}_n|^2\right)\right]^{1/2}} \tag{6.2}$$

Therefore, S^R_{plmn} isn't zero as (p,l) and (m,n) both have the same polarization, and we get four possibilities for (p,l,m,n), (0,0,0,0), (0,0,1,1), (1,1,0,0), and (1,1,1,1), with their values directly derived from the scalar S^R_{plmm}. For S^K_{plmn}, in addition to the permutations of S^R_{plmn}, we need to consider those from the fraction above which isn't zero as (p,l,m,n) is (0,0,0,0), (0,1,1,0), (1,0,0,1), and (1,1,1,1). Notice that some of them can add up with S^R_{plmn} while some of them can't, so the value has a prefactor of $1, \frac{2}{3}, \frac{1}{3}$.

The above generalization of the scalar S^R to polarized S^R, S^K needs each $\mathbf{F}_p$ to be either parallel or orthogonal to one another, so (i,j) has to be an orthogonal group in 2D, e.g., (x, y) or (σ_+, σ_-).

Chapter 7

Information and computation of gas and hollow-core fiber

This package implements two types of hollow-core fibers: anti-resonant fiber and capillary. Dispersion and mode-profile calculations of anti-resonant fiber is based on the poorman's model by Bache *et al.* [1], while that of capillary is based on the equation derived by Marcatili [2].

7.1 Gas info in simulations

To use the model of anti-resonant fiber, follow (the numeric values are chosen as examples. Modify them based on your needs.):

```
%% Gas info
[fiber,sim] = load_default_UPPE_propagate(fiber,sim);

gas.core_radius = 15e-6; % um
gas.temperature = 288; % K
gas.pressure = 15*1.01325e5; % Pa
gas.wavelength_order = 6; % keep this 6 all the time
gas.mode_profile_wavelength = 1030e-9; % m
gas.gas_material = 'H2';
gas.delta = 300e-9; % m; wall thickness of anti-resonant fibers
gas.f_FEM = 2e-3; % loss factor (check poorman's paper for detail)
gas.fiber_type = 'AR_HC_PCF';
gas.xy_sampling = 101; % the number of sampling points for the mode profiles,
    which affects calculationo of "norms" and "SR"

[fiber,sim,gas] = gas_info(fiber,sim,gas,lambda*1e-9);
```

We have implemented a few gases (for "gas.gas_material"): H_2, N_2, O_2, air, Xe, Ar, Ne, He, Kr, and CH_4. Raman models have been implemented for Raman-active gases: H_2, N_2, O_2, air, and CH_4. Photoionization has been implemented for all gases except air.

To use the model of capillary, follow (the numeric values are chosen as examples. Modify them based on your needs.):

```
%% Gas info
[fiber,sim] = load_default_UPPE_propagate(fiber,sim);

gas.core_radius = 250e-6; % m
```

```
gas.temperature = 300; % K
gas.pressure = 6.2*1.01325e5; % Pa
gas.wavelength_order = 6;
gas.mode_profile_wavelength = 1030e-9; % m
gas.gas_material = 'N2';
gas.fiber_type = 'no_coating';
gas.xy_sampling = 101; % the number of sampling points for the mode profiles,
    which affects calculationo of "norms" and "SR"

[fiber,sim,gas] = gas_info(fiber,sim,gas,lambda*1e-9);
```

In Marcatili's model, we can change the dielectric material of capillary. Based on this, we have implemented a few coatings, so "gas.fiber_type" has a few options:

'Ag_coating'	The capillary material is Ag
'no_coating'	The capillary material is silica
'MWLW_coating'	The capillary material is also silica, but the final "waveguide" loss is later entirely neglected.
'waveguide_loss_only'	The capillary material is silica whose loss is initially neglected. The result includes waveguide loss without exacerbated by silica loss.

7.2 Hollow-core fiber

It's sometimes helpful to know the fiber parameters separately from simulations. For example, the simulations compute based on propagation constants that are functions of frequency, but we sometimes would like to know the dispersion, $\frac{\mathrm{d}^2\beta}{\mathrm{d}\omega^2}$.

To compute these parameters, use the scripts with the name, "solve_for_EH_...()" in the folder "HCF/."

Use "calc_betaN()" to compute for dispersion.

Chapter 8

Simulations with multiple spatial modes

This package allows users to simulate multimode pulse propagation. This is enabled by setting "sim.midx":

```
sim.midx = 1:6; % mode 1 to 6 (LP01,LP02,...,LP06) are included for example
```

The order of mode indices follows the decreasing magnitudes of propagation constants. By default, this package is restricted to the first six circularly symmetric modes ($LP_{01} \dots LP_{06}$). The restriction can be easily modified in the HCF functions:

1. HCF/helper_functions/solve_for_EH_Ag_coating_beta_func.m
2. HCF/helper_functions/solve_for_EH_AR_HC_PCF_beta_func.m
3. HCF/helper_functions/solve_for_EH_MWLW_coating_beta_func.m
4. HCF/helper_functions/solve_for_EH_no_coating_beta_func.m

They have the following lines

```
% (n,m) modes to solve for
user_midx = [1,4,9,17,28,40]; % a maximum of 6 circular symmetric modes included
    here
user_midx = user_midx(sim.midx);
num_modes = length(user_midx);
```

Feel free to modify "user_midx" to include other modes. For example, user_midx=2 and 3 corresponds to LP_{11a} and LP_{11b}. Readers can plot the fields out with "solve_for_EH_...()" in the folder "HCF/" to see if they pick those they want. They have the same lines that can be modified as below:

```
% (n,m) modes to solve for
% [1,4,9,17,28,40] are the first six radial EH0m modes
user_midx = 1;%[1,4,9,17,28,40];
num_modes = length(user_midx);
```

The examples of multimode simulations are in "Examples/validation of the code/Extreme Raman redshift (Carpeggiani's Optica, 2020) (use capillary)," which indludes the first five circularly symmetric modes.

Bibliography

[1] M. Bache, M. S. Habib, C. Markos, and J. Lægsgaard, "Poor-man's model of hollow-core anti-resonant fibers", J. Opt. Soc. Am. B **36**, 69–80 (2019).

[2] E. A. J. Marcatili and R. A. Schmeltzer, "Hollow metallic and dielectric waveguides for long distance optical transmission and lasers", Bell Syst. Tech. J. **43**, 1783–1809 (1964).

[3] Y.-H. Chen, P. Sidorenko, E. Antonio-Lopez, R. Amezcua-Correa, and F. W. Wise, "Efficient soliton self-frequency shift in hydrogen-filled hollow-core fiber", Opt. Lett. **47**, 285–288 (2022).

[4] Y.-H. Chen, J. Moses, and F. Wise, "Femtosecond long-wave-infrared generation in hydrogen-filled hollow-core fiber", J. Opt. Soc. Am. B **40**, 796–806 (2023).