Received 3 May 2012
Revised 18 January 2013
Accepted 24 January 2013

Method of lines solutions for the three-wave model of Brillouin equations

Fikri Serdar Gokhan
Department of Electrical and Electronics Engineering, Hasan Kalyoncu University, Sahinbey, Turkey
Graham W. Griffiths
School of Engineering and Mathematical Sciences, City University, London, UK, and
William E. Schiesser
Department of Engineering and Mathematics, Lehigh University, Bethlehem, Pennsylvania, USA

Abstract

Purpose – The purpose of this paper is to present the method of lines (MOL) solution of the stimulated Brillouin scattering (SBS) equations (a system of three first-order hyperbolic partial differential equations (PDEs)), describing the three-wave interaction resulting from a coupling between light and acoustic waves. The system has complex numbers and boundary values.

Design/methodology/approach – System of three first-order hyperbolic PDEs are first transformed and then spatially discretized. Superbee flux limiter is proposed to offset numerical damping and dispersion, brought on by the low order approximation of spatial derivatives in the PDEs. In order to increase computational efficiency, the structured structure of the PDE Jacobian matrix is identified and a sparse integration algorithm option of the ordinary differential equation (ODE) solvers is used. The flux limiter based on higher order approximations eliminates numerical oscillation. Examples are presented, and the performance of the Matlab ODE solvers is evaluated by comparison.

Findings – This type of solution provides a rapid means of investigating SBS as a tool in fiber optic sensing.

Originality/value – To the best of the authors' knowledge, MOL solution is proposed for the first time for the modeling of three-wave interaction in a SBS-based fiber optic sensor.

Keywords Simulation, Brillouin scattering, Finite difference, Flux limiter, Method of lines, Sparse matrix integrator

Paper type Research paper

1. Introduction

Stimulated Brillouin scattering (SBS) in optical fibers permits us to measure temperature and/or strain on a truly distributed basis, over kilometric ranges with high resolution (Horiguchi *et al.*, 1995; Bao *et al.*, 1997, 2001). The SBS effect arises from the interaction between two counter propagating light waves with frequency shift v and an acoustic wave of frequency ν, the latter being driven through an electrostrictive process where the medium becomes more dense in regions of high optical density. In this three-wave mixing process, power is transferred from the pump light wave to

The University of Hasan Kalyoncu Optical Fibre Systems Laboratory is supported in part by grants from the TUBITAK.

Engineering Computations: International Journal for Computer-Aided Engineering and Software
Vol. 31 No. 3, 2014
pp. 388-405

0264-4401
DOI 10.1108/EC-05-2012-0096

the Stokes light wave (that is, to the light wave having a lower frequency) and also to the acoustic wave. The interaction strength depends on the frequency offset between the two light waves and attains its maximum at the so-called Brillouin frequency v_B. As the Brillouin frequency shift changes linearly with temperature and strain, a distributed temperature-strain sensor can be realized using SBS.

The equations governing the three-wave SBS transient model (Chow and Bers, 1993), in one spatial dimension, z [m], with the time dependence t [s] included, can be expressed as:

$$\frac{\partial a_p}{\partial t} + \frac{c}{n}\frac{\partial a_p}{\partial z} + \gamma_p a_p = -K a_s a_a e^{-j\delta(z)t} \tag{1a}$$

$$\frac{\partial a_s}{\partial t} - \frac{c}{n}\frac{\partial a_s}{\partial z} + \gamma_s a_s = K^* a_p a_a^* e^{-j\delta(z)t} \tag{1b}$$

$$\frac{\partial a_a}{\partial t} + u\frac{\partial a_a}{\partial z} + \gamma_a a_a = K^* a_p a_s^* e^{-j\delta(z)t} \tag{1c}$$

where a_{p}, a_{s}, and a_{a} [W] are the pump, Stokes, and acoustic fields respectively. The terms, γ_* [*MHz*], represent each field's respective damping constant, K is the Brillouin coupling constant, c is the speed of light [*m/sec*], u is the speed of sound in the fiber [*m/sec*] and n is the refractive index of the fiber. The resonance detuning parameter, $\delta(z)$ [*MHz*] is dependent on the temperature and strain along the fiber. This dependence can form the basis of distributed fiber optic strain and temperature sensors (Lecoeuche *et al.*, 2000).

Simulation has proven to be a valuable tool in understanding SBS-based distributed sensor systems (Marble *et al.*, 2004; Kalosha *et al.*, 2006; Minardo *et al.*, 2011). Accurate simulation models allow for the rapid design and optimization of both sensor infrastructure and signal processing techniques. Visualization of the spatiotemporal behavior of all three fields involved in SBS, at all points through the fiber, is especially desirable in investigating phenomena whose origins may not be clear from the "observable" data in an experiment.

If long pulses are assumed, the time dependence in Equation (1) can be neglected, resulting in a simple and accurate solution to the problem. However, in the short pulse regime, the temporal derivatives cannot be neglected and the solution becomes more complex (Chow and Bers, 1993). For this solution, high speed and accuracy with the low complexity is highly desirable. In general, for the solution of Equation (1), conventional first order finite difference time domain (FDTD) solution schemes are preferred since they offer simplicity with respect to the form of the solution, although they suffer from several drawbacks, e.g. they require more spatial nodes to achieve the same accuracy as higher order schemes, and this often results in longer computer run-times.

To solve Equation (1), a numerical method based on the Simpson's rule to approximate temporal integrals was introduced by Chu *et al.* (1992). However, they used an implicit method and employed linearization by assuming that $|\mathrm{E}_{\mathrm{sn}}^{\mathrm{m}+1}| \approx |\mathrm{E}_{\mathrm{sn}}^{\mathrm{m}}|$ and this replacement weakens the coupling of the two laser fields and hence the results deviate from the exact solutions. Marble *et al.* (2004) proposed a modified solution which is valid only when the time step size is equal to spatial step size. However, this solution differs considerably from the real solution at locations where abrupt change takes place.

In this contribution we introduce, for the first time to the best of our knowledge, the solution of Equation (1) via Matlab ordinary differential equation (ODE) solvers (MATLAB, 2012). The problem is first discretized over a finite grid and manipulated to provide solutions for the optical and acoustic fields. The resolution of the solution can thus be varied to suit computational requirements and we have employed an appropriate flux limiter to combat numerical damping/dispersion. We have used software which identifies the sparsity pattern of the solution and this is supplied as an input to the solver. Additionally, we have determined the appropriate Matlab ODE solver for this particular application. By use of an appropriate sparse matrix ODE solver along with an effective flux limiter, it is shown that the computation can be successfully performed in a reasonable computation time without the damping factor exceeding 3 percent, even with narrow pulse widths. Examples are presented, showing the utility of this efficient simulation technique.

2. Problem formulation

2.1 The dynamics of SBS

In this paper we consider a system whereby continuous wave (CW) light, at frequency, v_p, is injected into the fiber at position $z = 0$. Pulses of light, known as Stokes pulses, are injected into the fiber at $z = L$, at frequency, v_s. Due to the SBS, there will be coupling of a counter propagating CW pump wave and a Stokes pulse wave via an induced acoustic wave. An enhanced interaction between the two beams occurs when the frequency difference of the lasers matches the frequency of the longitudinal acoustic phonons of the optical fiber. There is then a transfer of energy from the high frequency beam (pump) to the low frequency beam (Stokes). The amount of loss of the pump is recorded at $z = L$, as a function of the frequency difference in the form of a Brillouin spectrum and thus the power of the CW field intensity over time at $z = L$ corresponds to the available time domain information in a real sensing experiment. The maximum loss occurs when the frequency difference of two beams matches the Brillouin frequency of the fiber. This maximum loss depends on the resonance detuning parameter, given by:

$$\delta(z) = 2\pi \cdot \left[v_p - v_s - \delta v^{\mathrm{Res}}(z)\right] \tag{2}$$

where $\delta v^{\mathrm{Res}}(z)$ is the Brillouin frequency along the fiber and depends on the local strain and temperature. The resonance detuning parameter, $\delta(z)$, is the value of the pump-Stokes frequency shift for a resonant interaction at a given point. The resonance detuning parameter, $\delta(z)$ is dependent on the temperature and strain along the fiber. This dependence can form the basis of distributed fiber optic strain and temperature sensors.

A distributed Brillouin sensor, based on a Brillouin optical time domain analysis (BOTDA), is a device where Stokes and pump beams counter-propagate and where frequency and time dependent variations of the Stokes (or pump) intensity is detected. These sensors have been shown to provide the best performances in terms of sensing length, spatial resolution, temperature, and strain accuracies (Bao *et al.*, 1997; Zou *et al.*, 2004). The interaction is the maximum when the optical frequency difference between pump (v_p) and Stokes (v_s) corresponds to the Brillouin frequency (v_B). At the Brillouin frequency, the Stokes beam is amplified at the expense of the pump. v_B is the longitudinal acoustic phonon frequency and is a local material signature. An increase (decrease) in temperature or strain induces a proportional increase (decrease) of v_B. The sensor detects the Brillouin frequency variation as a function of position.

The distributed nature of the sensor is achieved by field modulation of the probe beam. The topology of this sensor is shown in Figure 1 (Ravet *et al.*, 2006).

2.2 Theoretical model

In Equation (1), the pump and Stokes waves travel at the velocity of light in the fiber, c/n, while the acoustic wave travels with velocity, u. The counter propagating nature of the Stokes and pump field is reflected in the choice of positive and negative signs for the spatial derivatives in Equation's (1a) and (1b). The resonance detuning parameter, $\delta(z)$, governs the frequency at which SBS occurs at position, z, along the fiber. The resonance detuning parameter is given by $\delta(z) = 2\pi[v_p - v_s - \delta v^{Res}(z)]$. In a modern optical fiber, losses are low, and thus the damping terms for the Stokes and pump waves, γ_s and γ_p can be neglected. The spatial derivative of the acoustic field can also be neglected since the interaction occurs on a much smaller timescale than the propagation of the acoustic wave. By normalizing the time and spatial variables according to $\gamma_a t \to t$, $z\gamma_a(n/c) \to z$, and making the substitutions $\Delta(z) = \delta(z)/\gamma_a$, $E_p = a_p K/\gamma_a$, $E_s = a_s K/\gamma_a$, and $E_a = (\, a_a K/\gamma_a\,)\, e^{-j\Delta(z)t}$, Equation (1) can be expressed as a system of three first-order hyperbolic partial differential equations (PDEs) (Chow and Bers, 1993; Lecoeuche *et al.*, 2000; Marble *et al.*, 2004):

$$\frac{\partial E_p}{\partial t} + \frac{\partial E_p}{\partial z} = -E_s E_a \tag{3a}$$

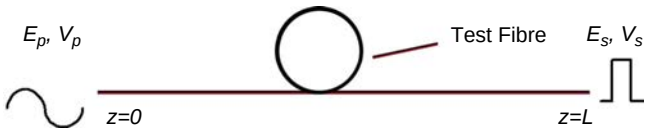
$$\frac{\partial E_s}{\partial t} - \frac{\partial E_s}{\partial z} = E_p E_a^* \tag{3b}$$

$$\frac{\partial E_a}{\partial t} + [1 + j\Delta(z)]E_a = E_p E_s^* \tag{3c}$$

2.3 Boundary and initial conditions

The CW light from a laser source representing the pump field, E_p is injected into the fiber at $z = 0$. Therefore, the boundary condition of the pump wave is:

$$E_p(z = 0, t) = C \tag{4}$$



Notes: The energy is transferred from the CW pump to the low frequency Stokes pulse. The amount of loss of the pump is recorded at $z = L$ as a function of the frequency difference in the form of the Brillouin spectrum and thus the power of the CW field intensity over time at $z = L$ corresponds to the available time domain information in a real sensing experiment

Figure 1.
Diagram of a typical Brillouin optical fiber time domain analysis (BOTDA) distributed sensor system

where C is the constant amplitude of the CW laser. At the opposite end, the Stokes field, E_s, is introduced via a pulsed source. The input Stokes pulse can be described by the two-kink profile function:

$$p(t) = [(\tanh(t_1) - \tanh(t_2))/2]^{1/2} \tag{4a}$$

where $t_{1,2} = (t \pm \tau_s/2)/a$ and τ_s [s] is the pulse duration. This function has a flat peak and a short rise time at the pulse edges as compared to the pulse duration and presents the typical shape of nanosecond optical pulses. The rise time is defined as the time interval between pulse power levels 0.1 and 0.9 of the peak power at leading and tailing edges; it is related to the parameter a as $t_{rise} = a/0.45$ and is independent of the pulse duration. We have assumed $t_{rise} = 0.1$ ns for all results presented below. Taking into account the existence of the base power, we model the boundary input Stokes pulse as (Kalosha *et al.*, 2006):

$$E_s(z = L, t) = (E_s - E_b)p(t) + E_b \tag{4b}$$

where $E_{s,b} = (P_{s,b}/A_{eff})^{1/2}$ for the field normalization Equations (3a)-(3c), P_s and P_b [W] are the peak and CW base powers of the Stokes pulse, respectively, and A_{eff} [m^2] is the fiber effective area. The base level of the Stokes pulse is characterized by the extinction ratio $ER = 10\log(P_s/P_b)$.

After propagating through the fiber, it is assumed that the pump and Stokes pulses exit the fiber without reflection. The acoustic field is assumed to be zero everywhere at $t = 0$, and the CW light and Stokes leakage intensities turn on everywhere at $t = 0$. Therefore the initial conditions are (Marble *et al.*, 2004):

$$E_p(z, t = 0) = C \tag{5a}$$

$$E_s(z, t = 0) = E_b \tag{5b}$$

$$E_a(z, t = 0) = 0 \tag{5c}$$

3. Numerical solution method

3.1 FDTD solution

To obtain a solution to Equations (3a)-(3c), we substitute the following (Chow and Bers, 1993):

$$E_p = A_p \cdot e^{j\Phi_p} \tag{6a}$$

$$E_s = A_s \cdot e^{j\Phi_s} \tag{6b}$$

$$E_a = A_a \cdot e^{j\Phi_a} \tag{6c}$$

into Equations (3a)-(3c), which yields the following equations:

$$\frac{\partial A_p}{\partial t} + \frac{\partial A_p}{\partial z} = -A_s A_a \cos(\Phi) \tag{7a}$$

$$\frac{\partial A_s}{\partial t} - \frac{\partial A_s}{\partial z} = A_p A_a \cos(\Phi) \tag{7b}$$

$$\frac{\partial A_a}{\partial t} + A_a = A_p A_s \cos(\Phi) \tag{7c}$$

$$\frac{\partial \Phi_p}{\partial t} + \frac{\partial \phi_p}{\partial z} = -\frac{A_s A_a}{A_p} \sin(\Phi) \tag{7d}$$

$$\frac{\partial \Phi_s}{\partial t} - \frac{\partial \phi_s}{\partial z} = -\frac{A_p A_a}{A_s} \sin(\Phi) \tag{7e}$$

$$\frac{\partial \Phi_a}{\partial t} + \Delta = -\frac{A_p A_s}{A_a} \sin(\Phi) \tag{7f}$$

where $\phi = \phi_a + \phi_s \phi_p$.

The steady state is obtained by setting the time derivatives to zero. Thus, combining Equations (7c) and (7f) yields:

$$A_a = A_p A_s \cos(\Phi) \tag{8a}$$

$$\Delta = -\frac{A_p A_s}{A_a} \sin(\Phi) \tag{8b}$$

and from Equations (8a)-(8b), we find $\tan \phi = -\Delta$. Taking the positive part of the amplitudes, from (7c) it can be concluded that:

$$\cos(\Phi) = \frac{1}{\sqrt{1+\Delta^2}} \tag{8c}$$

$$\sin(\Phi) = \frac{-\Delta}{\sqrt{1+\Delta^2}} \tag{8d}$$

The basic idea of the method of lines (MOL) is to replace the spatial derivatives with algebraic approximations (Schiesser and Griffiths, 2009). Since, spatial variables are discretized and time is used as the continuous variable, this approach is called a semidiscretization. This effectively removes the spatial derivatives from the PDE and, since only the initial value independent variable remains, e.g. t, the PDE has been converted to a system of approximating ODEs that can integrated by standard, well-established numerical algorithms for initial value ODEs.

Equations (3a)-(3c) are formulated by making some transformations which are $t_{real} \cdot \gamma_a = t_{transformed}$ and $z_{real} \cdot \gamma_a / (v) = z_{transformed}$, where v is the speed of the light in the

fiber ($v = c/n$). Thus, the problem is discretized over the transformed spatial and temporal domains:

$$(z_{transformed}, t_{transformed}) \in \Omega := \left\{ \begin{array}{l} 0 \leqslant \frac{z_{real}\gamma_a}{v} \leqslant z \\ 0 \leqslant t_{real} \cdot \gamma_a \leqslant z/v \end{array} \right\} \tag{9}$$

The solution is obtained on uniform grids in transformed z and transformed t:

$$w_{i,j} := \left\{ z_{transformed}(i) = i\Delta z, t_{transformed}(j) = j\Delta t, 0 \leqslant i \leqslant N, 0 \leqslant j \leqslant M \right\}$$

For the MOL solution of Equation (7a), $A_p(z, t)$ can be approximated by using upwind finite difference (FD) approximations for the first derivatives in z so that $A_p \approx (A_{\mathrm{p\_(i)}} - A_{\mathrm{p\_(i-1)}})/\Delta z$, where i is an index designating a position along a grid in z and Δz is the spacing in z along the grid (i−1 designates the upwind direction). For the MOL solution of Equation (7b), $A_s(z, t)$ can also be approximated by upwind FD approximations, i.e. $A_s \approx (A_{\mathrm{s\_(i+1)}} - A_{\mathrm{s\_(i)}})/\Delta z$ (i + 1 designates the upwind direction).

For the discretization of Equations (7a)−(7h), variables are defined as $y1 = A_p$, $y2 = A_s$, $y3 = A_a$, $y4 = \phi_p$, $y5 = \phi_s$, $y6 = \phi_a$. The boundary conditions are $y1$ ($z = 0$, t) $= C$, $y2$ ($z = L$, t) $= E_s(z = L, t)$, $y4$ ($z = 0$, t) $= 0$, $y5$ ($z = L$, t) $= 0$. The initial conditions are, $y1$ (z, $t = 0$) $= C$, $y2$ (z, $t = 0$) $= E_b$, $y3$ (z, $t = 0$) $= 0$, $y4$ (z, $t = 0$) $= 0$, $y5$ (z, $t = 0$) $= 0$, $y6$ (z, $t = 0$) $= 0$. By discretizing Equations (7a)-(7f) in the space variable z, the resulting system of ODEs is as follows:

$$\begin{aligned} \frac{dy1_i}{dt} &= -(y1_i - y1_{i-1})/(x_i - x_{i-1}) - y2_i \cdot y3_i \cdot \cos(\Phi) \qquad 2 \leqslant i \leqslant N \\ y1_1 &= C \end{aligned} \tag{10a}$$

$$\begin{aligned} \frac{dy2_i}{dt} &= -(y2_i - y2_{i+1})/(x_{i+1} - x_i) + y1_i \cdot y3_i \cdot \cos(\Phi) \qquad 1 \leqslant i \leqslant N-1 \\ y2_N &= E_s(z = L, t) \end{aligned} \tag{10b}$$

$$\frac{dy3_i}{dt} = -y3_i + y1_i \cdot y2_i \cdot \cos(\Phi) \qquad 1 \leqslant i \leqslant N \tag{10c}$$

$$\begin{aligned} \frac{dy4_i}{dt} &= -(y4_i - y4_{i-1})/(x_i - x_{i-1}) - y2_{,} \cdot y3_i/y1_i \cdot \sin(\Phi) \qquad 2 \leqslant i \leqslant N \\ y4_1 &= 0 \end{aligned} \tag{10d}$$

$$\begin{aligned} \frac{dy5_i}{dt} &= -(y5_i - y5_{i+1})/(x_{i+1} - x_i) - y1_i \cdot y3_i/y2_i \cdot \sin(\Phi) \qquad 1 \leqslant i \leqslant N-1 \\ y5_N &= 0 \end{aligned} \tag{10e}$$

$$\frac{dy6_i}{dt} = -\Delta - y1_i \cdot y2_i/y3_i \cdot \sin(\Phi) \qquad 1 \leqslant i \leqslant N \tag{10f}$$

In the appendix, the handling of Equations (10a)-(10f) is discussed. The solver solves the PDEs via the Matlab statement:

```
[t,u] = ode23t(@pde_1,tout,u0,options);
```

Using the initial conditions u0, transformed time interval `tout`, and `options` which includes tolerances and Jacobian sparsity.

3.2 Application of a sparse matrix integrator

For a large system of ODEs, it is typical that only a few components of y appear in each equation. If component j of y does not appear in component i of $f(t,y)$, then the partial derivative $\partial f_i/\partial y_j$ is zero. If most of the entries of a matrix are zero, the matrix is said to be sparse. By storing only the nonzero entries of a sparse Jacobian, storage is reduced from the square of the number of equations d to a modest multiple of d. If the Jacobian is sparse then so is the iteration matrix. As with storage, the cost of solving linear systems by elimination can be reduced dramatically by paying attention to zero entries in the matrix. By taking into account the known value of zero for most of the entries in the Jacobian, it is typically possible to approximate all the nonzero entries of several columns at a time. An important special case of a sparse Jacobian is one that has all its nonzero entries located in a band of diagonals (Shampine *et al.*, 2003; Shampine and Reichelt, 1997).

If PDE systems produced only banded Jacobians, then a banded integrator would be even more efficient than the sparse integrator since the banded integrator would know in advance where the nonzero elements occur (all are in the band) and it would therefore not have to search for the nonzero elements, and follow the resulting logic to use these nonzero elements, all of which add complexity to a sparse integrator. However, these nonzero elements are located/displayed with the Matlab command `spy( )`. For the PDE systems of Equations (10a)-(10f), some of the nonzero elements are located along the main diagonal as well as the others located outside the band. Therefore, for this case, a sparse integrator is particularly effective since it also operates on out-of-band elements, the so-called outliers, in performing the numerical integration. Thus, this combination of elements in a band along the main diagonal, plus outliers, leads to the efficiency of sparse matrix integration.

Since this particular application has a main diagonal, plus outliers, we have used the routine `jpattern_num` () to produce a Jacobian map for the sparse matrix option of the ODE solver. In this routine, the elements of the Jacobian matrix are computed by FDs. This requires a base point, or base points, around which the numerical derivatives are computed. A base point can be any value within the range of the variation of the associated dependent variable. A precise value is not required (but if ODE solver fails, some experimentation with this value may be required). In our application we set the base points `ybase(i)`, equal to the initial condition for each dependent variable which is sufficiently accurate to provide a good solution.

In order to calculate the partial derivatives in the Jacobian matrix by FD approximations, the derivatives at the base point, dy_i/dt, are also required. Note that the routine `pde_1()` (see Appendix), that calculates the derivatives dy_i/dt in Equations (10a)-(10f) is called by the Matlab command:

```
ytbase = pde_1(tbase,ybase);
```

The elements of the Jacobian matrix are evaluated numerically by the Matlab command:

```
[Jac,fac] = numjac(@pde_1,tbase,ybase,ytbase,thresh,fac,vectorized);
```

where the following argument values are used: `tbase = 0, thresh = 1x10^-16` and `fac = []`. The input variable thresh represents the threshold of significance for `Y(i)`, such that `Y(i) < thresh` is not important; the input variable `fac` represents a working storage column vector and should not be changed between calls; and

`vectorized` = "`off`" or "`on`" tells `numjac()` whether single or multiple values of Y, respectively, can be obtained with a single function evaluation.

Once the Jacobian matrix has been defined by this call to `numjac()`, a map of the Jacobian is plotted by the code that follows the call to `numjac()`. With a call to the routine `spones()`, each nonzero element is replaced by a "1" so as to create a "0-1" map of the Jacobian matrix (Schiesser and Griffiths, 2009).

```
S = spones(sparse(Jac));
```

In Figure 2, the sparsity pattern of Equations (10a)-(10f) is plotted by a subsequent call to the routine `spy(S)`. Code for `jpattern_num()` is available as a library routine which can be downloaded from www.pdecomp.net/TheCompendium/downloads.php under the compendiumSRC of Chapter 6 directory (accessed November 27, 2012).

Note that only 0.210 percent of the $6x300 \times 6x300$ elements of the Jacobian map of Figure 2 are nonzero. This is not uncommon; that is, most of the elements are zero. This condition is the reason for the use of sparse matrix integrators since they will in general process only the nonzero elements and will not expend computer time processing the many zero elements. The processing time with and without `jpattern_num()` is illustrated in Table I for the implicit (stiff) ODE solvers (ode15s, ode23s, ode23t, and ode23tb). The computation is performed on a i5 Core, 4GB RAM, Laptop computer running the Windows 7 operating system.

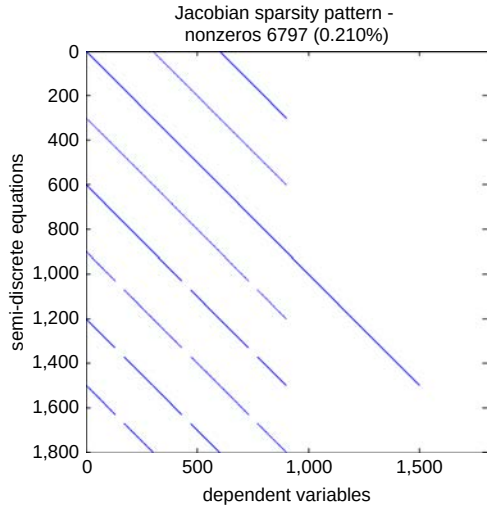


Figure 2.
Sparsity Pattern of the Equations (10a)-(10f)

Note: The solution is obtained with n=300

Table I.
Run-time comparison of implicit ODE solvers with/without use of jpattern_num()

	Run time (sec) AbsTol and RelTol = 1×10^{-5}, Pulse width = 5 ns. *Without* Flux-limiter Jpattern_num	
$n = 300$	*Without*	*With*
ode15s	102	29
ode23t	121	35
ode23tb	146	42
ode23s	Computation failed	61

From Table I, it can be seen that the saving in computer time can be very substantial. It is clear that the additional logic of the sparse matrix integrator (to detect the nonzero elements of the Jacobian matrix and then perform the numerical integration using only these nonzero elements) is well worth the additional complexity of the coding for the sparse matrix. Moreover, for ode23s, using `Jpattern_num()` is the difference between success and failure.

3.3 Use of a flux limiter

Numerical dissipation is generally observed when solving strongly convective or strongly hyperbolic PDEs. The main cause of the dissipation is low order approximation of the discretization. It should be noted that numerical dissipation produced by first order approximations is to be expected due to truncation of the Taylor series that is the basis for the approximations. In recent years, various high-resolution schemes have been developed to obviate this effect with a high degree of accuracy, albeit at the expense of algorithmic and computational complexity. Examples of particularly effective schemes are based upon flux/slope limiters (Wesseling, 2001) and WENO methods (Shu, 1998).

Flux limiters are used in numerical schemes to solve problems in science and engineering that are described by PDEs. Their main purpose is to avoid the spurious oscillations (wiggles) that would otherwise occur with high-order spatial discretization due to shocks, discontinuities, or steep gradients in the solution domain. They can be used directly on FD schemes for simple applications. Use of flux limiters, together with an appropriate high-resolution scheme, makes the solutions total variation diminishing. Flux limiters are also referred to as slope limiters because they both have the same mathematical form and both have the effect of limiting the solution gradient near shocks or discontinuities. In general, the term flux limiter is used when the limiter acts on system fluxes, and slope limiter is used when the limiter acts on system states. The main idea behind the construction of flux limiter schemes is to limit the spatial derivatives to realistic values; this usually means physically realizable values. They are used in high-resolution schemes for solving problems described by PDE's and only come into operation when sharp wave fronts are present (Griffiths and Schiesser, 2011).

Consider the semidiscrete scheme below:

$$\frac{du_i}{dt} + \frac{1}{\Delta x_i}[F(u_{i+\frac{1}{2}}) - F(u_{i-\frac{1}{2}})] = 0 \tag{11a}$$

where for a finite difference scheme, $F(u_{i+1/2})$ and $F(u_{i-1/2})$ represent flux values on the grid at point $x = x_{i+1/2}$ and point $x = x_{i-1/2}$. If these fluxes can be represented by low- and high-resolution schemes, then a flux limiter can switch between these schemes depending upon the gradients close to the particular cell as follows:

$$F(u_{i+\frac{1}{2}}) = f^{low}_{i+\frac{1}{2}} - \phi(r_i)(f^{low}_{i+\frac{1}{2}} - f^{high}_{i+\frac{1}{2}}) \tag{11b}$$

$$F(u_{i-\frac{1}{2}}) = f^{low}_{i-\frac{1}{2}} - \phi(r_{i-1})(f^{low}_{i-\frac{1}{2}} - f^{high}_{i-\frac{1}{2}}) \tag{11c}$$

where f^{low} low-resolution flux, f^{high} high-resolution flux, $\phi(r)$ = flux limiter function, and r represents the ratio of successive gradients on the solution mesh, i.e.:

$$r_i = (u_i - u_{i-1})/(u_{i+1} - u_i) \tag{11d}$$

The limiter function is constrained to be greater than or equal to zero, i.e. $r \geqslant 0$. Therefore, when the limiter is equal to zero (sharp gradient, opposite slopes, or zero gradient), the flux is represented by a low-resolution scheme. Similarly, when the limiter is equal to 1 (smooth solution), it is represented by a high-resolution scheme.

The various limiters listed below have differing switching characteristics and are selected to suit the particular problem and numerical solution scheme. No particular limiter has been found to work well for all problems, and a particular choice is usually made on a trial-and-error basis (Griffiths and Schiesser, 2011).

van Leer:

$$\phi_{vl} = \frac{r + |r|}{1 + r}; \quad \lim_{r \to \infty} \phi_{vl}(r) = 2 \tag{12a}$$

smart:

$$\phi_{sm}(r) = \max[0, \min(2r, (0.25 + 0.75r), 4]; \quad \lim_{r \to \infty} \phi_{sm}(r) = 4 \tag{12b}$$

superbee:

$$\phi_{sb}(r) = \max[0, \min(2r, 1), \min(r, 2)]; \quad \lim_{r \to \infty} \phi_{sb}(r) = 2 \tag{12c}$$

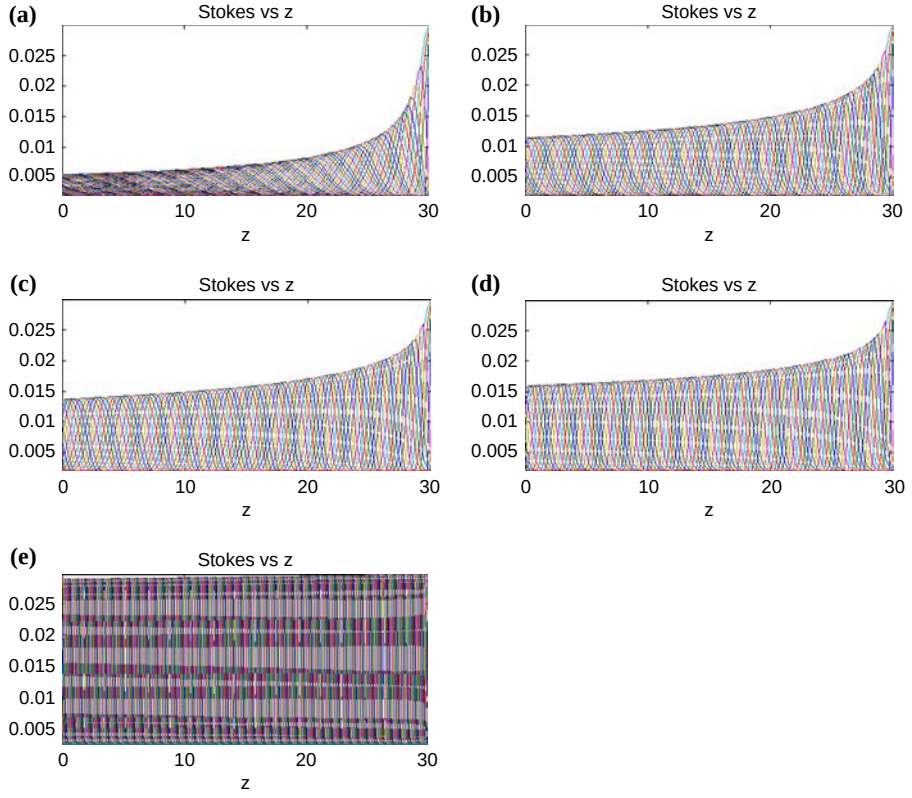
The parameters of the flux limiter, which available as a library routine which can be downloaded from www.pdecomp.net/TravelingWaves/downloadsTW.php (accessed November 27, 2012) are as follows:

```
y1x = -flux_function(xl,xu,n,y1,1);
y2x = flux_function(xl,xu,n,y2,-1);
```

The fifth parameter is direction of the flow for the linear advection equation $u_t + cu_x = 0$. This argument is required, since the limiter requires the direction of flow or wave propagation. It takes the values $+1$ or -1 depending on the direction of the upwinding of the FD. The fifth parameter value of -1 for `y2x` indicates that the pulse moves right to left in z at velocity $c = -1$ (i.e. counter propagating in z). The spatial domain in z is defined as $xl \leqslant z \leqslant xu$ with n points. The interval *xu-xl* is determined by the fiber length.

In order to establish the best flux limiter function to suit our particular application we performed numerical experiments to compute the degree of smearing. In Figure 3, the effect of *van Leer*, *Smart* and *Suberbee* limiters in terms of smearing can be seen.

In Figure 3(a), it is clearly seen that the Stokes magnitude is damped due to the numerical dissipation. In Figure 3(b-e), the effect of different flux limiters on the Stokes pulse is demonstrated. Damping which can be considered as the amplitude ratio at $z = 0$ and at $z = L$ is calculated as, Figure 3(b) 62.2 percent for *van Leer*, Figure 3(c) 55.44 percent for *smart* Figure 3(d) 47.74 percent for *superbee* flux limiter, respectively ($n = 100$). In Figure 3(e), the damping effect is decreased to only 2.8 percent with *superbee* flux limiter for 300 discretization points. In Figure 3(b-e), it is clearly seen that, Superbee is more capable of recovering numerical damping compared with *van Leer* and *Smart* flux limiter for this particular application. Since its superior effect, in this application, *Superbee* flux limiter is preferred. Additional performance data is given in section 4.



Notes: (a) Without using flux limiter with using; (b) van Leer; (c) smart; (d) superbee flux limiter with n=100 discretization points. In (e) superbee flux limiter is used with n=300 discretization points

Figure 3.
The simulation of the pulse

4. Example

4.1 Numerical results

The simulation models an optical fiber, 30 m in length, and the parameter values of the simulation are inferred from the paper (Marble *et al.*, 2004). The Brillouin frequency, $\delta v^{\mathrm{Res}}(z)$, is 12.780 GHz along the fiber, except for a 10 m section in the center where $\delta v^{\mathrm{Res}}(z) = 12.500$ GHz, representing a strained section. The moment in time, when the Stokes pulse enters the fiber is assumed to be equal to $t_0 = 1$ ns with respect to the pulse leading edge. The other relevant parameters are: CW power = 3 mW; Stokes power = 30 mW; Stokes pulse rise time = 100 ps; pulse width = 5 ns; extinction ratio (ER) 12 dB. The simulation models a pulse on-resonance, with the strained section ($v_{\mathrm{p}} - v_{\mathrm{s}} = 12.500$ GHz). The acoustic damping constant γ_{a}, was chosen to be 110 MHz which reflects the 35 MHz Brillouin linewidth of standard SMF28 fiber. $A_{\mathrm{eff}} = 50\,\mathrm{m}^2$, $g_{\mathrm{B}} = 5 \times 10^{11}$ m/W.

The expected result of this simulation is that the Stokes pulse will propagate down the section of fiber with no loss or dispersion. Since the fiber's natural $\delta v^{\mathrm{Res}}(z)$ is much different from $v_p - v_s$, there should be little interaction in the "unstrained" portions of the fiber. Upon entering the "strained" section, the pulse will interact with the CW field,

causing a rise in the acoustic field intensity. The CW field will be depleted, and the pulse will be enhanced. The range of the depletion depends on the extinction ratio and the interaction is stronger for a low extinction ratio. After the Stokes pulse passes through the strained section, the acoustic field will decay off and the CW field will return to its steady state value. The effect of the interaction will take the form of a depleted region of CW light propagating to the end of the fiber (Marble *et al.*, 2004).

Figure 4 shows the Stokes, CW, and acoustic fields, simulated with and without the *superbee* flux limiter. As can be seen in Figure 4(a), dissipation and damping is clearly observed. Over time, the Stokes pulse broadens, as well as decreases in intensity. However, the expectation from this simulation is that Stokes pulse will propagate

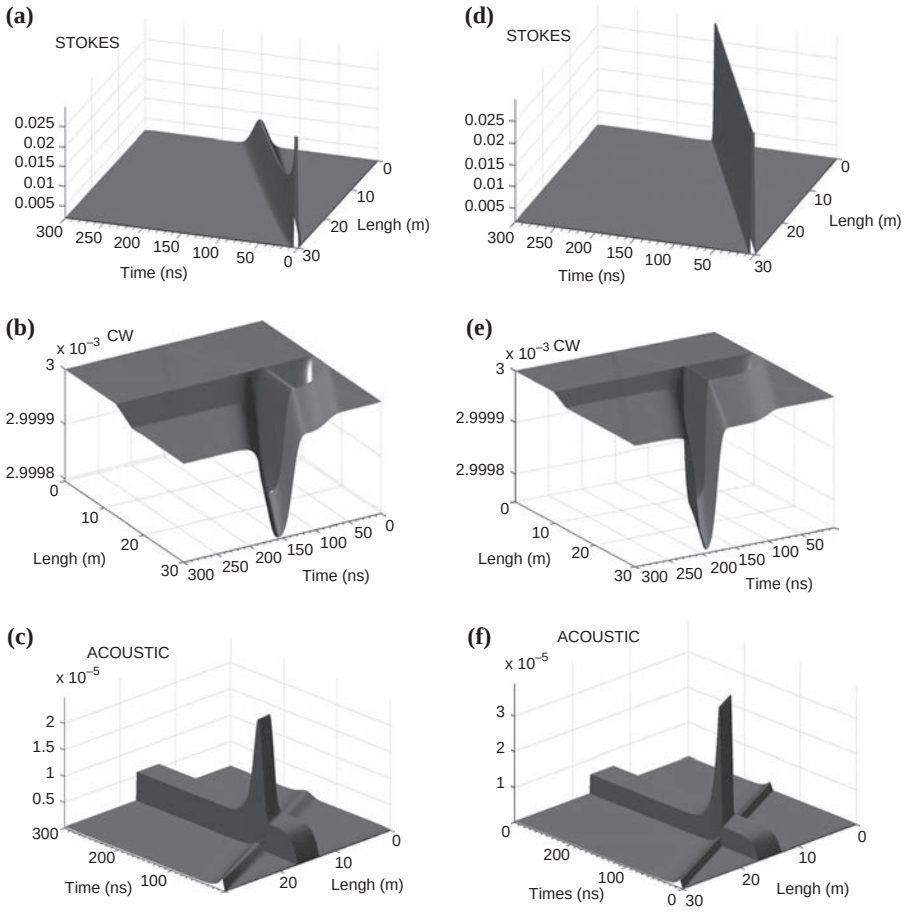


Figure 4.
Using the simulation of Equations (10a)-(10f) with $n = 300$ discretization points, evolution of the Stokes (*y2*), CW (*y1*), and Acoustic (*y3*) fields for a 30 m section of fiber is plotted in (a-d), (b-e) and (c-f), respectively

Notes: The frequency difference of CW pump and Stokes pulse wave was set to be on resonance with a 10 m section in the center of the fiber. Since, in plots (a), (b) and (c), flux limiter is not used, the dispersion and attenuation is evident. Plots (d), (e), and (f) show the results when the flux limiter is used. Compared with the low order discretization, with the introduction of the flux limiter, dispersive attenuation (smoothing of sharp edges) along the fiber obviously is eliminated

down the section of fiber with no loss or dispersion. It must be noted that, in the "unstrained" portions of the fiber $\Delta \neq 0$, i.e., $\nu_p - \nu_s \neq \delta\nu^{Res}(z)$, in the "strained" section of the fiber $\Delta = 0$, i.e. $\nu_p - \nu_s = \delta\nu^{Res}(z)$. Plots Figure 3(d-f) show the fields simulated with the *superbee* flux limiter. Especially in plot (d), with flux limiter, it can be clearly seen that damping is suppressed and the ration of the amplitudes at $z = 0$ and at $z = L$ becomes 0.972 which means that Stokes wave lost only 2.8 percent of its magnitude. Another expectation of the simulation is that, when the pulse enters the strained section there should be a rise in the acoustic field intensity. However, the *superbee* flux limiter prevents this rise during stabilization.

4.2 Comparison of Matlab ODE solver performances

A time domain measurement of the CW light exiting the fiber gives spatially resolved information about the local value of $\delta\nu^{Res}(z)$. Thus, computing the CW traces is particularly important for BOTDA sensors. To better understand the performance of ODE solvers we made some numerical experiments regarding with AbsTol, RelTol, and number of discretization values.

To better illustrate the performance of the ode15s solver, two traces of the CW field intensity over time at $z = L$ 30 m from the simulation described above are shown in Figure 5. The dip in each trace corresponds with the 10 m strained section in the center of the fiber. Up to $n = 300$ points, implicit solvers show the same performance in terms of oscillation. However, more than $n = 300$ points ode15 solver oscillate for AbsTol and Rel Tol $= 10^{-5}$. However, after improving the tolerances to 10^{-6}, oscillation disappears.

As can be seen in Figure 6, even with the high tolerance values, ode15s cannot suppress the oscillation of ϕ_p (*y4*). However, ode23tb and ode23t solvers are successful in handling oscillation. The numerical experiment in Figure 6 is performed with RelTol and AbsTol $= 10^{-6}$ using 300 discretization points. Even for more stringent tolerances and discretization points, ode15s can extinguish but cannot fully eliminate the oscillation in computing ϕ_p.

As the pulse width decreases, damping is more pronounced and the suppression can only be coped with improving the discretization points. In Table II, pulse with and corresponding discretization points to guarantee the amplitude ratio at $z = 0$ and at $z = L$ as 97 percent is shown. The computation is performed using results from the `jpattern_num()` routine.

It must be emphasized that, only the solvers `ode15s`, `ode23t`, `ode23b`, `ode23s` could solve the equations. The computation times of the mentioned solvers are shown

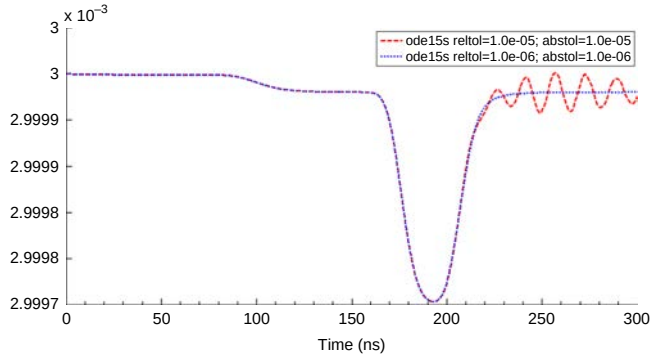


Figure 5.
Comparison of ode15s performance in terms of AbsTol, RelTol values for 400 discretization points

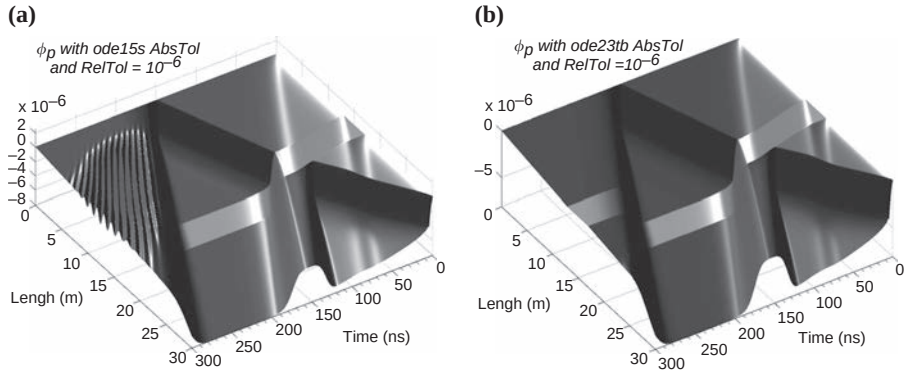


Figure 6.
Comparison of (a) ode15s and (b) ode23tb for the computation of ϕ_p

Notes: Simulation parameters are AbsTol, RelTol =10^{-6} and n=300

Table II.
Run-time and discretization number comparison of implicit ODE solvers to guarantee the maximum damping of 3 percent

	Run Time (sec) (*using* jpattern_num() with AbsTol and RelTol = 1×10^{-5}) with flux limiter *Pulse width*			
	5 ns (300 points)	4 ns (400 points)	3 ns (650 points)	2 ns (1,050 points)
ode15s	124	260	661	1749
ode23t	149	306	731	1827
ode23tb	186	400	953	2382
ode23s	379	690	1700	4352

in Table II. The computation time of `ode23s` is considerably higher and thus it is not effective. As per Table II and Figure 6, it can be concluded that, if it is intended to use `ode15s`, `AbsTol`, and `RelTol` values should be in the range of 10^{-7} (otherwise it cannot eliminate oscillations) which in turn increases the computation time. Flux limiter application which is necessary for the elimination of numerical damping increases the computation time considerably for the pulse width below 4 ns. It must be emphasized that without using the `Jpattern` routine the computation becomes considerably time consuming. For this particular application, `ode23t` can be regarded as suitable solver since it does not oscillate and work well in moderate `AbsTol` and `RelTol` values (1×10^{-5}). Also, its run time is better than the `ode23tb` and `ode23s` solvers. In Table III, the computation times of the solvers versus `AbsTol` and `RelTol` values are shown.

Table III.
Run-time comparison of implicit ODE solvers vs different AbsTol and RelTol values

$n = 100$	Run Time (sec) (*using* jpattern_num(), with 5 ns pulse width, $n = 300$) with flux limiter AbsTol and RelTol = 1×10^{-5}	AbsTol and RelTol = 1×10^{-6}	AbsTol and RelTol = 1×10^{-7}
ode15s	20	30	49
ode23t	23	41	81
ode23tb	28	83	105
ode23s	63	115	257

5. Conclusions

The MOL solution to the three-wave interaction problem of SBS, representing the typical BOTDA distributed sensor system, is demonstrated for the first time to the best of our knowledge. Thanks to the MOL scheme, the solution is adapted to the proper mesh size allowing rapid calculation of the fields involved in three-wave interaction. Numerical damping is suppressed by the introduction of an appropriate flux limiter. Numerical efficiency is achieved by the use of the sparsity of the Jacobian matrix. It is demonstrated that, `ode23t` is more suitable for the solution of three-wave SBS equations since it has better performance in terms of running time and nonoscillatory computation. The simulation method presented here can be used to foster a rapid understanding of the internal dynamics of the three-wave SBS equations, lending insight for both experimental design and interpretation of results.

References

Bao, X., DeMerchant, M., Brown, A. and Bremner, T. (2001), "Tensile and compressive strain measurement in the lab and field with the distributed Brillouin scattering sensor", *J. Lightwave Technol*, Vol. 19 No. 11, pp. 1698-1704.

Bao, X., Dhliwayo, J., Heron, N., Webb, D.J. and Jackson, D.A. (1997), "Experimental and theoretical studies on a distributed temperature sensor based on Brillouin scattering", *J. Lightw. Technol*, Vol. 15 No. 10, pp. 1842-1851.

Chow, C.C. and Bers, A. (1993), "Chaotic stimulated Brillouin scattering in a finite-length medium", *Physical Review A*, Vol. 47 No. 6, pp. 5144-5150.

Chu, R., Kanefsky, M. and Falk, J. (1992), "Numerical study of transient stimulated Brillouin scattering", *J. Appl. Phys*, Vol. 71 No. 10, pp. 4653-4658.

Griffiths, G.W. and Schiesser, W.E. (2011), *Traveling Wave Solutions of Partial Differential Equations: Numerical and Analytical Methods with Matlab and Maple*, Academic Press, Burlington, MA.

Horiguchi, T., Shimizu, K., Kurashima, T., Tateda, M. and Koyamada, Y. (1995), "Development of a distributed sensing technique using Brillouin scattering", *J. Lightwave Technol*, Vol. 13 No. 7, pp. 1296-1302.

Kalosha, V.P., Ponomarev, E., Chen, L. and Bao, X. (2006), "How to obtain high spectral resolution of SBS-based distributed sensing by using nanosecond pulses", *Opt. Express*, Vol. 1 No. 6, pp. 2071-2078, available at: www.opticsinfobase.org/oe/abstract.cfm?URI=oe-14-6-2071 (accessed November 27, 2012).

Lecoeuche, V., Webb, D.J., Pannell, C.N. and Jackson, D.A. (2000), "Transient response in high-resolution Brillouin-based distributed sensing using probe pulses shorter than the acoustic relaxation time", *Optics Letters*, Vol. 25 No. 3, pp. 156-158.

Marble, A.E., Brown, K.A. and Colpitts, B.G. (2004), "Stimulated Brillouin scattering modeled through a finite difference time domain approach", *Proc. SPIE 5579*, Photonics North 2004: Photonic Applications in Telecommunications, Sensors, Software, and Lasers, November, pp. 404-414, doi:10.1117/12.567356, available at: http://dx.doi.org/10.1117/12.567356

MATLAB (2012), *MathWorks*, MathWorks Inc., Natick, MA, available at: www.mathworks.com/help/pdf_doc/matlab/getstart.pdf (accessed November 27, 2012).

Minardo, A., Bernini, R. and Zeni, L. (2011), "Numerical analysis of single pulse and differential pulse-width pair BOTDA systems in the high spatial resolution regime", *Opt. Express*, Vol. 19 No. 20, pp. 19233-19244.

Ravet, F., Chen, L., Bao, X., Zou, L. and Kalosha, V. (2006), "Theoretical study of the effect of slow light on BOTDA spatial resolution", *Opt. Express*, Vol. 14 No. 22, pp. 10351-10358.

Schiesser, W.E. and Griffiths, G.W. (2009), *A Compendium of Partial Differential Equation Models: Method of Lines Analysis with Matlab*, Cambridge University Press, New York, NY.

Shampine, L.F. and Reichelt, M.W. (1997), "The MATLAB ODE Suite", *SIAM J. Sci. Comput*, Vol. 18 No. 1, pp. 1-22.

Shampine, L.F., Gladwell, I. and Thompson, S. (2003), *Solving ODEs with MATLAB*, 1st ed., Cambridge University Press, New York, NY.

Shu, C.-W. (1998), "Essentially non-oscillatory and weighted essential non-oscillatory schemes for hyperbolic conservation laws", in Cockburn, B., Johnson, C., Shu, C.-W. and Tadmor, E. (Eds), *Advanced Numerical Approximation of Nonlinear Hyperbolic Equations, Lecture Notes in Mathematics*, *Vol. 1697*, Springer, Berlin, pp. 325-432.

Wesseling, P. (2001), *Principles of Computational Fluid Dynamics*, Springer, Berlin.

Zou, L., Bao, X., Afshar, V.S. and Chen, L. (2004), "Dependence of the Brillouin frequency shift on strain and temperature in a photonic crystal fiber", *Opt. Lett*, Vol. 29 No. 13, pp. 1485-1487.

Appendix. Matlab code for the function `pde_1()`

```
function ut = pde_1(t,u)

% Problem parameters
global xl xu n ncall coeff1 coeff2 dx1 dx2
global delta1 delta2
global coeffsin1 coeffsin2

i = 1:n; % spatial index
i1 = i(dx1<=i & i<=dx2); % 'inner' region
i2 = i(i<dx1 | dx2<i); % 'outer' region

coeff = zeros(n,1);
coeff(i1) = coeff1;
coeff(i2) = coeff2;
coeffsin = zeros(n,1);
coeffsin(i1) = coeffsin1;
coeffsin(i2) = coeffsin2;

delta = zeros(n,1);
delta(i1) = delta1;
delta(i2) = delta2;

% u to y, column vectors
u = reshape(u,n,6);
y1 = u(:,1);
y2 = u(:,2);
y3 = u(:,3);
y4 = u(:,4);
y5 = u(:,5);
y6 = u(:,6);

% BCs
y1(1)=3e-3;
y2(n)=pulse(t);
y4(1)=0;
y5(n)=0;
```

```
% 'pure' spatial derivative
% y1x = [0;-(y1(2:n)-y1(1:n-1))/dx];
% y2x = [-(y2(1:n-1)-y2(2:n))/dx;0];
% y4x = [0;-(y4(2:n)-y4(1:n-1))/dx];
% y5x = [-(y5(1:n-1)-y5(2:n))/dx;0];

% spatial derivative with limiter
y1x=-super(xl,xu,n,y1,1);
y2x=super(xl,xu,n,y2,-1);
y4x=-super(xl,xu,n,y4,1);
y5x=super(xl,xu,n,y5,-1);

% d(y1)/dt
y1t = y1x' - y2.*y3.*coeff;
y1t(1) = 0; % boundary condition

% d(y2)/dt
y2t = y2x' + y1.*y3.*coeff;
y2t(n) = 0; % boundary condition

% d(y3)/dt
y3t = y1.*y2.*coeff - y3;

% d(y4)/dt
y4t = y4x' - y2.*y3./y1.*coeffsin;
y4t(1) = 0; % boundary condition

% d(y5)/dt
y5t = y5x' - y1.*y3./y2.*coeffsin;
y5t(n) = 0; % boundary condition

% d(y6)/dt
y6t = -y1.*y2./y3.*coeffsin - delta;

% yt to ut, column vector
ut = [y1t;y2t;y3t;y4t;y5t;y6t];

% Increment calls to pde_1
ncall = ncall+1;
```

Corresponding author
Assistant Professor Fikri Serdar Gokhan can be contacted at: fsgokhan@gmail.com