Naval Research Laboratory

Washington, DC 20375-5320

NRL/MR/6795--20-10,121

SNOPROP: A Solver for Nonlinear Optical Propagation

J. Ryan Peterson

Physics Department, Stanford University
Stanford, CA

Theodore G. Jones

Bahman Hafizi

Luke A. Johnson

Daniel F. Gordon

Directed Energy Physics Branch
Plasma Physics Division

July 23, 2020

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
23-07-2020	NRL Memorandum Report	October 2019 – July 2020

4. TITLE AND SUBTITLE

SNOPROP: A Solver for Nonlinear Optical Propagation

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
61553N

6. AUTHOR(S)

J. Ryan Peterson*, Theodore G. Jones, Bahman Hafizi, Luke A. Johnson, and Daniel F. Gordon

5d. PROJECT NUMBER

5e. TASK NUMBER
EL011-09-41

5f. WORK UNIT NUMBER
1R03

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
4555 Overlook Avenue, SW
Washington, DC 20375-5320

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/MR/6795--20-10,121

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
875 N. Randolph Street, Suite 1425
Arlington, VA 22203-1995

10. SPONSOR / MONITOR'S ACRONYM(S)

ONR Code 08

11. SPONSOR / MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A: Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

*Physics Department, Stanford University, 450 Serra Mall, Stanford, CA 94305
SLAC National Accelerator Laboratory, 2575 Sand Hill Rd, Menlo Park, CA 94025

14. ABSTRACT

The Solver for Nonlinear Optical Propagation (SNOPROP) code models the interaction of intense lasers with background media in two dimensions. This report derives the envelope propagation equations for an intense laser beam including self- and cross-phase modulation, first-order Stokes and anti-Stokes stimulated Raman scattering, and four-wave mixing. Ionization is included using a multiphoton and collisional ionization model. We also discuss the implementation of the solver and the numerical methods used as well as demonstrate a basic example of underwater laser propagation featuring the effects mentioned above.

15. SUBJECT TERMS

Nonlinear optics Lasers Laser propagation
Raman scattering Simulation

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Theodore G. Jones
Unclassified Unlimited	Unclassified Unlimited	Unclassified Unlimited	Unclassified Unlimited	33	**19b. TELEPHONE NUMBER** *(include area code)* (202) 767-3379

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

This page intentionally left blank.

CONTENTS

FIGURES

TABLES

This page intentionally left blank.

EXECUTIVE SUMMARY

The Solver for Nonlinear Optical Propagation (SNOPROP) code models the interaction of intense lasers with background media in two dimensions. This report derives the envelope propagation equations for an intense laser beam including self- and cross-phase modulation, first-order Stokes and anti-Stokes stimulated Raman scattering, and four-wave mixing. Ionization is included using a multiphoton and collisional ionization model. We also discuss the implementation of the solver and the numerical methods used as well as demonstrate a basic example of underwater laser propagation featuring the effects mentioned above.

This page intentionally left blank.

SNOPROP: A SOLVER FOR NONLINEAR OPTICAL PROPAGATION

1. INTRODUCTION

Intense laser pulses experience many nonlinear effects when propagating through dielectrics. Among others, these include self-focusing and stimulated Raman scattering (SRS) which both become important when megawatt beam powers focus in condensed media. As Raman Stokes light is generated by SRS, the resulting beams are subject to four-wave mixing which can create Raman anti-Stokes light under the proper conditions. Ionization through multiple mechanisms is often strong at these intensities and can greatly affect propagation. Many of these effects can simultaneously have a strong impact on the beam propagation. Solving for the nonlinear propagation of a single beam is a daunting task analytically, and the interaction of three beams becomes analytically intractable very quickly; three nonlinear terms for a single beam becomes 24 nonlinear terms with three beams. To analyze the propagation characteristics we must therefore resort to a numerical approach for solving the complete set of propagation equations.

Computational methods have a long history in nonlinear laser propagation. While solving the full set of vector Maxwell's equations in multiple dimensions can be prohibitively slow for many problems, various approximations have been developed which retain the essential physics with much less demanding computational requirements. The unidirectional pulse propagation equation (UPPE), which only restricts the solution to forward-propagating waves in the vector Maxwell's equations, can model short pulses over long distances [1]. Envelope approximations may be derived from the UPPE or Maxwell's equations directly and neglect rapid changes in the pulse profile over distances comparable to the wavelength. Envelope solvers have been used to successfully model both short (fs) and long (ns) beams over large distances [2, 3]. Particle dynamics may even be included in envelope solvers for beams propagating in tenuous plasmas and permit much larger simulation sizes than full particle-in-cell Maxwell solvers would allow [4]. In our case, the higher-order effects captured by UPPE are unneccessary and particle dynamics are irrelevant, so we use an envelope solver.

Envelope propagation equations may account for all of the nonlinear effects mentioned above and we will build on their successfull use elsewhere [2, 3, 5]. These solvers have been shown to accurately reproduce experimental data in several experiments involving intense laser propagation through both air and water [6–9]. From the physics perspective we expand these previous studies significantly by including a larger set of propagation equations, enabling simulations of a far more complex set of interactions.

In this report, we derive paraxial slowly-varying envelope propagation equations for the interaction of a laser, Raman Stokes, and Raman anti-Stokes beams. We include group delay, group velocity dispersion, first-order SRS, four-wave mixing, self- and cross-phase modulation, multiphoton ionization (MPI), and collisional ionization. Afterwards, we discuss the implementation of and demonstrate use of our simulation code SNOPROP: A Solver for Nonlinear Optical Propagation.

Manuscript approved July 17, 2020.

2. PROPAGATION EQUATIONS

We begin by assuming that our electric field $\mathbf{E}(\vec{r},t)$ is linearly polarized in the x-direction and travelling approximately in the z-direction. This field is composed of three frequencies and written as $\mathbf{E}(\vec{r},t) = \sum_{n=S,L,A} \mathbf{E}_n(\vec{r},t)$, where the subscript $n = L$ is the laser field, $n = S$ is the Raman Stokes field, and $n = A$ is the Raman anti-Stokes field. We let each field propagate with a midband frequency ω_n and midband wavenumber $k_n = \beta(\omega_n)$. We assume that each frequency satisfies the nonlinear wave equation

$$\left(\nabla_\perp^2 + \frac{\partial^2}{\partial z^2} - \frac{1}{c^2}\frac{\partial^2}{\partial t^2}\right)\mathbf{E}_n(\vec{r},t) = \mathbf{S}_n^L(\vec{r},t) + \mathbf{S}_n^{NL}(\vec{r},t) \tag{1}$$

where $\mathbf{S}_n^L(\vec{r},t)$ and $\mathbf{S}_n^{NL}(\vec{r},t)$ represent source terms with frequency ω_n that are respectively linear and nonlinear in the electric field. From here on, the explicit space and time dependence of fields and sources will be dropped. The source terms relate to the linear and nonlinear polarization $\mathbf{P}_n^L$ and $\mathbf{P}_n^{NL}$ as $\varepsilon_0 c^2 \mathbf{S}_n^L = \partial^2 \mathbf{P}_n^L/\partial t^2$ and $\varepsilon_0 c^2 \mathbf{S}_n^{NL} = \partial^2 \mathbf{P}_n^{NL}/\partial t^2$ respectively.

Next, we introduce envelopes for the electric field and source terms in the form

$$\begin{aligned}
\mathbf{E}_n &= \mathbf{e}_x\left(A_n e^{i(\beta(\omega_n)z-\omega_n t)} + \text{c.c.}\right)\\
\mathbf{S}_n^L &= \mathbf{e}_x\left(S_n^L e^{i(\beta(\omega_n)z-\omega_n t)} + \text{c.c.}\right)\\
\mathbf{S}_n^{NL} &= \mathbf{e}_x\left(S_n^{NL} e^{i(\beta(\omega_n)z-\omega_n t)} + \text{c.c.}\right)
\end{aligned} \tag{2}$$

where $\mathbf{e}_x$ is a unit vector in the x-direction and represents the matching linear polarization of each beam. The linear source amplitude S_n^L includes media effects such as the group velocity and group velocity dispersion (GVD). We include only up to second-order dispersive effects. Inserting S_n^L gives the propagation equation [2]

$$\left(\nabla_\perp^2 + 2ik_n\frac{\partial}{\partial z} + \frac{\partial^2}{\partial z^2} + 2ik_n\frac{1}{v_{gn}}\frac{\partial}{\partial t} - \left(\frac{1}{v_{gn}^2} + k_n\beta_n''\right)\frac{\partial^2}{\partial t^2}\right)A_n = S_n^{NL} \tag{3}$$

for our envelope fields with frequency-dependent wavenumber $\beta(\omega)$ and group velocity at frequency ω_n given by $v_{gn} = 1/\beta_n'$ with parameters β_n' and β_n'' given by

$$\begin{aligned}
\beta_n' &= \left.\frac{\partial\beta(\omega)}{\partial\omega}\right|_{\omega=\omega_n},\\
\beta_n'' &= \left.\frac{\partial^2\beta(\omega)}{\partial\omega^2}\right|_{\omega=\omega_n}.
\end{aligned} \tag{4}$$

The group index is then simply $n_{gn} = c/v_{gn}$.

The next task is to determine the nonlinear response S_n^{NL} at each frequency ω_n. We set $S_n^{NL} = S_n^{\text{Raman}} + S_n^{\text{Kerr}} + S_n^{\text{plasma}} + S_n^{\text{ion}}$ and discuss each of these contributions in turn.

2.1 Stimulated Raman Scattering

As an intense laser pulse propagates through dielectric materials, the material vibrations modulate the optical field and produce sidebands in the optical field separated from the laser by the vibrational frequency ω_v. These are known as the Stokes sideband (with frequency $\omega_S = \omega_L - \omega_v$) and anti-Stokes sideband ($\omega_A = \omega_L + \omega_v$). Typical vibration frequencies include liquid water at 102 THz [10] and N_2 gas at 69.8 THz [11]. These sidebands then beat with the laser electric field at the vibrational frequency and increase the vibration amplitude which scatters more light into the Stokes sideband. The cycle repeats, and a large fraction of the laser energy can eventually be converted to Stokes energy in a process known as Stimulated Raman Scattering.

The nonlinear polarization for SRS in an isotropic medium can be calculated through a classical [11] or quantum [12] approach. We here include pump, Stokes, and anti-Stokes together and allow all three fields, including the pump, to vary. In the result, we ignore all but the terms oscillating at the laser, Stokes, and anti-Stokes frequencies. We also ignore the non-resonant interactions as they are strongly absorbed due to the significant real component of their susceptibility. This yields the Raman source terms

$$\begin{aligned} S_S^{\text{Raman}} &= -6\frac{\omega_S^2}{c^2}\left(\chi_{RS}|A_L|^2 A_S + \chi_{RS} A_L^2 A_A^* e^{i\Delta k z}\right), \\ S_L^{\text{Raman}} &= -6\frac{\omega_L^2}{c^2}\left(\chi_{RS}|A_A|^2 A_L + \chi_{RA}|A_S|^2 A_L\right), \\ S_A^{\text{Raman}} &= -6\frac{\omega_A^2}{c^2}\left(\chi_{RA}|A_L|^2 A_A + \chi_{RA} A_L^2 A_S^* e^{i\Delta k z}\right), \end{aligned} \tag{5}$$

where $\Delta k = 2k_L - k_S - k_A$, χ_{RS} is the third-order Raman Stokes susceptibility, $\chi_{RA} = \chi_{RS}^*$ (ignoring dispersive effects) is the third-order Raman anti-Stokes susceptibility, and we have assumed that $\partial^2 \mathbf{P}_n^{NL}/\partial t^2 \approx -\omega_n^2 \mathbf{P}_n^{NL}$. The $A_L^* A_S A_A$ term in S_L^{Raman} has dropped out due to our assumption of $\chi_{RA} = \chi_{RS}^*$. Since χ_{RS} is purely imaginary with a positive imaginary part, SRS is a pure-gain process for the Stokes wave and a pure-loss process for the anti-Stokes wave.

Four-wave mixing is captured by the term with $e^{i\Delta k z}$ dependence and allows energy transfer between the Stokes and anti-Stokes waves. Because of the normal dispersion inherent in most cases of interest, we usually have $\Delta k \neq 0$. Efficient energy transfer from Stokes to anti-Stokes beam requires the phase matching condition that S_A^{Raman} have the same z-dependence as A_A. This is the same as requiring that a system of plane waves with the Stokes, laser, and anti-Stokes beams travelling with wavevectors $\mathbf{k_S}$, $\mathbf{k_L}$, and $\mathbf{k_A}$ satisfy the phase-matching condition

$$2\mathbf{k_L} - \mathbf{k_S} - \mathbf{k_A} = 0. \tag{6}$$

This can only be satisfied when the laser and Stokes waves travel at an angle relative to one another. For maximum anti-Stokes gain, the angle between the laser and Stokes waves is

$$\theta_S = \cos^{-1}\left(\frac{k_S^2 + 4k_L^2 - k_A^2}{4k_S k_L}\right) \tag{7}$$

which will produce anti-Stokes light at the angle

$$\theta_A = \cos^{-1}\left(\frac{k_A^2 + 4k_L^2 - k_S^2}{4k_A k_L}\right) \tag{8}$$

in the other direction relative to the laser pulse. This can be an angle of a few degrees in condensed media. In water with a laser wavelength of 355 nm, the angles are approximately $\theta_S = 3.13$ degrees and $\theta_A = 2.42$ degrees.

2.2 The Kerr effect

Intense optical pulses are also subject to self- and cross-phase modulation—also known as the optical Kerr effect. The strong electric field at the center of an intense optical pulse can locally increase the refractive index due to anharmonic response of bound electrons. This effect can focus the intense beam itself (self-phase modulation) or another beam at a different frequency (cross-phase modulation). While Ref. [2] includes only self-phase modulation, we look to Ref. [12] for the cross-phase modulation. Our method ignores dispersion in the Kerr effect by assuming a constant nonresonant susceptibility χ_{NR} for each wavelength. This yields the nonlinear source terms due to the Kerr effect

$$\begin{aligned}
S_S^{\text{Kerr}} &= -6\frac{\omega_S^2}{c^2}\left(\frac{1}{2}\chi_{NR}|A_S|^2A_S + \chi_{NR}|A_L|^2A_S + \chi_{NR}|A_A|^2A_S + \chi_{NR}A_L^2A_A^* e^{i\Delta kz}\right), \\
S_L^{\text{Kerr}} &= -6\frac{\omega_L^2}{c^2}\left(\chi_{NR}|A_S|^2A_L + \frac{1}{2}\chi_{NR}|A_L|^2A_L + \chi_{NR}|A_A|^2A_L + 2\chi_{NR}A_L^*A_SA_A e^{-i\Delta kz}\right), \\
S_A^{\text{Kerr}} &= -6\frac{\omega_A^2}{c^2}\left(\chi_{NR}|A_S|^2A_A + \chi_{NR}|A_L|^2A_A + \frac{1}{2}\chi_{NR}|A_A|^2A_A + \chi_{NR}A_L^2A_S^* e^{i\Delta kz}\right).
\end{aligned} \tag{9}$$

2.3 Ionization and plasma effects

Intense laser pulses may ionize the media through which they propagate. The plasma generated by a laser also serves to defocus the laser in contrast to the self-focusing from the Kerr effect. This counteracting tendency between self-focusing and plasma defocusing is necessary to generate large-scale optical filaments in water [8].

We include multiphoton and collisional ionization. In this MPI model, the number of photons to ionize an atom is assumed constant for each of the three beams. The electron density N_e and MPI rate ν_{MPI} obey the equations [2, 13]

$$\frac{\partial N_e}{\partial t} = \nu_{\text{MPI}}N_0 + \nu_i N_e - \eta N_e \tag{10}$$

$$\nu_{\text{MPI}} = \sum_{n=S,L,A} \nu_{\text{MPI},n} \tag{11}$$

$$\nu_{\mathrm{MPI},n} = \frac{2\pi}{(l-1)!}\omega_n \left[\frac{I_n}{I_{\mathrm{MPI}}}\right]^l \tag{12}$$

where ν_{MPI} is the total MPI rate, N_0 the density of neutral atoms or molecules, ν_i the avalanche ionization rate, η the electron loss rate, $\nu_{\mathrm{MPI},n}$ the MPI rate exclusively via photons with frequency ω_n, l the number of photons required to ionize, and I_{MPI} the MPI threshold intensity. The avalanche ionization rate ν_i is then defined as [14–16]

$$\nu_i = \frac{\nu_e}{U_{\mathrm{ion}}}\frac{2e^2}{m_e}\left(\frac{|A_S|^2}{\omega_S^2} + \frac{|A_L|^2}{\omega_L^2} + \frac{|A_A|^2}{\omega_A^2}\right) \tag{13}$$

$$\nu_e = N_0 v_{\mathrm{rms}} \sigma_c \tag{14}$$

$$v_{\mathrm{rms}} = \frac{e\sqrt{2}}{m_e}\sqrt{\frac{|A_S|^2}{\omega_S^2} + \frac{|A_L|^2}{\omega_L^2} + \frac{|A_A|^2}{\omega_A^2}} \tag{15}$$

with U_{ion} the ionization energy, e (>0) the elementary charge, m_e the electron mass, ν_e the electron collision frequency, v_{rms} the RMS electron velocity, and σ_c and the electron-neutral collision cross section.

Next we consider plasma refraction and energy depletion by plasma heating and ionization. Refraction as well as electric field energy loss (Joule heating and inverse bremsstrahlung) via collisional ionization are included via the term [3]

$$S_n^{\mathrm{plasma}} = \frac{\omega_p^2}{c^2}\left(1 - \frac{i\nu_e}{\omega_n}\right)A_n, \tag{16}$$

and MPI losses are included through the term

$$S_n^{\mathrm{ion}} = -\frac{i\omega_n}{2c^2}\frac{U_{\mathrm{ion}}}{|A_n|^2}N_0\nu_{\mathrm{MPI},n}A_n. \tag{17}$$

In low-density media, U_{ion} represents the ionization energy to free an electron of mass m_e from the neutral atom. In condensed media, we can model ionization as excitation of electrons from a valence band to a conduction band with energy U_{ion}. These conduction band electrons may have an effective mass m_{eff} different than the electron rest mass due to a lattice or quasi-periodic lattice-like structure of the media. We can account for this by replacing m_e with m_{eff} where $m_{\mathrm{eff}} < m_e$ in the propagation equations and ionization model if desired, although we will leave m_e in the present equations. It should be noted that this ionization formulation will break down at high intensities when tunneling becomes dominant. In addition, its accuracy in condensed media will vary as it does take into account details of the true band structure.

2.4 Variable transform

Next we transform Eq. 3 from the lab frame (z,t) to the pulse frame $(z,\tau = t - z/v_{gL})$ where v_{gL} is the group velocity of the laser [2]. This gives the transformed derivatives

$$\begin{aligned} \frac{\partial}{\partial t} &\rightarrow \frac{\partial}{\partial \tau}, \\ \frac{\partial}{\partial z} &\rightarrow \frac{\partial}{\partial z} - \frac{1}{v_{gL}}\frac{\partial}{\partial \tau}. \end{aligned} \tag{18}$$

In addition, we make the slowly-varying envelope approximation and discard $\partial^2 A_n/\partial z^2$ assuming

$$\left|\frac{\partial^2 A_n}{\partial z^2}\right| \ll \left|k_n \frac{\partial A_n}{\partial z}\right| \tag{19}$$

and we similarly discard $\partial^2 A_n/\partial z \partial \tau$. Applying these changes to Eq. 3 yields the pulse-frame propagation equation

$$\left(\nabla_\perp^2 + 2ik_n\frac{\partial}{\partial z} + 2ik_n\left(\frac{1}{v_{gn}} - \frac{1}{v_{gL}}\right)\frac{\partial}{\partial \tau} - k_n\beta_n''\frac{\partial^2}{\partial \tau^2}\right)A_n = S_n^{NL} \tag{20}$$

and the electron density evolution in Eq. 10 becomes

$$\frac{\partial N_e}{\partial \tau} = \nu_{\mathrm{MPI}} N_0 + \nu_i N_e - \eta N_e. \tag{21}$$

2.5 Final propagation equations

The last step is to insert the nonlinear source function $S_n^{NL} = S_n^{\mathrm{Raman}} + S_n^{\mathrm{Kerr}} + S_n^{\mathrm{plasma}} + S_n^{\mathrm{ion}}$ into Eq. 20. With S_n^{NL} inserted and by assuming azimuthal symmetry for the transverse derivative, we arrive at the final propagation equations

$$
\begin{aligned}
&2ik_S\frac{\partial A_S}{\partial z}+\frac{\partial^2 A_S}{\partial r^2}+\frac{1}{r}\frac{\partial A_S}{\partial r}-k_S\beta_S''\frac{\partial^2 A_S}{\partial \tau^2}+2ik_S\left(\frac{1}{v_{gS}}-\frac{1}{v_{gL}}\right)\frac{\partial A_S}{\partial \tau}\\
&\qquad-\frac{N_e e^2}{m_e\varepsilon_0 c^2}\left(1-\frac{i\nu_e}{\omega_S}\right)A_S+\frac{i\omega_S}{2c^2}\frac{U_{\text{ion}}}{|A_S|^2}N_0\nu_{\text{MPI},S}A_S\\
&\qquad=\frac{-6\omega_S^2}{c^2}\left[\frac{1}{2}\chi_{NR}|A_S|^2A_S+(\chi_{RS}+\chi_{NR})|A_L|^2A_S+\chi_{NR}|A_A|^2A_S+(\chi_{RS}+\chi_{NR})A_L^2A_A^*e^{i\Delta kz}\right]\\
&2ik_L\frac{\partial A_L}{\partial z}+\frac{\partial^2 A_L}{\partial r^2}+\frac{1}{r}\frac{\partial A_L}{\partial r}-k_L\beta_L''\frac{\partial^2 A_L}{\partial \tau^2}\\
&\qquad-\frac{N_e e^2}{m_e\varepsilon_0 c^2}\left(1-\frac{i\nu_e}{\omega_L}\right)A_L+\frac{i\omega_L}{2c^2}\frac{U_{\text{ion}}}{|A_L|^2}N_0\nu_{\text{MPI},L}A_L\\
&\qquad=\frac{-6\omega_L^2}{c^2}\left[(\chi_{RA}+\chi_{NR})|A_S|^2A_L+\frac{1}{2}\chi_{NR}|A_L|^2A_L+(\chi_{RS}+\chi_{NR})|A_A|^2A_L+2\chi_{NR}A_L^*A_SA_Ae^{-i\Delta kz}\right]\\
&2ik_A\frac{\partial A_A}{\partial z}+\frac{\partial^2 A_A}{\partial r^2}+\frac{1}{r}\frac{\partial A_A}{\partial r}-k_A\beta_A''\frac{\partial^2 A_A}{\partial \tau^2}+2ik_A\left(\frac{1}{v_{gA}}-\frac{1}{v_{gL}}\right)\frac{\partial A_A}{\partial \tau}\\
&\qquad-\frac{N_e e^2}{m_e\varepsilon_0 c^2}\left(1-\frac{i\nu_e}{\omega_A}\right)A_A+\frac{i\omega_A}{2c^2}\frac{U_{\text{ion}}}{|A_A|^2}N_0\nu_{\text{MPI},A}A_A\\
&\qquad=\frac{-6\omega_A^2}{c^2}\left[\chi_{NR}|A_S|^2A_A+(\chi_{RA}+\chi_{NR})|A_L|^2A_A+\frac{1}{2}\chi_{NR}|A_A|^2A_A+(\chi_{RA}+\chi_{NR})A_L^2A_S^*e^{i\Delta kz}\right].
\end{aligned}
\tag{22}
$$

2.6 Model accuracy

Our model is motivated by underwater laser propagation experiments where only first-order SRS, Kerr focusing, four-wave mixing, and ionization were observed—hence these were the main nonlinear effects included. These experiments involve laser powers typically above the critical power for self-focusing (on the order of 1 MW in water), but low enough to assume relatively weak ionization ($N_e \ll N_0$). Previous work on underwater laser propagation found good agreement with experiment considering only some of these effects [8]; another work discusses the regimes where including Brillouin backscatter may be important [3]. This code may also prove useful for atmospheric propagation (see Ref. [2]) where the SRS, Kerr, and MPI effects are dominant. In addition, more basic problems with lower laser powers and linear or only weakly nonlinear effects may be modeled accurately.

3. SNOPROP: A SOLVER FOR NONLINEAR OPTICAL PROPAGATION

In addition to the analytical model discussed above, we present SNOPROP: A Solver for Nonlinear Optical Propagation. This Python-based simulation code solves the model in Eq. 22 on a cylindrical axisymmetric two-dimensional grid in radius r and time delay τ.

The code models each of the Stokes, laser, and anti-Stokes beams as 2D complex arrays, as well as the electron density as a 2D float array, in r and τ. The corresponding derivatives $\partial A_n/\partial z$ are then found for each beam using Eq. 22. The solution is advanced along z until reaching the user-specified distance.

3.1 Installation

The SNOPROP code is available via Zenodo [17]. After downloading the code, enter the root code directory and run `pip install .` to install SNOPROP. The `snoprop` package will now be available for importing.

Python 3 is required. Several python packages including Numpy [18], Scipy [19], and Numba [20] are required. Numba must be version 0.42 or greater. All of these packages are included by default with the Anaconda python distribution [21], which is the easiest way to use SNOPROP.

3.2 Numerical methods and accuracy

The code solves the propagation equations in Eq. 22 numerically using the Crank-Nicolson method, which offers 2nd-order accuracy in position z, and we similarly use 2nd-order derivatives in r and τ. This implicit method brings high stability and accuracy at the cost of requiring several iterations to solve at each step. The electron density in Eq. 21 is solved explicitly using a 4th-order Runge-Kutta scheme. The error is calculated as the mean absolute value of the change in each envelope field from one iteration to the next iteration of the solver.

The z step size may be specified manually or automatically chosen by a Courant-Friedrichs-Lewy (CFL) condition at the start. The code also features an adaptive z step option. At present, the $r = 0$ boundary forces $\partial A_n/\partial r|_{r=0} = 0$ as required by axisymmetry, but the other boundaries are free, they do not permit outgoing fields, and they can lead to unphysical reflections; care should be taken that the simulation box is large enough to accommodate the beam without interference by reflections.

Several unit tests are implemented to check against analytical results for linear propagation and diffraction, ionization rate, plasma refraction, Raman gain, Kerr focusing, and four-wave mixing. These tests are available via the command `snoprop.run_tests(v)` where `v` is an optional flag for more verbose output.

3.3 Input guide

SNOPROP simulations are typically run using a user-specified python input file that imports the `snoprop` package, defines the laser pulse and medium, and calls the `.run()` method. A template input script reads as follows.

```
import snoprop
params = { } # Define simulation parameters in here
sim = snoprop.Simulation(params)
sim.run()
```

Aside from any extra computations or setup the user wishes to do, the bulk of a simulation script usually consists of defining the parameter dictionary. The parameters fall into five main categories: material, model, grid, pulse, and output parameters. Each of these will be discussed in the next sections.

In each of the following sections, we will include a section of an example simulation script. When combined, these portions of code produce a complete simulation script which is reproduced in full in Appendix A. The motivation for these parameters and the simulation results will be discussed later in Chapter 4. The example input script in Appendix A begins with the following.

```
import numpy as np
import snoprop

# Before specifying the parameter dictionary, we can calculate
# needed quantities or define custom pulse profiles.
# In this example, we will use a custom Stokes pulse radial profile.
r0 = .3e-3 # Annulus major diameter in m
rwidth = .02e-3 # Annulus minor width in m
def radialProfile(r):
   return np.exp(-(r-r0)**2/(2*rwidth**2))

params = { # Now we will enter the parameter dictionary
```

3.3.1 Material specification

The parameter dictionary must include material parameters for the homogeneous background medium. All options are specified in Table 1.

The user may specify refractive indices in one of three methods. In the first method, the `material` parameter is set to `'custom'` and the user may specify the relevant linear refractive indices n_n, group indices n_{gn}, and GVD parameters β_n'' (see Eq. 4) for each beam in the input parameter dictionary.

In the second method, the user sets the `material` parameter to a python function which accepts a float (wavelength in m) and returns a float (index of refraction). The index of refraction, group indices, and GVD parameters for each beam are calculated from the refractive index data. With this method, the user can input custom refractive index data from tables or using the Sellmeier equation.

In the third method, the user may set `material` to `'vacuum'` which assumes a refractive index of unity.

The example input script in Appendix A continues with the following material parameters.

```
   # Material parameters
   'wavelength': 355e-9, # Pump laser wavelength in vacuum
   'wV': 2*3.14159*1.019e14, # Water vibrational freq. is 101.9 THz
   'material': 'custom', # We supply refractive indices manually
   'nS': 1.34927, # Stokes index of refraction
   'nL': 1.35721, # Laser index of refraction
   'nA': 1.36624, # Anti-Stokes index of refraction
   'nSg': 1.40398, # Stokes group index
   'nLg': 1.42694, # Laser group index
   'nAg': 1.45577, # Anti-Stokes group index
   'Uion': 9.0, # Ionization energy (eV)
   'N0': 3.33679e28, # Water molecular number density (1/m^3)
   'sigmaC': 3e-20, # collision cross section (m^2)
   'IMPI': 1.84e18, # Characteristic ionization intensity (W/m^2)
```

```
'eta': 1/1e-12, # Electron reattachment rate (s^-1)
'IBackground': 1e2, # Background intensity (W/m^2)
'n2Kerr': 5e-20, # Kerr index (n = n0 + n2*I) (m^2/W)
'n2Raman': -1.7e-20j, # Raman index (should be imaginary) (m^2/W)
```

3.3.2 *Model components*

Individual model components in Eq. 22, such as SRS, four-wave mixing, GVD, ionization, and others can be enabled or disabled in the simulation input and are detailed in Table 2.

By default, the code will only advance the Stokes and anti-Stokes fields if Raman scattering is enabled. Thus, a single-envelope simulation including only the laser field can be performed (with significantly shorter execution time) by disabling Raman scattering entirely.

Additional options in the solver can also be enabled here such as the adaptive z step. If the adaptive z step is enabled, then the solver automatically reduces the z step size by half when a large number of iterations is required to meet the error threshold. If the solver converges in a single iteration (and the current z step is less than the CFL condition requires), then the z step size is doubled. If the error remains large after many iterations and the code has already reduced the z step size by a factor of 10^4, the solver will exit with a description of the convergence failure.

The code also features an optional radial filter which damps high frequencies in either the electron density or the electric fields and may be useful to suppress numerical instabilities [22]. When filtering the electron density, the user should remember that it is calculated anew at every z-step and so it should be filtered at every z step. However, filtering the electric field at every step can quickly lead to significant energy loss and so it should be used sparingly. The filter may operate in real space (as a Gaussian filter with standard deviation of 1 cell in the radial direction), or in frequency space (using a Hankel transform and smoothly damping the upper half of frequencies). The Hankel filter is weighted as $w(f) = -0.5\cos[\pi(0.5\cos(\pi f/F)+0.5)]+0.5$ for frequency f with $F = 1/(2dr)$ being the maximum frequency. The Hankel filter may significantly reduce performance with a large number of grid points in r. More advanced filtering could be performed manually in the simulation script by interacting with the simulation object during execution (see Table 6).

The example input script in Appendix A continues with the following model parameters.

```
# Toggle model components
'include_plasma_refraction': True, # Toggle plasma refraction
'include_ionization': True, # Toggle ionization
'include_energy_loss': True, # Toggle energy loss to plasma
'include_raman': True, # Toggle stimulated Raman scattering
'include_fwm': True, # Toggle four-wave mixing
'include_kerr': True, # Toggle Kerr focusing
'include_group_delay': True, # Toggle group delay
'include_gvd': False, # Toggle group velocity dispersion
'adaptive_zstep': True, # Toggle adaptive zstep
'radial_filter': True, # Smooth the electron density at each step
```

3.3.3 Simulation grid

The simulation box size and resolution is specified for both the time τ and radius r axes. In addition, the user specifies z limits and may optionally specify a step size dz. All relevant options are detailed in Table 3.

Operation in 1D is not currently supported; the simulation box must have at least 4 cells in each direction to run.

The example input script in Appendix A continues with the following grid parameters.

```
# Grid parameters
'zrange': [0, .011], # Stop the simulation at 1cm
'trange': [-25e-12,25e-12], # simulate a 40 ps box
't_clip': 5e-12, # Cut off temporal profile 5ps from box edge
'tlen': 200, # Number of cells in time
'rrange': [0., 1e-3], # Radial boundary at 1 mm
'rlen': 6000, # Number of cells in radius
```

3.3.4 Pulse profile

Each of the three beams (laser, Stokes, and anti-Stokes) may be initialized with a user-specified profile and unique background intensities. To input a laser profile, set the `profile_L` parameter in the main simulation dictionary to another dictionary containing the necessary pulse parameters as shown in Table 4. The laser profile (`profile_L`) is required, while the Stokes (`profile_S`) and anti-Stokes (`profile_A`) are optional and will default to the uniform background intensity specified globally in the `IBackground` parameter (see Table 1). The pulse profiles may be input in three ways:

1. Users may specify an arbitrary number of radially- and temporally-Gaussian pulses in a pulse train. In this case, each pulse is given a duration, temporal offset, spot size, and fraction of the total energy, and the whole pulse is given an energy, focal length, and optionally a background intensity.

2. Users may input the radial or temporal profile or both as an array of fluence vs r or power vs τ. In this case, the waveform resulting from a radial profile $F(r)$ and temporal profile $P(\tau)$ is simply $I(\tau, r) \propto F(r)P(\tau)$ scaled to reach the appropriate total beam energy. Even if both radial and temporal profiles are suppled, the user must still specify a pulse in the pulse train and associated energy and focal length.

3. Users may explicitly input the full 2D complex envelope fields A_S, A_L, or A_A ordered with the first index being τ and the second r. In this case, no other options must be specified. If an optional energy is specified, the entire field will be scaled accordingly.

With each of these methods, if a background intensity is specified in the pulse parameters, this background will be added to the field after the pulse has already been scaled to the appropriate energy.

The example input script in Appendix A continues with the following pulse parameters.

```
# Pulse profiles
'profile_L': {
   'pulse_length_fwhm': [10e-12], # Temporal lengths of each pulse
   'toffset': [0], # Offsets for the multi-pulses
   'efrac': [1], # Energy fraction in each pulse
   'pulse_radius_e2': [.1e-3], # Intensity to 1/e^2 radius
   'focal_length': 0.006,
   'energy': 6e-06, # Pulse energy in J
},
'profile_S': {
   'pulse_length_fwhm': [10e-12], # Temporal lengths of each pulse
   'toffset': [0], # Offsets for the multi-pulses
   'efrac': [1], # Energy fraction in each pulse
   'focal_length': 0.0055,
   'radial_func': radialProfile, # Custom radial profile function
   'energy': 2e-06, # Pulse energy in J
},
```

3.3.5 Output parameters

When a simulation is initialized, the code attempts to create folders for the output files in the current working directory if file output is enabled. These folders are named `data/`, where all diagnostic data is saved, and `logs/`, where input and calculated parameters are saved at the start and end of each simulation. If these folders already exist, the code will inform the user and exit.

Various scalar, 1D, and full 2D diagnostics may be saved to disk at intervals specified by the user. All intervals are specified as number of integration steps. If the user wishes constant spacing between diagnostic saves, then they should not use the adaptive z step feature.

Scalar diagnostics are saved in a single file called `scalars.csv`; new scalar data is appended to the end of this file as the simulation progresses. 1D data are saved as a new Python pickle (`.pckl`) file for each z-step which matches the specified interval with the prefix `Data1D_` followed by the step number. 2D data are also saved as `.pckl` files with the prefix `Data2D_` followed by the step number. The user may choose specific scalar, 1D and 2D diagnostics for specific beams; the options detailed in Table 5.

Restart files may also be saved at user-specified intervals. The code simply saves the simulation object in a `.pckl` file in the `data/` directory with the prefix `Restart_` followed by the z-step. To use a restart file, the user should import the `pickle` and `snoprop` packages and then load the simulation object with (for example) `sim = pickle.load(open('data/Restart_001000.pckl','rb'))`. Then the simulation may be run with `sim.run()` as before.

The example input script in Appendix A continues with the following output parameters.

```
# Data output
'save_restart_interval': 0, # Restarts disabled
'save_scalars_interval': 20, # Interval at which to save scalars
```

```
    'save_scalars_which': [ # Select which scalars to save to file
        # Individual and total energies
        'Energy_S','Energy_L','Energy_A','Energy_T',
        # Individual and total beam spot sizes (FWHM)
        'FWHM_S','FWHM_L','FWHM_A','FWHM_T',
        # Individual and total beam spot sizes (RMS integrated)
        'RMSSize_S','RMSSize_L','RMSSize_A','RMSSize_T'
    ],
    'save_1D_interval': 100, # Interval at which to save 1D data
    'save_1D_which': [ # Select which 1D data to save to file
        # Power, fluence, and final/peak electron densities
        'PS','PL','PA','FS','FL','FA','Ne_end','Ne_max',
    ],
    'save_2D_interval': 9900, # Interval at which to save 2D data
    'save_2D_which': [ # Select which 2d data to save to file
        'IS','IL','IA','Ne', # Intensities and electron density
    ],
```

3.3.6 Interacting with the simulation

In addition to running the entire simulation via the `.run()` method, the user may also run it step-by-step with the `.move()` method. Many more public methods are implemented to extract simulation parameters and fields as well as manually edit fields using the simulation object. These methods are detailed in Table 6. Note that all 2D data types are Numpy arrays.

The example input script in Appendix A ends with the following code to execute the simulation.

```
}

sim = snoprop.Simulation(params)
sim.run()
```

3.4 Performance

The solver is implemented in python using the just-in-time compiler Numba. The Numba library allows simultaneous multithreading of the core math, and the number of threads can be specified with the `NUMBA_NUM_THREADS` environment variable when running the script (e.g. as `NUMBA_NUM_THREADS=4 python simulation_script.py`). In addition, Numba uses the LLVM compiler library to accelerate with other methods including loop unrolling and SIMD vectorization where possible, tailored at runtime to the specific system on which the code is run. Because Numba is a just-in-time compiler, it can optimize the solver including only the effects turned on in each individual simulation. Numba is very easy to use as it requires no additional software beyond a standard Python package, allows fully portable code, and infers types automatically. It is included with the Anaconda python distribution.

Numba is powerful but has some limitations. The Numba compiler requires a few seconds to run when the code is first launched. Ahead-of-time Numba compilation could mitigate this at the cost of lower portability and possibly lower performance. The ease of use comes at the cost of reduced control over memory

management and possibly lower performance than could be achieved with a lower-level carefully optimized implementation.

Future improvements could allow simulation of a wider variety of problems or better performance. Implementing Stimulated Brillouin Scattering or second-order Stokes effects may be important for some problems. Absorbing boundaries would allow smaller simulation boxes. Other performance improvements could include using PDE stenciling and writing the core code in a language with better control over multithreading and SIMD as well as implementing message passing for distributed computing.

4. DEMONSTRATION: PICOSECOND PULSE PROPAGATION IN WATER

Simulations can help interpret experimental results, optimize future experiments, and explore new concepts before experimental testing begins. SNOPROP was written with these goals in mind applied to underwater picosecond laser propagation. We will demonstrate the code with a simulation of underwater laser propagation which highlights the four-wave mixing modeling capability and numerous diagnostics.

Our model system includes a narrowband 355 nm, 10 ps tightly-focused collimated pump laser propagating in water. A power of 0.6 MW—roughly twice the critical power for self-focusing given our choice of nonlinear Kerr index—ensures that many nonlinear effects such as Kerr focusing, SRS, and ionization will be simultaneously important. We also include an annulus of Stokes light with a lower 0.2 MW power which is initially located radially outside of the pump and is propagating inwards at the optimal angle given by Eq. 7. While the pump will produce Stokes light naturally from the background through SRS, starting with an annular Stokes seed pulse allows us to determine the radial profile and ensures that the phase matching condition Eq. 6 is best satisfied to allow for maximum anti-Stokes production. The bandwidth of a 10 ps pulse is low enough that we ignore GVD altogether over our short propagation distance of only 1.1 cm. All other effects in Eq. 22 will be included.

The pump pulse is initialized as a gaussian pulse in time and radius by specifying the width and FWHM duration as input parameters to the code. However, the radial profile of the annular Stokes wave is input as a python function, demonstrating SNOPROP's flexibility to accept unusual pulse shapes. We record the material and pulse parameter input for the simulation in Table 7. The box size was chosen to be large enough that reflections do not reach the outermost r boundary and the group delay does not reach the τ boundaries. We utilize the scalar, 1D lineout, and full 2D diagnostics and run the simulation. The exact input deck is given in Appendix A.

4.1 Simulation results

The simulation is advanced to $z = 1.1$ cm using the `.run()` command. During execution, various diagnostics are saved as described in Section 3.3.5. The evolution of the energy is shown in Fig. 1. As the annular Stokes beam comes to focus, it gains energy from the focusing pump pulse due to SRS and also produces a significant amount of anti-Stokes light through four-wave mixing. Several percent of the total energy are converted to anti-Stokes light. The drop in total energy reflects the losses to medium vibrational excitation by SRS as well as ionization losses.

Several 1D profiles, including the starting and final radial and temporal profiles, are shown in Fig. 2. The final radial profile shown in 2(c), taken at exactly twice the Stokes focal distance $z = 2f_S$, shows that the anti-Stokes light produced is localized in a hollow cone with a divergence angle of approximately 2.4 degrees as

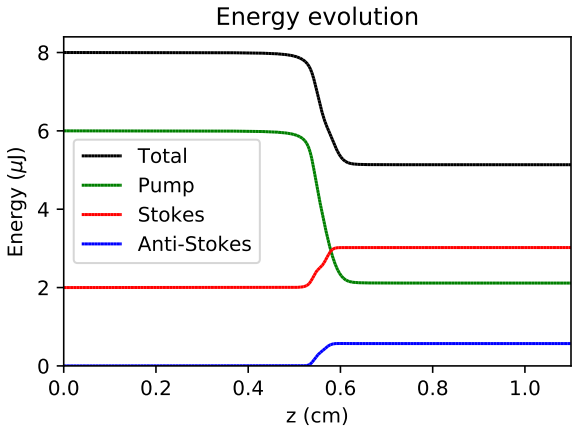

Fig. 1—Simulated energies of the pump beam, Stokes beam, anti-Stokes beam, and total energy during underwater propagation.

predicted by Eq. 8. We also see in Fig. 2(d) that the Stokes and anti-Stokes beams are slightly temporally shifted due to the group delay with the Stokes wave travelling faster and the anti-Stokes wave travelling slower than the pump. The temporal asymetry in the pump profile could be a result of the asymmetric plasma density which will more strongly defocus the rear of the pulse or the group delay causing the Stokes beam to absorb the head of the pump more strongly.

The 2D pump profile near focus is shown in Fig. 3. We see that the center of the pulse is beginning to be hollowed out by the Stokes wave, and the higher tail power in Fig. 2(d) correlates with a lower tail intensity on axis at focus. In some cases, the 2D diagnostics may help interpret ambiguities in 1D diagnostics such as the presence of possible ionization-induced scattering instabilities [23].

5. CONCLUSION

Intense nonlinear optical propagation in dielectric media can be explored by numerical modeling. Propagation equations are derived for several important nonlinear effects such as group velocity dispersion, stimulated Raman scattering, self-phase and cross-phase modulation, four-wave mixing, and multiphoton and collisional ionization. These equations are implemented in the 2D axially symmetric envelope simulation code SNOPROP: A Solver for Nonlinear Optical Propagation. After detailing the implementation, giving a thorough user guide, and discussing performance and possible future work, we demonstrate the code's performance with results of an underwater laser propagation simulation. We expect that SNOPROP will continue to be useful for many future problems involving nonlinear laser propagation.

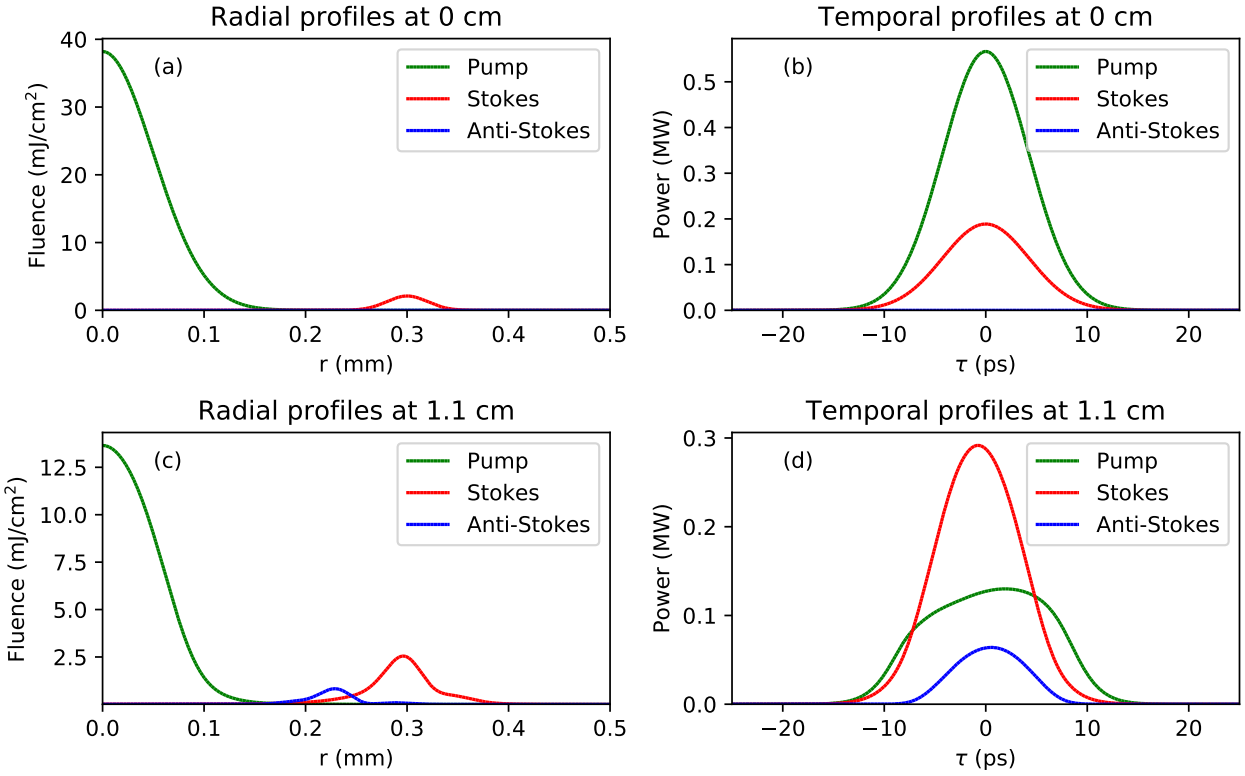

Fig. 2—Simulated 1D radial (left) and temporal (right) beam profiles before (top) and after (bottom) 1.1 cm propagation in water.

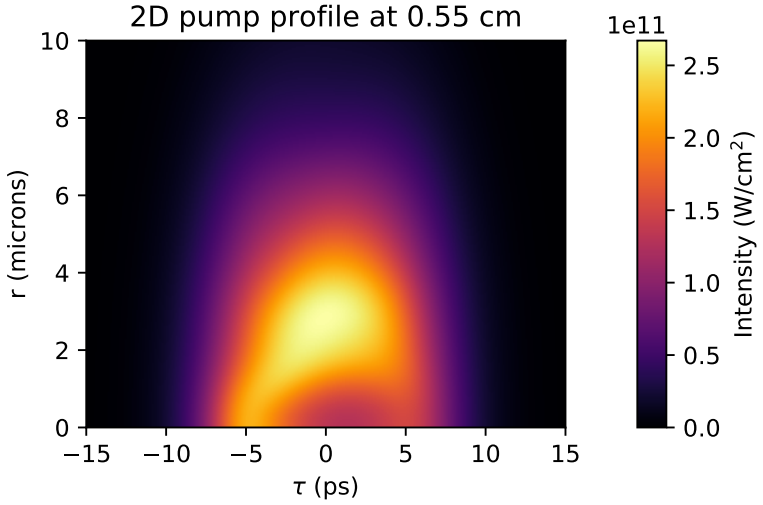

Fig. 3—Simulated 2D profile of the pump beam after 0.56 cm propagation in water.

Table 1—Material input parameters

Key	Req.	Type	Description
`wavelength`	yes	`float`	Laser wavelength (m)
`material`	yes	`string or function`	Material name. May be set to `'vacuum'` to ignore all material effects, `'custom'` in which case several optional parameters for refractive indices, group indices, and group velocity dispersion parameters are required (see below), or a python function accepting as input a float (wavelength in m) and returning a float (refractive index).
`wV`	yes	`float`	Material vibration frequency (1/s)
`Uion`	yes	`float`	Material ionization energy (eV)
`sigmaC`	yes	`float`	Collision cross section (m^2)
`IMPI`	yes	`float`	Multiphoton ionization characteristic intensity (W/m^2)
`eta`	yes	`float`	Electron loss rate (1/s)
`IBackground`	yes	`float`	Initial background intensity for all fields (W/m^2). This can be overridden if `IBackground` is specified as an element of the pulse profiles `profile_S`, `profile_L`, `profile_A` (see Table 4).
`n2Kerr`	yes	`float`	Kerr index n_2 as in $n = n_0 + n_2 I$ (m^2/W)
`n2Raman`	yes	`complex`	Raman index; should be imaginary (m^2/W)
`effective_mass`	no	`float`	Effective electron mass (fraction of electron mass) (default 1.0)
`nS`	no	`float`	Stokes refractive index (req. if `material = 'custom'`)
`nL`	no	`float`	Laser refractive index (req. if `material = 'custom'`)
`nA`	no	`float`	Anti-Stokes refractive index (req. if `material = 'custom'`)
`nSg`	no	`float`	Stokes group index (req. if `material = 'custom'`)
`nLg`	no	`float`	Laser group index (req. if `material = 'custom'`)
`nAg`	no	`float`	Anti-Stokes group index (req. if `material = 'custom'`)
`gvd_bS`	no	`float`	Stokes GVD parameter (s^2/m) (req. if `material = 'custom'`)
`gvd_bL`	no	`float`	Laser GVD parameter (s^2/m) (req. if `material = 'custom'`)
`gvd_bA`	no	`float`	Anti-Stokes GVD parameter (s^2/m) (req. if `material = 'custom'`)

Table 2—Model component parameters

Key	Req.	Type	Description
`include_plasma_refraction`	yes	`bool`	Toggle plasma refraction
`include_ionization`	yes	`bool`	Toggle ionization
`include_energy_loss`	yes	`bool`	Toggle laser energy loss to ionization and heating
`include_raman`	yes	`bool`	Toggle stimulated Raman scattering
`include_fwm`	yes	`bool`	Toggle four-wave mixing
`include_kerr`	yes	`bool`	Toggle Kerr focusing
`include_group_delay`	yes	`bool`	Toggle group delay
`include_gvd`	yes	`bool`	Toggle group velocity dispersion
`include_stokes`	no	`bool`	Toggle whether to update the Stokes field. Default is the value of `include_raman`.
`include_antistokes`	no	`bool`	Toggle whether to update the anti-Stokes field. Default is the value of `include_raman`.
`adaptive_zstep`	no	`bool`	Toggle adaptive z step. Default is true.
`radial_filter`	no	`bool`	Toggle the radial filter. Default is false.
`radial_filter_interval`	no	`int`	Specify the interval (number of simulation steps) at which to apply the radial filter. Default is 1.
`radial_filter_type`	no	`string`	Specify the filter type: either "gaussian" (real-space filter) or "bessel" (frequency-space filter). Default is "gaussian".
`radial_filter_field`	no	`string`	Specify which field to apply the radial filter to. Options are "electron density" and "electric field"; currently, only one option is possible at a time. Default is "electron density."

Table 3—Simulation grid parameters

Key	Req.	Type	Description
`zrange`	yes	`[float]`	Simulation z limits (m). Should be a 2-tuple with the first element 0.
`dz`	no	`float`	Manually specify the maximum simulation z step size (m). If `adaptive_zstep` is disabled, then the z step size will be fixed to this value for the duration of the simulation.
`trange`	yes	`[float]`	Simulation τ limits (s). Should be a 2-tuple.
`t_clip`	no	`float`	Temporal boundary width in which to clip the pulses (s). Useful to clip gaussian waveforms so they do not extend to the boundary.
`tlen`	yes	`int`	Number of steps in τ.
`rrange`	yes	`[float]`	Simulation r limits (m). Should be a 2-tuple with the first element 0.
`rlen`	yes	`int`	Number of steps in r.

Table 4—Pulse profile parameters

Key	Req.	Type	Description
`pulse_length_fwhm`	yes	`[float]`	Intensity FWHM temporal lengths of each pulse in pulse train (s).
`toffset`	yes	`[float]`	Temporal offsets for each pulse in the pulse train (s).
`efrac`	yes	`[float]`	Fraction of total energy in each pulse in pulse train.
`pulse_radius_half`	yes	`[float]`	Spot radius to half max intensity for each pulse in pulse train (m).
`pulse_radius_e2`	no	`[float]`	Spot radius to $1/e^2$ intensity for each pulse in pulse train (m) (may specify this parameter alternatively to `pulse_intensity_radius_half`)
`focal_length`	yes	`float`	Focus distance (m).
`energy`	yes	`float`	Total energy of the entire pulse train (J).
`radial_data`	no	`[[float], [float]]`	Radial data for the pulse profile. The first array is radius in m and the second is fluence (which will be normalized by the code to reach the specified pulse energy).
`temporal_data`	no	`[[float], [float]]`	Temporal data for the pulse profile. The first array is time in s and the second is power (which will be normalized by the code to reach the specified pulse energy).
`2D_data`	no	`2D complex`	Spatiotemporal electric field envelope profile. The user may specify the entire electric field envelope as a 2D complex array where the first axis is time τ and the second axis is radius r. The resulting profile will be normalzed to the correct power and a background field added (if a background field has been specified either globablly or for this particular profile). The field will be normalized by the code to reach the specified pulse energy.

Table 5—Output parameters

Key	Req.	Type	Description
`file_output`	no	`bool`	Toggle all file output including all diagnostics described in this section. Default is true (permit file output).
`console_logging_interval`	no	`int`	Specify interval at which to log simulation progress in the console; set to 0 to disable console logging. Default is 20.
`save_restart_interval`	no	`int`	Interval in simulation steps at which to save restart files. Default is 0 (no saves).
`save_scalars_interval`	no	`int`	Interval in simulation steps at which to save scalar data. Default is 0 (no saves).
`save_scalars_which`	no	`[string]`	Select scalars to save to file. Currently supported options are the Stokes, laser, anti-Stokes, and total energies (`Energy_S`, `Energy_L`, `Energy_A`, `Energy_T`), FWHM spot size for the Stokes, laser, anti-Stokes, and total beams (`FWHM_S`, `FWHM_L`, `FWHM_A`, `FWHM_T`), and the RMS spot sizes for the Stokes, laser, anti-Stokes, and total beams (`RMSSize_S`, `RMSSize_L`, `RMSSize_A`, `RMSSize_T`). Simulation step and z position are saved as well.
`save_1D_interval`	no	`int`	Interval in simulation steps at which to save 1D profiles. Default is 0 (no saves).
`save_1D_which`	no	`[string]`	Select which 1D data to save to file. Currently supported options are the Stokes, laser, anti-Stokes, and total fluence vs r (`FS`, `FL`, `FA`, `FT`), the Stokes, laser, anti-Stokes, and total beam powers vs τ (`PS`, `PL`, `PA`, `PT`), and the peak (`Ne_max`) or final (`Ne_end`) electron density vs r. The z position and axes information are also saved.
`save_2D_interval`	no	`int`	Interval in simulation steps at which to save 2D data. Default is 0 (no saves).
`save_2D_which`	no	`[string]`	Select which 2D data to save to file. Currently supported options are the Stokes, laser, anti-Stokes, and total intensity (`IS`, `IL`, `IA`, `IT`), Stokes, laser, and anti-Stokes electric fields (`ES`, `EL`, `EA`), Stokes, laser, and anti-Stokes envelope fields (`AS`, `AL`, `AA`), and electron density (`Ne`). The z position as well as axes limits are also saved.

Table 6—Public methods

Method	Input	Returns	Description
`.run()`			Run the simulation until z reaches the end specified in the parameter dictionary.
`.move()`			Step the simulation forward by one z-step.
`.getZ()`		`float`	Run the simulation until z reaches the end specified in the parameter dictionary.
`.getBeamParams()`		`dict`	Returns a dictionary with the vacuum wavelenths, refractive indices, group indices, and GVD beta parameters for each beam.
`.getGrid()`		`dict`	Returns a dictionary with the grid parameters in r, τ, and z as well as the 1D and 2D grids themselves.
`.getEnS()`		`float`	Get the Stokes energy (J).
`.getEnL()`		`float`	Get the laser energy (J).
`.getEnA()`		`float`	Get the anti-Stokes energy (J).
`.getIS()`		`2D float`	Get the Stokes intensity (W/m^2).
`.getIL()`		`2D float`	Get the laser intensity (W/m^2).
`.getIA()`		`2D float`	Get the anti-Stokes intensity (W/m^2).
`.getAS()`		`2D complex`	Get the Stokes envelope field (V/m^2).
`.getAL()`		`2D complex`	Get the laser envelope field (V/m^2).
`.getAA()`		`2D complex`	Get the anti-Stokes envelope field (V/m^2).
`.setAS(A)`	`2D complex`		Set the Stokes envelope to a field `A` (V/m^2).
`.setAL(A)`	`2D complex`		Set the laser envelope to a field `A` (V/m^2).
`.setAA(A)`	`2D complex`		Set the anti-Stokes envelope to a field `A` (V/m^2).
`.getEnField(A,n)`	`2D complex, float`	`float`	Get the energy (J) of an envelope field `A` (V/m) with index of refraction `n`.

Table 7—Demonstration parameters

Variable	Symbol	Value
Vibrational frequency	ω_v	6.4×10^{14} s^{-1}
Ionization energy	U_{ion}	9.0 eV
Collision cross section	σ_c	3×10^{-16} cm^2
Multiphoton ionization threshold	I_{MPI}	1.84×10^{14} W/cm^2
Electron attachment rate	η	10^{11} s^{-1}
Kerr susceptibility	χ_{NR}	3.3×10^{-18} cm^2/V^2
Raman susceptibility	χ_{RS}	$-1.1i \times 10^{-18}$ cm^2/V^2
Density of neutral water molecules	N_0	3.34×10^{22} cm^{-3}
Effective electron mass	m_{eff}	$1.0\ m_e$
Background intensity	I_{BG}	10^{-2} W/cm^2
Stokes wavelength	λ_S	404 nm
Laser wavelength	λ_L	355 nm
Anti-Stokes Wavelength	λ_A	317 nm
Stokes refractive index	n_S	1.3493
Laser refractive index	n_L	1.3572
Anti-Stokes refractive index	n_A	1.3663
Stokes group index	n_{gS}	1.40398
Laser group index	n_{gL}	1.42694
Anti-Stokes group index	n_{gA}	1.45577
Laser energy	U_L	6 µJ
Laser FWHM duration	τ_L	10 ps
Laser 1/e^2 spot size	D	0.1 mm
Laser focal length	f_L	0.6 cm
Stokes energy	U_S	2 µJ
Stokes FWHM duration	τ_S	10 ps
Stokes annulus major diameter	D_S	0.6 mm
Stokes annulus 1/e^2 minor width	D_{S2}	0.04 mm
Stokes focal length	f_S	0.55 cm
Stokes focusing angle	θ_S	3.13 degrees

REFERENCES

1. M. Kolesik, J. V. Moloney, and M. Mlejnek, "Unidirectional Optical Pulse Propagation Equation," *Phys. Rev. Lett.* **89**(28), 283902 (2002), doi:10.1103/PhysRevLett.89.283902. URL https://doi.org/10.1103/PhysRevLett.89.283902.

2. P. Sprangle, J. R. Peñano, and B. Hafizi, "Propagation of intense short laser pulses in the atmosphere," *Phys. Rev. E* **66**, 046418 (Oct 2002), doi:10.1103/PhysRevE.66.046418. URL https://link.aps.org/doi/10.1103/PhysRevE.66.046418.

3. B. Hafizi, J. P. Palastro, J. R. Peñano, T. G. Jones, L. A. Johnson, M. H. Helle, D. Kaganovich, Y. H. Chen, and A. B. Stamm, "Stimulated Raman and Brillouin scattering, nonlinear focusing, thermal blooming, and optical breakdown of a laser beam propagating in water," *J. Opt. Soc. Am. B* **33**(10), 2062–2072 (Oct 2016), doi:10.1364/JOSAB.33.002062. URL http://josab.osa.org/abstract.cfm?URI=josab-33-10-2062.

4. P. Mora and J. Thomas M. Antonsen, "Kinetic modeling of intense, short laser pulses propagating in tenuous plasmas," *Phys. Plasmas* **4**(1), 217–229 (1996), doi:10.1063/1.872134. URL https://doi.org/10.1063/1.872134.

5. B. Hafizi, J. P. Palastro, J. R. Peñano, D. F. Gordon, T. G. Jones, M. H. Helle, and D. Kaganovich, "Stimulated Raman scattering and nonlinear focusing of high-power laser beams propagating in water," *Opt. Lett.* **40**(7), 1556–1558 (Apr 2015), doi:10.1364/OL.40.001556. URL http://ol.osa.org/abstract.cfm?URI=ol-40-7-1556.

6. L. Bergé, S. Skupin, F. Lederer, G. Méjean, J. Yu, J. Kasparian, E. Salmon, J. P. Wolf, M. Rodriguez, L. Wöste, R. Bourayou, and R. Sauerbrey, "Multiple Filamentation of Terawatt Laser Pulses in Air," *Phys. Rev. Lett.* **92**(22), 225002 (2004), doi:10.1103/PhysRevLett.92.225002. URL http://dx.doi.org/10.1103/PhysRevLett.92.225002.

7. S. Champeaux, L. Bergé, D. Gordon, A. Ting, J. Peano, and P. Sprangle, "(3+1)-dimensional numerical simulations of femtosecond laser filaments in air: Toward a quantitative agreement with experiments," *Phys. Rev. E* **77**(3), 036406 (2008), doi:10.1103/PhysRevE.77.036406. URL http://dx.doi.org/10.1103/PhysRevE.77.036406.

8. M. H. Helle, T. G. Jones, J. R. Peñano, D. Kaganovich, and A. Ting, "Formation and propagation of meter-scale laser filaments in water," *Applied Physics Letters* **103**(12), 121101 (2013), doi:10.1063/1.4821447. URL https://doi.org/10.1063/1.4821447.

9. Y. H. Chen, A. Stamm, J. Palastro, B. Hafizi, T. G. Jones, and D. Kaganovich, "Nonlinear Propagation of 100 ps, UV Laser Pulses in Water with Strong Stimulated Raman Stokes Coupling," Proceedings of the Conference on Lasers and Electro-Optics (Optical Society of America), 2017, p. JW2A.45. URL http://www.osapublishing.org/abstract.cfm?URI=CLEO_QELS-2017-JW2A.45.

10. H. Yui and T. Sawada, "Interaction of Excess Electrons with Water Molecules at the Early Stage of Laser-Induced Plasma Generation in Water," *Phys. Rev. Lett.* **85**(16), 3512–3515 (2000), doi:10.1103/PhysRevLett.85.3512. URL https://doi.org/10.1103/PhysRevLett.85.3512.

11. R. W. Boyd, *Nonlinear Optics*, third ed. (Academic Press, Burlington, 2008), ISBN 978-0-12-369470-6, doi:https://doi.org/10.1016/B978-0-12-369470-6.00007-1. URL http://www.sciencedirect.com/science/article/pii/B9780123694706000071.

12. N. Bloembergen, *Nonlinear Optics*, 4th ed. (Harvard University, 1996), ISBN 981-02-2598-9.

13. C. G. Morgan, "Laser-induced breakdown of gases," *Rep. Prog. Phys.* **38**(5), 621–655 (1975), doi:10.1088/0034-4885/38/5/002. URL https://doi.org/10.1088/0034-4885/38/5/002.

14. A. D. MacDonald, *Microwave Breakdown in Gases*, first ed. (John Wiley & Sons, Inc., 1966).

15. Y. P. Raizer, *Laser-Induced Discharge Phenomena* (Consultants Bureau, 1977).

16. Y. P. Raizer, "Optical Discharges," *J. Phys. Colloques* **40**(C7), C7–141–C7–147 (1979), doi:10.1051/jphyscol:19797436. URL https://doi.org/10.1051/jphyscol:19797436.

17. J. R. Peterson, "SNOPROP: A Solver for Nonlinear Optical Propagation," https://doi.org/10.5281/zenodo.3908609, 2020.

18. T. E. Oliphant, *A guide to NumPy*, volume 1 (Trelgol Publishing USA, 2006).

19. P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and S... Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods* **17**, 261–272 (2020), doi:10.1038/s41592-019-0686-2. URL https://doi.org/10.1038/s41592-019-0686-2.

20. S. Lam, A. Pitrou, and S. Seibert, "Numba: a LLVM-based Python JIT compiler," Proceedings of the LLVM '15: Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC, New York, NY, USA (Association for Computing Machinery), November 2015, pp. 1–6. doi:10.1145/2833157.2833162. URL https://doi.org/10.1145/2833157.2833162.

21. "Anaconda Software Distribution." URL https://anaconda.com.

22. D. Potter, *Computational Physics* (Wiley, 1973).

23. J. Thomas M. Antonsen and Z. Bian, "Ionization Induced Scattering of Short Intense Laser Pulses," *Phys. Rev. Lett.* **82**(18), 3617–3620 (1999), doi:10.1103/PhysRevLett.82.3617. URL https://doi.org/10.1103/PhysRevLett.82.3617.

Appendix A

SAMPLE SIMULATION FILE

We include here a sample SNOPROP input file which simulates intense underwater laser propagation.

```
import numpy as np
import snoprop

# Before specifying the parameter dictionary, we can calculate
# needed quantities or define custom pulse profiles.
# In this example, we will use a custom Stokes pulse radial profile.
r0 = .3e-3 # Annulus major diameter in m
rwidth = .02e-3 # Annulus minor width in m
def radialProfile(r):
   return np.exp(-(r-r0)**2/(2*rwidth**2))

params = { # Now we will enter the parameter dictionary
   # Material parameters
   'wavelength': 355e-9, # Pump laser wavelength in vacuum
   'wV': 2*3.14159*1.019e14, # Water vibrational freq. is 101.9 THz
   'material': 'custom', # We supply refractive indices manually
   'nS': 1.34927, # Stokes index of refraction
   'nL': 1.35721, # Laser index of refraction
   'nA': 1.36624, # Anti-Stokes index of refraction
   'nSg': 1.40398, # Stokes group index
   'nLg': 1.42694, # Laser group index
   'nAg': 1.45577, # Anti-Stokes group index
   'Uion': 9.0, # Ionization energy (eV)
   'N0': 3.33679e28, # Water molecular number density (1/m^3)
   'sigmaC': 3e-20, # collision cross section (m^2)
   'IMPI': 1.84e18, # Characteristic ionization intensity (W/m^2)
   'eta': 1/1e-12, # Electron reattachment rate (s^-1)
   'IBackground': 1e2, # Background intensity (W/m^2)
   'n2Kerr': 5e-20, # Kerr index (n = n0 + n2*I) (m^2/W)
   'n2Raman': -1.7e-20j, # Raman index (should be imaginary) (m^2/W)


   # Toggle model components
   'include_plasma_refraction': True, # Toggle plasma refraction
   'include_ionization': True, # Toggle ionization
   'include_energy_loss': True, # Toggle energy loss to plasma
   'include_raman': True, # Toggle stimulated Raman scattering
```

```
'include_fwm': True, # Toggle four-wave mixing
'include_kerr': True, # Toggle Kerr focusing
'include_group_delay': True, # Toggle group delay
'include_gvd': False, # Toggle group velocity dispersion
'adaptive_zstep': True, # Toggle adaptive zstep
'radial_filter': True, # Smooth the electron density at each step

# Grid parameters
'zrange': [0, .011], # Stop the simulation at 1cm
'trange': [-25e-12,25e-12], # simulate a 40 ps box
't_clip': 5e-12, # Cut off temporal profile 5ps from box edge
'tlen': 200, # Number of cells in time
'rrange': [0., 1e-3], # Radial boundary at 1 mm
'rlen': 6000, # Number of cells in radius

# Pulse profiles
'profile_L': {
   'pulse_length_fwhm': [10e-12], # Temporal lengths of each pulse
   'toffset': [0], # Offsets for the multi-pulses
   'efrac': [1], # Energy fraction in each pulse
   'pulse_radius_e2': [.1e-3], # Intensity to 1/e^2 radius
   'focal_length': 0.006,
   'energy': 6e-06, # Pulse energy in J
},
'profile_S': {
   'pulse_length_fwhm': [10e-12], # Temporal lengths of each pulse
   'toffset': [0], # Offsets for the multi-pulses
   'efrac': [1], # Energy fraction in each pulse
   'focal_length': 0.0055,
   'radial_func': radialProfile, # Custom radial profile function
   'energy': 2e-06, # Pulse energy in J
},

# Data output
'save_restart_interval': 0, # Restarts disabled
'save_scalars_interval': 20, # Interval at which to save scalars
'save_scalars_which': [ # Select which scalars to save to file
   # Individual and total energies
   'Energy_S','Energy_L','Energy_A','Energy_T',
   # Individual and total beam spot sizes (FWHM)
   'FWHM_S','FWHM_L','FWHM_A','FWHM_T',
   # Individual and total beam spot sizes (RMS integrated)
   'RMSSize_S','RMSSize_L','RMSSize_A','RMSSize_T'
],
'save_1D_interval': 100, # Interval at which to save 1D data
'save_1D_which': [ # Select which 1D data to save to file
```

```
        # Power, fluence, and final/peak electron densities
        'PS','PL','PA','FS','FL','FA','Ne_end','Ne_max',
    ],
    'save_2D_interval': 9900, # Interval at which to save 2D data
    'save_2D_which': [ # Select which 2d data to save to file
        'IS','IL','IA','Ne', # Intensities and electron density
    ],
}

sim = snoprop.Simulation(params)
sim.run()
```