COLLISIONS OF ORTHOGONALLY POLARIZED SOLITARY WAVES

by

John Demeritt Moores

B.S., Electrical Engineering
Rensselaer Polytechnic Institute
(1986)

Submitted in partial fulfillment
of the requirements for the Degree of

Master of Science
in Electrical Engineering and Computer Science
at the

Massachusetts Institute of Technology

January 1989
© John D. Moores 1989. All rights reserved

The author hereby grants to MIT permission to reproduce and to
distribute copies of this thesis document in whole or in part.

Signature of Author

Department of Electrical Engineering and Computer Science
November 21, 1988

Certified by

Hermann A. Haus
Institute Professor of Electrical Engineering
Thesis Supervisor

Accepted by

Arthur C. Smith
Chairman, Department Committee on Graduate Students

MASSACHUSETTS INSTITUTE
OF TECHNOLOGY

MAY 10 1989

LIBRARIES
Acknowledgements

First, I wish to thank Prof. Hermann A. Haus for his support and insightful guidance. Prof. Haus is indeed an inspiration to me.

I am grateful to the CNSF and IBM for providing our group the opportunity to employ supercomputing power in our modelling.

Thanks are due to MIT's Research Laboratory of Electronics (RLE) for access to its IBM 4381 Model 14, and to RLE and IBM for providing an IBM PS/2 Model 80 with color display upon which this document was prepared, and which serves as a rather smart terminal.

Dave Foss, System Manager of the RLE mainframe has greatly facilitated interfacing to the mainframe, to personal computers, and to the Cornell National Supercomputing Facility (CNSF).

Finally, I wish to thank the Office of Naval Research for generous financial support in the form of a 1986-1989 ONR Fellowship.
COLLISIONS OF ORTHOGONALLY POLARIZED SOLITARY WAVES

by

John Demeritt Moores

Submitted to the
Department of Electrical Engineering and Computer Science
on November 21, 1988 in partial fulfillment of the requirements
for the Degree of Master of Science
in Electrical Engineering and Computer Science.

Abstract

Collisions of orthogonally polarized solitary waves can be applied to all-optical switching. The governing equations for unidirectional propagation of orthogonal linearly polarized waves, which are coupled nonlinear Schrödinger equations, are derived. The derivation accounts for both natural linear birefringence and for waves of different frequencies in the two channels. A perturbative approach to estimating effects of collisions is developed. Collisions are simulated numerically, and certain results are compared with the perturbation theory. Several approaches to switching have been investigated and are discussed. Suggestions for further work are included.

Thesis Advisor: Hermann A. Haus
Title: Institute Professor of Electrical Engineering
Biographical note

EDUCATION

Rensselaer Polytechnic Institute Troy, NY
B.S. degree in Electrical Engineering, May 1986.
Graduated summa cum laude. Completed Electrical, Computer, and
Systems Engineering Honors Program. Physics Honors course se-
quence. Grade Point Average: 3.975/4.0.

EXPERIENCE

S.D. Warren Research Laboratory, Westbrook, ME

December 1985 to January 1986
Computer simulation consultant.

June through August each of three years: 1986, 1985, 1984
Designed, tested and implemented finite element and finite difference
dynamic simulation software in FORTRAN on a DEC VAX 11/750.
Employed software in thermal stress and heat/mass transport analysis
for several proprietary projects.

Dept. of Mathematics, RPI

January 1985 to May 1985
Grader for undergraduate complex analysis course.

HONORS

Office of Naval Research Fellow 1986-1989
Tau Beta Pi Fellow 1986-1987
David M. Darrin Scholar, RPI 1982-1986
National Merit Scholar 1982-1983
High School Quiz Prizes
Member Eta Kappa Nu, IEEE, Pi Mu Epsilon, Tau Beta Pi

BACKGROUND

U.S. citizen, born in Portland, ME (10/9/64). At RPI, editor of Rens-
sselaer Engineer magazine, three time member Wm. Lowell Putnam
Mathematical Competition team, epee fencer, member Astrophysical
Society. Percussionist. Enjoy early music and art, church activities,
weightlifting, hiking, canoeing, racquetball, recreational mathematics.
# Contents

1 Introduction .............................................. 6

2 Coupled Nonlinear Schrödinger Equations ................. 10

3 Perturbation Theory ........................................ 15
   3.1 Derivation ........................................... 17
   3.2 Application - Simplified, Analytic Theory .......... 19

4 Numerical Analysis ....................................... 23
   4.1 Software ............................................ 24
   4.2 Pole and residue extraction .......................... 26
   4.3 Examples of collision simulations .................. 29
   4.4 Comparison with perturbation theory ............... 43
   4.5 Alternate approaches to switching .................. 43

5 Discussion ................................................. 55

A Simulation code .......................................... 57

B Sample input datafile ..................................... 85
Chapter 1

Introduction

Optical solitary waves in single-mode fibers have tremendous potential for high bit rate, high bandwidth communications. The need for an all-optical switch for fiber networks is the motivation for this work.

What exactly is a solitary wave? It is simply a localized disturbance or wavepacket whose derivatives with respect to the coordinate along which it propagates vanish at ±∞. This definition allows for different equilibrium values at each side of the wave. So-called kink soliton solutions, with nonzero values at infinity, exist for certain equations such as the sine-Gordon. For optical pulses, however, the relevant equilibrium values are zero. It is difficult to find a rigorous definition of "solitary wave," but some authors (e.g. [1]) restrict the application of the term to "single humps." If this refers to entities with single maxima, then we shall allow for more general pulses which although still localized may "wiggle." Our initial conditions will be exclusively single-humped in this work, but we shall find that collisions may give rise to more complicated structures.

A soliton is a specific type of solitary wave. Solitons propagate in nonlinear,
dispersive media as a balance between the competing effects of the nonlinearity and dispersion. Launch a soliton and it will either maintain a permanent profile or deform and return to its initial profile periodically. Solitons neither form shocks nor disperse. The most stringent test of soliton behavior involves collisions. If two pulses with the above properties collide, and the effect of the scattering is merely to delay or advance and/or phaseshift (in the case of envelope solitons—discussed below) the pulses, then they qualify as solitons. The optics community has tended to be somewhat careless in applying the term “soliton” to pulses which in isolation behave like solitons, but which do not satisfy the collision condition.

In 1971, Zakharov and Shabat [2] discovered soliton solutions to the cubic nonlinear Schrödinger equation (NLS) via inverse scattering techniques. Amongst its applications, as described by Hasegawa and Tappert in 1973 [3], the NLS is a simple model for pulse propagation in single-mode optical fibers. This discovery sparked interest in the use of solitonlike pulses in optical communications systems. Interest increased dramatically with the successful generation of such pulses in 1980 [4].

If we restrict ourselves to a discussion of pulses in optical fibers, we can elaborate on soliton behavior. The NLS exhibits envelope soliton solutions. These are wavepackets with carriers travelling at the phase speed and envelopes travelling at the group speed. It is the envelope which possesses soliton properties. The presence of the carrier permits a collisional phaseshift. The nonlinearity is balanced by material dispersion, resulting from a frequency-dependent permittivity, and to a lesser extent by waveguide dispersion [5].

There are several reasons why optical fibers are desirable media for construct-
ing all-optical switches. They exhibit relatively strong nonlinearity, which is employed advantageously by confining pulses within the medium for long distances, and they are very low loss media. The fact that they allow soliton propagation is very important as well. As suggested above, in the absence of collisions, lowest-order-solitonlike solitary waves are very robust, maintaining their shapes and spectra for very long distances. While this is wonderful for communications, we must investigate the effects of collisions if we intend to use them in switches.

One potential means for constructing an all-optical switch is to make use of the fact that a so-called single-mode fiber is actually bimodal in the presence of birefringence. Linearly birefringent fibers are available at present, while circularly or elliptically birefringent fibers are speculative. The proposed switching mechanism is easily described by comparison with a standard waveguide Mach-Zender interferometer (see e.g. [5]). An incoming signal pulse is split into two equal pulses which travel in separate interferometer arms. Switching is achieved by introducing a $\pi$-phaseshift to one arm, resulting in destructive interference when the two pulses are recombined. Our proposed switch involves the collision of the signal pulse in one arm with an orthogonally polarized control pulse. We wish to introduce a $\pi$-phaseshift to the signal pulse via this scattering process. It is our intent to predict whether or not this mechanism is feasible.

To conclude the introduction, we summarize the contents of this work. In the following chapter, we shall derive the coupled nonlinear Schrödinger equations, which model pulse propagation in single-mode fibers with linear birefringence. Chapter 3 involves a perturbative approach to predicting the effects of collisions upon solitary waves. The method is explained in detail and is applied, in a sim-
plied and fully analytic form, to the specific case of collisions. Chapter 4 first describes our simulation software, which provides a much more reliable means of determining pulse behavior. Examples of simulation results are presented. One set of simulations is compared with the perturbative approach. A number of alternative switching approaches, each involving a pair of orthogonal pulses, are discussed. Finally, in Chapter 5, conclusions will be drawn concerning the usefulness of each method of estimating scattering effects. We shall also address the issue of the effectiveness of solitary wave collisions in providing the phase-shifts required for optical logic gate operation. Suggestions for further work are interspersed within each chapter.
Chapter 2

Coupled Nonlinear Schrödinger Equations

Much of the notation adopted here is due to Menyuk [6]. However, we extend Menyuk's derivation to allow for orthogonal pulses of slightly different carrier frequencies.

The geometry of the fiber, ignoring (small) ellipticity in cross-section, suggests the use of cylindrical coordinates. The electric field within the fiber may then be expressed as the superposition of two polarizations:

\[
E(\rho, \phi, z, t) = \sum_{j=1}^{n} U^{(j)}(z, t) R^{(j)}(\rho, \phi) e^{i(k^{(j)}z - \omega^{(j)}t)} \hat{e}_j + \text{c.c.} \tag{2.1}
\]

where \( j \) labels the polarization with \( \hat{e}_j \) unit vector, \( U \) is the slowly varying envelope, \( R \) is the transverse mode pattern, \( k \) is wavenumber, and \( \omega \) is frequency.

First consider the linear behavior. Suppose

\[
U \propto e^{i[(k^{(j)}(\omega^{(j)}) - k^{(j)}_0)z - (\omega^{(j)} - \omega^{(j)}_0)t]}. \tag{2.2}
\]
Now expand the dispersion relations about the respective carriers:

\[ k^{(j)}(\omega^{(j)}) \approx k^{(j)}_0 |_{\omega^{(j)}} + k^{(j)}_0 |_{\omega^{(j)}} (\omega^{(j)} - \omega^{(j)}_0) + \frac{1}{2} k''^{(j)} |_{\omega^{(j)}} (\omega^{(j)} - \omega^{(j)}_0)^2 \]  

(2.3)

where \( \omega^{(1)}_0 = \bar{\omega} + \Delta \omega \), and \( \omega^{(2)}_0 = \bar{\omega} - \Delta \omega \). We shall assume that \( k''^{(1)} = k''^{(2)} \equiv k'' \). A distinction would have been justifiable had we expanded to third order.

Returning from the Fourier domain, the dispersion relations imply

\[ i \frac{\partial U^{(j)}_{\text{linear}}}{\partial z} = -ik^{(j)} \frac{\partial U^{(j)}}{\partial t} + \frac{1}{2} k'' \frac{\partial^2 U^{(j)}}{\partial t^2} \]  

(2.4)

where the coefficients are implicitly evaluated at \( \omega^{(j)}_0 \).

Next consider the nonlinearity. If the fiber is idealized as possessing inversion symmetry, or the small local deviations are assumed to effectively cancel, then the second order dielectric susceptibility tensor \( \tilde{\chi}^{(2)} \) must vanish [7]. If we further assume isotropy and time-reversal symmetry (i.e. losslessness), \( \tilde{\chi}^{(3)} \) reduces to a single parameter [8]:

\[ P_{z}^{(3)} (r, t) = \tilde{\chi}^{(3)} : \mathbf{E} (r, t) \mathbf{E} (r, t) \mathbf{E} (r, t) = \epsilon_0 \chi^{(3)} E_{z,y} (\mathbf{E} \cdot \mathbf{E}). \]  

(2.5)

With harmonic time dependence, we can write

\[ E_{z,y} = \mathcal{E}_{z,y} + \mathcal{E}_{z,y}^*, \]  

(2.6)

with \( \mathcal{E}_{z,y} \propto e^{-i\omega t} \). In words, we are expressing the sinusoidally time varying electric field components as sums of complex exponential terms. Substituting in (2.5) and collecting the complex exponential terms which contribute to polarization near frequency \( \omega (\omega^{(1)} \approx \omega^{(2)}) \), we find

\[ P_z = \epsilon_0 \chi^{(3)} (3 |\mathcal{E}_z|^2 \mathcal{E}_z + 2 |\mathcal{E}_y|^2 \mathcal{E}_z + \mathcal{E}_y^2 \mathcal{E}_x^*). \]  

(2.7)

11
The expression for $P_y$ is analogous.

In the slowly varying envelope approximation (SVEA) [9],

$$P = (\text{const.}) \frac{\partial \mathcal{E}}{\partial z}. \quad (2.8)$$

From (2.5) and (2.8),

$$\frac{\partial \mathcal{E}_{\text{nonlinear}}}{\partial z} = i \frac{\chi}{3} (\mathcal{E} \cdot \mathcal{E}) \mathcal{E} \quad (2.9)$$

with $\chi$ being the Kerr coefficient [6]. Upon substitution of (2.1) into (2.9), averaging over the transverse dimensions introduces another factor $\mathcal{A} \sim \frac{1}{2}$. Then, (2.7) and (2.9) give the nonlinear variation in the slowly varying fields:

$$\frac{\partial U^{(j)}}{\partial z} = i \frac{\chi}{2} \left[ \left( |U^{(j)}|^2 + \frac{2}{3} |U^{-j}|^2 \right) U^{(j)} + \frac{1}{3} U^{-j} U^{(j)} e^{-2i(k_0^{(j)} - k_0^{-j})z} e^{2i(\omega_0^{(j)} - \omega_0^{-j})t} \right] \quad (2.10)$$

where $-j$ is the complement of $j \in \{1, 2\}$.

Together the linear (2.4) and nonlinear (2.10) behavior yield:

$$-i \left( \frac{\partial U^{(j)}}{\partial z} + k^{(j)} |_{\omega_0^{(j)}} \frac{\partial U^{(j)}}{\partial t} \right) = -\frac{1}{2} k'' \frac{\partial^2 U^{(j)}}{\partial t^2} + \frac{\chi}{2} \left[ \left( |U^{(j)}|^2 + \frac{2}{3} |U^{-j}|^2 \right) U^{(j)} + \frac{1}{3} U^{-j} U^{(j)} e^{-2i(k_0^{(j)} - k_0^{-j})z} e^{2i(\omega_0^{(j)} - \omega_0^{-j})t} \right]. \quad (2.11)$$

We can write [6]

$$k'' = -\frac{\lambda_0}{2\pi c^2} D(\lambda) \quad (2.12)$$

where $\lambda_0 = \frac{1}{2}(\lambda^{(1)} + \lambda^{(2)})$ is the arithmetic mean of the vacuum wavelengths of the two channels, $c$ is the speed of light in vacuo, and $D(\lambda)$ is dimensionless group velocity dispersion ($D(\lambda) = c\lambda D$, where $D$ is the more familiar measure of dispersion, expressed in ps/nm-km) [10]. If we approximate

$$k^{(j)}(\omega_0^{(j)}) = k^{(j)}|_{\omega_0} + \Delta \omega k''$$

12
\[ k^{(2)}|_{\omega_0^{(2)}} = k^{(2)}|_{\omega_0} - \Delta \omega k'' \]
\[ k^{(1)}|_{\omega_0^{(1)}} - k^{(2)}|_{\omega_0^{(2)}} = \left( k^{(1)} - k^{(2)} \right)|_{\omega_0} + 2\Delta \omega k'' \]

and assume
\[ \left( k^{(1)} - k^{(2)} \right)|_{\omega_0} = \frac{\left( k^{(1)}_0 - k^{(2)}_0 \right)|_{\omega_0}}{\bar{\omega}}, \]

then
\[ k^{(1)}|_{\omega_0^{(1)}} - k^{(2)}|_{\omega_0^{(2)}} = \frac{\left( k^{(1)}_0 - k^{(2)}_0 \right)|_{\omega_0}}{\bar{\omega}} + 2\Delta \omega k''. \]  

Next, we normalize [4,10] introducing
\[ u^{(j)} = \left( \frac{t_0^2 \chi}{2k''} \right)^{\frac{1}{2}} U^{(j)}, \quad z_0 = \frac{\pi^2 e^2 z_0^2}{D(\lambda) \bar{\lambda}}, \quad t_0 = 0.568 \tau \]

\[ \xi = \frac{\pi z}{2z_0}, \quad s = \frac{1}{t_0} \left( t - \frac{z}{\nu_s} \right), \quad \nu_s = \frac{2}{\left( k^{(1)}|_{\omega_0^{(1)}} + k^{(2)}|_{\omega_0^{(2)}} \right)} \]
\[ \delta_b = \frac{\left( k^{(1)} - k^{(2)} \right)|_{\omega_0}}{2|k''|} t_0 = \frac{\left( k^{(1)}_0 - k^{(2)}_0 \right)|_{\omega_0}}{2|k''|\bar{\omega}} t_0 = \frac{\pi c \Delta n}{D(\lambda) \bar{\lambda}} t_0 \]

\[ \delta_w = t_0 \Delta \omega, \quad \delta = \delta_b - \delta_w, \quad R = \frac{8\pi c}{\bar{\lambda}} t_0, \]

in which \( \tau \) is the FWHM pulsewidth, \( \Delta n \) is the difference in the indices of refraction of the two channels, \( \delta_b \) is the natural linear birefringence, and \( \delta_w \) is the effective birefringence from the difference in carrier frequencies. The normalized system of evolution equations may now be written
\[ -i \left( \frac{\partial u^{(j)}}{\partial \xi} \pm \delta \frac{\partial u^{(j)}}{\partial s} \right) = \frac{1}{2} \frac{\partial^2 u^{(j)}}{\partial s^2} + \left( |u^{(j)}|^2 + \frac{2}{3}|u^{(-j)}|^2 \right) u^{(j)} \]
\[ + \frac{1}{3} u^{(-j)} u^{(j)} e^{\frac{1}{R} \left( R \delta_b \xi - 4\delta_w s \right)} \]  

(2.15)
where the primary symbols of $\pm$ and $\mp$ are to be used for $j = 1$. The first term in the rhs accounts for group velocity dispersion, the second and third terms are the self- and cross-phase modulation, and the final term is the so-called coherence term.

This completes the derivation of the coupled cubic nonlinear Schrödinger equations for linear polarizations and with linear birefringence. In the remainder of this work, we shall be concerned with large birefringence, i.e. the pulse envelopes will slide through each other during the interaction. The phase velocity slip is sufficiently large that the coherence terms are highly oscillatory and affect the pulse behavior insignificantly. This issue will be examined further in chapter 4.
Chapter 3

Perturbation Theory

Analytic solutions for pulse collisions governed by the CNLS have evaded researchers to date. This holds true even in the absence of the coherence terms and for $\delta=0$. Christodoulides and Joseph [11] have recently contributed to our understanding of solutions to the CNLS. Although they did not address the issue of collisional robustness, they claim to have discovered soliton solutions with $\delta=0$, using the method of Kay and Moses [12]. These solutions are stationary, but it remains unclear whether or not they are actually solitons. One source of doubt is the fact that solitons are associated with completely integrable Hamiltonian systems, solvable by the inverse scattering transform (IST) method, and no one to date, to the authors awareness, has demonstrated that the CNLS possesses a “Lax pair” [13], which is required for the IST.

There is a system similar to the CNLS (in the absence of coherence terms and with $\delta=0$), which does exhibit a Lax pair. It can be used to model self-focusing of circularly polarized EM waves in two dimensions and was investigated by Manakov [14]. We can generalize Manakov’s system slightly and show that the
following set of equations exhibits a Lax pair and soliton solutions:

\[-i \frac{\partial u^{(j)}}{\partial \xi} = \frac{1}{2} \frac{\partial^2 u^{(j)}}{\partial s^2} + \left( a|u^{(1)}|^2 + b|u^{(2)}|^2 \right) u^{(j)}, \quad (3.1)\]

\(j \in \{1, 2\} \).

Perhaps the key distinction between this generalized Manakov problem and the CNLS for linear birefringence and linear polarization is the fact in the former system, both channels are under the influence of the same potential well. It is well known that if one considers circular polarization in lieu of linear, the \(\frac{2}{3}\) coupling coefficient increases to 2, and there are no coherence terms. Nevertheless, the potential wells are again different in the two channels. If we could somehow obtain a coupling coefficient of unity, without introducing other terms, we would have the Manakov system with \(a=b=1\).

Is it possible to choose some elliptic polarization for which the potential wells for the two channels are identical, in which case the system would be integrable? The answer is that we can equate the coefficients of the two nonlinear terms of interest but not without introducing some horrible additional terms. In fact, there are not two nonlinear terms, as in the Manakov system or for circular polarization, nor are there three as for linear polarization, rather there are six nonlinear terms! The potential wells are not in fact the same, even in regimes in which we can neglect coherence type terms. More specifically, if we name our orthogonal elliptically polarized fields \(e\) and \(f\), then the equation for the evolution of \(e\) contains terms of the form \(|e|^2 e\), \(|e|^2 f\), \(e^* f^2\), \(e^2 f^*\), \(|f|^2 e\), and \(|f|^2 f\) (to determine which terms appear in the \(f\) equation, simply interchange all \(e\)'s and \(f\)'s in this list).

The point of this rather lengthy introduction is to motivate the development of a perturbation theory. It is well-known that a NLS soliton of Nth order is com-
pletely characterized by $4N$ real parameters [2]. To each order there corresponds a complex pole and residue, so named for the role of these parameters in inverse scattering formalism. The perturbation theory developed below is used to determine the shifts in the poles and residues resulting from either (a) the addition of a small modification to a solitonic initial profile, or (b) small deviations from NLS behavior in the medium. In this paper, the theory will be applied to the latter class of problems.

In addition, only perturbations of $N=1$ solitons will be considered here. For clarity, $N=1$ will be assumed in the derivations. Generalization is straightforward, but the application to higher order problems can be an algebraic quagmire.

3.1 Derivation

Let us begin with the unadorned cubic NLS

$$-i\frac{\partial u}{\partial \xi} = \frac{1}{2} \frac{\partial^2 u}{\partial s^2} + |u|^2 u. \quad (3.2)$$

Taking $u$ to be a solution, we introduce a perturbation $v$, $u \rightarrow u + v$ which satisfies

$$-i\frac{\partial (u + v)}{\partial \xi} = \frac{1}{2} \frac{\partial^2 (u + v)}{\partial s^2} + (u + v)^2 (u + v)^*. \quad (3.3)$$

Using (3.2) and linearizing in $v$ and $v^*$, we obtain

$$-i\frac{\partial v}{\partial \xi} = \frac{1}{2} \frac{\partial^2 v}{\partial s^2} + 2|u|^2 v + u^2 v^*. \quad (3.4)$$

We observe that

$$\frac{\partial}{\partial \xi} \int vv^* \, ds \neq 0 \quad (3.5)$$

17
or \(v\) does not obey a self energy conservation law. We shall say that the solutions to (3.4) are not self-adjoint with respect to our inner product. An adjoint system can be constructed such that

\[
\frac{\partial}{\partial \xi} \text{Re} \int v \bar{u} \, ds = 0
\]

(3.6)

where the overbar indicates adjoint. We can make the integrand pure imaginary by simply choosing \(\bar{v} = \pm iv^*\). The conservation law is then valid for all complex \(v\).

Let us select \(\bar{v} = iv^*\). Substitution in the linearized equation (3.4) provides our adjoint system:

\[
\frac{i}{2} \frac{\partial \bar{v}}{\partial \xi} = \frac{1}{2} \frac{\partial^2 \bar{v}}{\partial s^2} + 2|u|^2 \bar{v} - u^* v^*.
\]

(3.7)

It is at this point that we introduce the pole and residue. The (small) perturbation can be expanded as a sum of changes to pole and residue plus a dispersive (radiative, nonsoliton) component \(\delta\):

\[
v = \sum_{j=1}^{4} \frac{\partial u}{\partial p_j} \Delta p_j + \delta,
\]

(3.8)

where the \(p_j\) denote the real and imaginary parts of the pole (\(\xi_1\) and \(\eta_1\)) and the magnitude and phase of the residue (\(|c_1|\) and \(\phi_1\)).

Below we shall introduce overlap integrals in \(s\) of \(\overline{\partial u/\partial p_k}\) with eq. (3.8), taking the real part. To see that the overlap with \(\delta\) vanishes, first note that \(\overline{\partial u/\partial p_k}\) is associated with the soliton and therefore local. Since \(\delta\) disperses as \(\xi \to \infty\), the overlap \(\to 0\) as \(\xi \to \infty\). Next observe that since each of the five terms (and any linear combination thereof) in the rhs of (3.8) satisfies the perturbed NLS (3.4), each term also satisfies the conservation law (3.6). In order for \(\delta\), \(\overline{\partial u/\partial p_k}\), and
\( \dot{v} + \partial u / \partial p_k \) to simultaneously satisfy (3.6), we require

\[
\frac{\partial}{\partial \xi} \text{Re} \int \dot{v} \frac{\bar{u}}{\partial p_k} \, ds = 0. \tag{3.9}
\]

In conclusion, the overlap integral is independent of \( \xi \), vanishes as \( \xi \to \infty \), and therefore vanishes for all \( \xi \).

The overlap with (3.8) is

\[
\text{Re} \int v \frac{\partial u}{\partial p_k} \, ds = \sum_{j=1}^{4} \Delta p_j \text{Re} \int \frac{\partial u}{\partial p_j} \frac{\bar{u}}{\partial p_k} \, ds \tag{3.10}
\]

which may be solved for the \( \Delta p_j \).

### 3.2 Application - Simplified, Analytic Theory

One would use expression (3.10) to find parameter shifts upon perturbing an initial soliton. In this work, we are interested in the reaction of a soliton to a medium perturbation, for which we must perform the analysis incrementally. We can equate any extra terms added to the rhs of the NLS (3.2) to \(-i \frac{\partial u}{\partial \xi}\).

For our specific case, the coupled NLS in the absence of coherence terms, first define

\[
\sigma \equiv 2 \delta
\]

which is the total slip or initial “relative speed” of the two pulses along the \( s \)-axis.

With a simple change of coordinates, we can transform the CNLS, sans coherence terms, to the form

\[
- i \frac{\partial u}{\partial \xi} = \frac{1}{2} \frac{\partial^2 u}{\partial s^2} + \left( |u|^2 + \frac{2}{3} |u^{(2)}|^2 \right) u. \tag{3.11}
\]

\[
- i \left( \frac{\partial u^{(2)}}{\partial \xi} - \sigma \frac{\partial u^{(2)}}{\partial s} \right) = \frac{1}{2} \frac{\partial^2 u^{(2)}}{\partial s^2} + \left( |u^{(2)}|^2 + \frac{2}{3} |u|^2 \right) u^{(2)}. \tag{3.12}
\]
where the superscript has been dropped from \( u^{(1)} \). We shall treat the coupling from channel 2 as a perturbation upon the behavior of channel 1:

\[
\dot{v} = \frac{2}{3} |u^{(2)}|^2 u. \tag{3.13}
\]

In the manner of Newton, a superior dot denotes differentiation. Here, it is with respect to \( \xi \).

In channel 1, we shall choose an \( N = 1 \) soliton as the initial profile. The most general expression for the \( N = 1 \) is [15]

\[
u(s, \xi) = \frac{-4\eta_1 e^{-2i[(\xi_1 s + (\xi_1^2 - \eta_1^2)\xi + \xi_1)]}}{\left( \frac{2^n}{|c_1|} e^{2n_1(s + 2\xi_1\xi)} + \frac{|c_1|}{2n_1} e^{-2n_1(s + 2\xi_1\xi)} \right)}. \tag{3.14}\]

From this expression, one can see that \( \eta_1 \) determines the amplitude and width of the soliton, while \( \eta_1 \) and \( |c_1| \) together determine the position of the soliton along the \( s \)-axis. For fixed \( \eta_1 \), the center of the soliton approaches \( s = \infty \) as \( |c_1| \to 0 \) and approaches \( s = \infty \) as \( |c_1| \to \infty \). We shall select the \( N = 1 \) of unit amplitude, centered at the \( s \)-origin. The appropriate soliton parameters are \( \xi_1 = 0, \eta_1 = \frac{1}{2}, \) and \( c_1 = 1 \) (recall \( c_1 = |c_1|e^{i\phi_1} \)). Upon differentiating (3.14) with respect to each parameter and evaluating for the above choice of pole and residue,

\[
\frac{\partial u}{\partial \xi_1} = 2u(\imath s - \xi \tanh s) \tag{3.15}
\]

\[
\frac{\partial u}{\partial \eta_1} = 2u(1 - \imath \xi - \tanh s - s \tanh s) \tag{3.16}
\]

\[
\frac{\partial u}{\partial |c_1|} = u \tanh s \tag{3.17}
\]

\[
\frac{\partial u}{\partial \phi_1} = \imath u \tag{3.18}
\]

The system of four equations (3.10) now reduces to the following:

\[
-2\Delta \xi_1 = \text{Re} \int \left( v \frac{\partial u}{\partial |c_1|} \right) ds \tag{3.19}
\]
\[ 2\Delta \eta_1 = \text{Re} \int \left( v \frac{\partial u}{\partial \phi_1} \right) ds \]  

(3.20)

\[ -4\Delta \eta_1 + 2\Delta |c_1| = \text{Re} \int \left( v \frac{\partial u}{\partial \xi_1} \right) ds \]  

(3.21)

\[ 4\Delta \xi_1 - 2\Delta \phi_1 = \text{Re} \int \left( v \frac{\partial u}{\partial \eta_1} \right) ds \]  

(3.22)

where we are assuming that \( u \) and its parametric derivatives remain essentially unchanged during the interaction. This assumption breaks down as the slip approaches zero (strong coupling).

The choice \( \xi_1 = 0, \eta_1 = \frac{1}{2} \), and \( c_1 = 1 \) simplifies (3.14) to

\[ u = e^{\frac{i}{2} \text{sech} s}, \]  

(3.23)

which is our channel 1 initial condition. In channel 2, we choose \( \xi_2 = 0, c_2 = 2\eta_2 \) and incorporate the group velocity slip into the envelope argument:

\[ u^{(2)} = 2\eta_2 e^{i2\eta_2 \xi} \text{sech} 2\eta_2 \left(s - \sigma \xi\right). \]  

(3.24)

We can substitute in (3.13)

\[ \dot{v} = \frac{2}{3} |u^{(s)}|^2 u = \frac{8\eta_2^2}{3} e^{i\eta_2 \text{sech}^2} 2\eta_2 \left(s - \sigma \xi\right) \text{sech} s. \]  

(3.25)

Note however that we are attempting to solve (3.19)- (3.22) which are expressed in terms of \( v \), not \( \dot{v} \). To lowest order, upon \( \xi \)-differentiating these equations,

\[ -2\Delta \dot{\xi}_1 = \text{Re} \int \left( \dot{v} \frac{\partial u}{\partial |c_1|} \right) ds \]  

\[ 2\Delta \dot{\eta}_1 = \text{Re} \int \left( \dot{v} \frac{\partial u}{\partial \phi_1} \right) ds \]  

\[ -4\Delta \dot{\eta}_1 + 2\Delta \dot{|c_1|} = \text{Re} \int \left( \dot{v} \frac{\partial u}{\partial \xi_1} \right) ds \]  

\[ 4\Delta \dot{\xi}_1 - 2\Delta \dot{\phi}_1 = \text{Re} \int \left( \dot{v} \frac{\partial u}{\partial \eta_1} \right) ds \]
The parameter shifts may now be evaluated as double integrals. The evaluation is greatly facilitated by performing the $\xi$-integration prior to the $s$-integration.

From the antisymmetry of the $s$-integrand, $\Delta \xi = 0$. For $\Delta \eta$, the $s$-integrand is pure imaginary, and the expression vanishes. The remaining two parameter shifts are nonvanishing. Summarizing the results,

$$\begin{align*}
\Delta \xi_1 &= 0 \\
\Delta \eta_1 &= 0 \\
\Delta |c_1| &= \frac{8 \eta_2}{3 \sigma^2} \text{sgn}(\sigma) \\
\Delta \phi_1 &= \frac{8 \eta_2}{3 |\sigma|}.
\end{align*}$$

We shall discuss these results below. In the next chapter we will consider numerical simulations of the coupled NLS's and compare the results of this section with simulations.
Chapter 4

Numerical Analysis

A superior method of determining collision-induced pulse modifications is to perform dynamic computer simulations. In this chapter, we compare the results obtained from the simple perturbation theory with solitary wave collision simulations, particularly for the case of equal pulses in the two channels. While this comparison is the main thrust of the chapter, various related simulations will be discussed as well.

In order to compare the two approaches, perturbative and numerical, it is necessary to extract from the simulations the "soliton parameters" of the scattered solitary waves. The very fact that the pulses are not true solitons makes this a somewhat artificial exercise, but for sufficiently large slip, or walkoff, the approximation is reasonable.
4.1 Software

The simulation code employed in this research was written by three MIT students. Sumanth Kaushik, an undergraduate at the time, wrote the original version in 1986. At this stage, the coupled NLS’s with no coherence terms, nor any other perturbative terms, and with pulses of the same frequency, could be simulated. Output consisted of periodically generated datafiles with amplitude and phase information, which could then be converted to graphical format.

Randa Seif Hartemann, a visiting scholar from France, improved the output of the code, making phase information more discernible, mainly by subtracting a phase reference. This reference is simply the dynamical phase.

Through 1987 and 1988, I have substantially modified the code. Considerable effort has gone into code optimization, including partial vectorization for supercomputer runs. The code has also been changed to take advantage of certain symmetries which exist in certain cases, such as the zero slip case. Coherence terms, pulses of different frequencies, and self-Raman frequency shift are now available and easily incorporated. The user is provided greater flexibility with initial conditions and output. The code can automatically track pulse area, energy, “soliton parameters,” phase and center shift, and provide pulse Fourier transforms. The code in its form as of November 15, 1988 appears in Appendix A.

At present, the IBM 4381 Model 14 mainframe of MIT’s Research Laboratory of Electronics and the original IBM 3090 600E of the Cornell National Supercomputing Facility, are the machines with which simulations are run. CPU requirements on the RLE machine are on the order of 15 minutes for the simple case of no coherence or Raman terms, but without taking advantage of any symmetries.
A 7-8-fold speedup is achieved on the supercomputer.

The code is written in FORTRAN with double precision variables (eight bit real, sixteen bit complex) for which both machines are optimized. A personal computer would be unacceptably slow because of the large number of simulations which must be run. In addition, the code calls subroutines from the Engineering and Scientific Subroutine Library (ESSL) for discrete Fourier transforms and for locating maximum elements of arrays, making comparable software a requirement for PC implementation.

A copy of the present code and its required input file (with typical parameters selected) is found in the appendices. The input file has been commented extensively, with the function of each input parameter explained. The simulation program is also commented, but deserves additional explanation here.

The notation in the code differs from that in this document. Most notably, our shifting time coordinate \( s \) is named \( x \) in the code, and the spatial coordinate \( \xi \) is called \( t \). These variable names had been selected to correspond to those typically used in the 1–D Schrödinger equation as applied in quantum mechanics. However, the switched roles of space and time make this notation somewhat confusing.

The method of numerical simulation is as follows: At a given point in space, the temporal profile is obtained via an explicit sixth order finite difference scheme. We evolve the system spatially with a leapfrog scheme, which requires the first two temporal profiles. The first profile is provided by the user, and the second is obtained from the first with explicit finite difference.

Kaushik provided three ways to handle temporal boundary conditions, as described in the “boundary conditions” section in the program. I have consistently
used "hard" B.C.'s. That is, the field is set to zero outside our finite temporal domain, or "window."

This program has the unusual feature of separate coordinate systems for the two channels. Mathematically, a change of variables is introduced which eliminates the linear first $s$-derivative term in the first channel. Subsequently, a different coordinate system, call it $s'$, is introduced for the second channel in order to eliminate its linear first $s$-derivative term. In other words, each coordinate system travels at the group speed at which an $N = 1$ soliton would propagate in the absence of the other channel, making the pulses stationary in their respective frames.

It is a relatively straightforward matter to incorporate the relative motion of these coordinate systems into the code. It is not so straightforward, however, to handle the coupling terms. Kaushik handled this by interpolating the fields between frames at each spatial increment.

Other features of the code were mentioned at the beginning of the chapter. Another entire chapter could be devoted to optimization, but this is rather tangential. It is however important that I describe the "soliton parameter tracking."

## 4.2 Pole and residue extraction

For the simple reason that our solitary waves are not solitons, it is somewhat artificial to associate poles and residues with the scattered pulses. This fact implies that a scheme for determining said parameters is not unique. It is the purpose of this section to describe the method which I selected.

The portion of the main program to which I will be referring is titled "Energy,
area output + soliton parameter extraction." This section of code is accessed periodically. It is within the space loop, following the time loop. This means that the temporal profile for this spatial increment will have been determined just prior to the tracking section of code.

Next we describe the function of this portion of code. First, an ESSL routine is called to determine the location of the maximum field amplitude in channel 1. In the code, this maximum amplitude is named u0, and its \( x \)-location is named s0. Also of use are the nearest neighbor array elements. To the "left" on the \( x \)-axis, the amplitude is named u9, at \( x = s9 \). The righthand neighbor is likewise u1 at s1. With this information, we interpolate to estimate the actual pulse center location (named center) and maximum amplitude (peak). I have written both quadratic and hyperbolic secant fit routines. The former is less computation intensive and appears in the version of the program which appears in this document. The derivation of the fitting routines is quite straightforward and will not be included here.

In order to determine the artificial poles and residues, we must compare our numerically determined temporal profile with the general \( N = 1 \) soliton. Recall from the perturbation theory chapter that

\[
    u(s, \xi) = \frac{-4 \eta_1 e^{-2i[\xi_1 s + (\xi_1^2 - \eta_1^2)\xi + \frac{\eta_1^2}{2}]} \left( \frac{2 \eta_1}{|c_1|} e^{2\eta_1(s + 2\xi_1 \xi)} + \frac{|c_1|}{2 \eta_1} e^{-2\eta_1(s + 2\xi_1 \xi)} \right)}{\left( \frac{2 \eta_1}{|c_1|} e^{2\eta_1(s + 2\xi_1 \xi)} + \frac{|c_1|}{2 \eta_1} e^{-2\eta_1(s + 2\xi_1 \xi)} \right)}. \tag{4.1}
\]

which can be written

\[
    u(s, \xi) = -2 \eta_1 e^{2i[\xi_1 s + (\xi_1^2 - \eta_1^2)\xi + \frac{\eta_1^2}{2}]} \text{sech} 2 \eta_1 \left[ (s + 2 \xi_1 \xi) - s_{\text{ctr}} \right], \tag{4.2}
\]

where

\[
    s_{\text{ctr}} \equiv - \frac{1}{2 \eta_1} \ln \left( \frac{2 \eta_1}{|c_1|} \right). \tag{4.3}
\]
I first determine \( \eta_1 \). This parameter establishes both the amplitude and width of the pulse. From the fitting routine, we know the maximum amplitude, \( \text{peak} \). Although a portion of code is present which finds the full pulse width at half maximum intensity (FWHM), \( \text{sfwhm} \) in the code, it is not used in parameter extraction; the value is merely dumped to an output file. It is useful to compare \( \text{peak} \) with \( \text{sfwhm} \) as a measure of deviation from soliton character. At present, \( \eta_1 \) is estimated as half of \( \text{peak} \).

The second parameter which I determine is \( \xi_1 \). From (4.2), we see that \( 2\xi_1 \) is the slope of the phase. I simply take two nearby values of pulse phase and divide by the separation in \( x \). This could definitely be improved. In particular, the phase is not always linear across an actual scattered wave. Preferable would be a least squares fit for several array elements across the pulse. There is no least squares routine in Release 1 of ESSL, and for the range of slip values for which the perturbation theory provides reasonable results, the simple approximation is adequate.

Unfortunately, to determine the parameters of greatest interest, namely the magnitude and phase of the residue, we must rely upon the estimates for \( \xi_1 \) and \( \eta_1 \). For the residue magnitude, we use the fact that at the soliton peak,

\[
\text{center} + 2\xi_1 \xi - s_{\text{ctr}} = 0.
\]

Then from the definition of \( s_{\text{ctr}} \) (4.3), we find

\[
|c_1| = 2\eta_1 e^{2\eta_1 (\text{center} + 2\xi_1 \xi)}.
\]

Finally, \( \phi_1 \) is determined directly from the expression for the residue phase
(recall that pulse phase and residue phase are distinct quantities):

\[ \phi_1 = -\left( \theta + 2|\xi_1 \text{ center}| + (\xi_1^2 - \eta_1^2)\xi_1 \right) \]

where \( \theta \) is the numerical value of phase at the pulse center.

At this point note that in describing the effects of scattering upon a solitary wave, we are more interested in the center of mass shift than in the change in the magnitude of the residue. In fact, in the regime in which the coherence terms may be neglected, the symmetry of the coupled NLS's requires that the center shift be the same regardless of which channel is the fast axis. In terms of our parameters, the magnitude of the center shift should be the same regardless of the sign of \( \sigma \). As we constructed our perturbation theory, however, it is \( \Delta|c_1| \) whose absolute value is conserved under channel interchange. The center shift is related to the logarithm of \( |c_1| \) and in general the perturbative approach fails to reflect the above symmetry. Of course, this is very dramatic when \( \Delta|c_1| \) becomes of the same order as \( |c_1| \).

### 4.3 Examples of collision simulations

In this section, we shall look at the results of several simulations. Although motivated by switching, several "interesting" but impractical results will be presented in addition to those in more practical regimes.

In order to relate the simulation results to reality, it is useful to consider some typical numerical values. Consider a unit amplitude \( N = 1 \) soliton solution to the NLS (3.23). In \( \xi \), its period is \( 4\pi \). In real world units, this period is

\[ \text{soliton period} = \frac{8(0.568\pi cr)^2}{D(\bar{\lambda})/\bar{\lambda}} \quad (4.4) \]
If we choose $D(\bar{\lambda}) = 6.5 \times 10^{-3}$ and $\bar{\lambda} = 1.55 \mu m$, then the period $= (2.84 \times 10^{25})\tau^3$ meters, $\tau$ being the pulsewidth in meters. For a 500 fs pulse, the period is a mere 7.1 m. If the pulsewidth exceeds 5.94 ps, the period is greater than 1 km.

Shorter pulses appear to be more desirable, but one must bear in mind that the soliton self-frequency shift (SSFS), proportional to $\tau^{-4}$, may be significant for subpicosecond pulses, if the propagation distances are sufficiently long [16]. Although our simulation software permits the user to include SSFS, this effect was not studied in this work.

It is important to note that in all collisions for which we have not omitted the coherence terms, the signal is polarized along the slow axis. This is to lessen the effects upon the signal of the polarization instability. This instability has been demonstrated numerically for linearly polarized solitary waves in linearly birefringent fibers [17].

Let us first consider the simplest case. Initially place two unity amplitude NLS $N = 1$'s ($\eta_1 = \eta_2 = 0.5$) so that the separation is sufficient for the coupling to be negligible. Allow them to collide and find the soliton parameters. Vary the slip from large values (fast collisions) to small values.

Equal pulses are convenient if we neglect coherence terms. Because of the symmetry of the equations, knowledge of the channel 1 profile and phase plus the center of mass of the two pulses provides complete information about channel 2: simply reflect the $u$-waveform about the center of mass. Note that the appropriate phase reference has been subtracted from each channel.

In figure 4.1, we see the initial $u$ pulse and the same pulse long after collision with an equal $v$ pulse. The slip $\sigma = 5$. In each case, the darker curve represents
the magnitude of the pulse envelope, while the lighter curve represents the phase. Phase is measured in $\pi$ radian increments, and plotted modulo 2. The initial pulse phase is chosen to be zero. Thus, in (a), the lighter curve is not evident. Phase in (b) may be somewhat confusing. Moving left to right across the pulse, the phase increases from zero to roughly $0.1\pi$ radians. To the right of the pulse, the phase decreases, and in fact dips below zero. The tall vertical segments are artifacts of plotting phase modulo 2. In the plots, all phase data have been connected, resulting in artificial discontinuities. There is a very minor change in the envelope following the collision, too minor to be visible in the figure. The pulse has developed a small tail on the right. The coupling is weak for $\sigma = 5$.

The abstruse collection of numbers and letters at the top of each plot actually contains a great deal of information. The interested reader can ascertain the meaning of each portion by locating the section of subroutine FILHND in the simulation code in which this information is printed to the data files. The extensively commented input data file identifies most of the variables. A few of the items deserve explanation here.

At the upper left of the headings in figure 4.1 is the letter ‘U’. This indicates that the plot shows the channel 1 magnitude and phase. A ‘V’ would identify channel 2, while a ‘UFT’ would indicate a plot of the magnitude of the (complex) Fourier transform of the channel 1 pulse. The code also generates plots of various parameters as functions of $\xi$. These are also identified in the upper left corner of the heading.

The second line of the enigmatic heading begins with ‘s’ followed by a number. This number is the slip $\sigma$. The number following ‘sb’ (or ‘b’ in the most recent
Figure 4.1: High speed collision, equal pulses, $\sigma = 5$: (a) initial, (b) post-collision.
plots) is the slip due to natural birefringence alone. All simulations considered in this work involve pulses with equal carrier frequencies, which implies that the values following ‘s’ and ‘sb’ should be identical – do not be mislead by the fact that the former is represented as mantissa and exponent while the latter is floating point. In some plots, the ‘sb’ figure may be a sequence of *’s, which merely indicates that the value could not be represented as specified by the FORMAT statement in subroutine FILHND.

On the third line, we find ‘vctr’ followed by a very useful number. This is the predicted location of the center of the v pulse, based upon its initial position, the slip, and the distance over which the system has propagated. The next item on the third line, ‘t,’ is this “elapsed” propagation distance. Of course, in a collision with significant interaction, the pulses undergo center shifts, invalidating this simple estimate of the v pulse center location. The value of ‘vctr’ is still useful, however, because equal pulses undergo equal and opposite center shifts. This implies that halving the value of ‘vctr’ provides the correct location of the center of mass of the entire system. The final item on the third line will be either ‘c0’ or ‘c1’ depending (respectively) upon whether or not the coherence terms were neglected in the simulation.

In this and most simulations which will be discussed, the coherence terms were neglected. In fact, I did simulate the same conditions without neglecting the coherence terms, and the results were indistinguishable from those in the figure. One expects the coherence terms to be significant only when both $R\delta$ and $\sigma$ are small. These quantities are small if the pulses overlap for a significant distance, i.e. $\sigma < 1$, and if the coherence terms do not oscillate very rapidly, i.e. $R\delta$ is of
order unity.

Let us return to the discussion of the simulation which provided figure 4.1. From the (a) heading, ‘vctr’ is initially $-20.0$. This tells us that the initial second channel pulse, call it the $v$ pulse, which is identical to the first channel, or $u$ pulse, is located at $s = -20.0$. In (b), the $v$ pulse has moved to the right, to $s = 26.0$. From the symmetry of the equations and I.C.’s, we know that the $v$ magnitude and phase corresponding to (b) can be obtained by reflecting the plot about the center of mass $s = 26.0/2 = 13.0$.

Figure 4.2 shows results of a $\sigma=2$ collision. This is a fairly slow collision, as we can see from the scattering products. In (b), note the small bump, centered at $s = 28.0$ (c.f. ‘vctr’). A portion of each pulse, a “shadow,” has been “captured” by the other pulse and copropagates with it. The motion of the “shadow,” at a speed different from the channel group speed is evident from the slope, or “chirp” of its phase.

Perhaps it is instructive to further describe the phase plot. First note that for very small amplitudes, phase is irrelevant. In fact the output subroutine of the simulation code compares the amplitude at a given $s$ with a threshold ($10^{-3}$ in the version in Appendix A), and prints zero phase to the output file if the amplitude is below this value. This explains the first discontinuity seen on the left. The next discontinuity is a typical modulo 2 feature. Just to the left of the pulse, the phase drops below zero, then rises to a relatively flat value ($\sim 0.2\pi$ radians). To the right, it drops below zero again. Subsequently, the phase increases rapidly, indicating that the small amplitude field is spreading away from the main pulse.

The appearance of a shadow is of course a significant departure from the ideal
Figure 4.2: Relatively slow collision, equal pulses, $\sigma=2$ : (a) initial, (b) post-collision.
NLS soliton behavior upon which our perturbative approach was founded. In fact, as we shall see, the predictions of the perturbation theory are unreliable for smaller slip values ($\sigma < 2$).

One would expect that further decreasing the slip would give rise to stronger capture. This is in fact what I have found. However, in addition to the capture, frequency shifts of the main pulses are induced. The slower pulse is accelerated during the collision, and the faster pulse is decelerated. A rather extreme case is shown in figure 4.3 for $\sigma = 0.625$. The reader is again reminded that the $v$ profile can be obtained by reflecting the $u$ profile about $vctr/2$, so there are two pulses. Additional structure aside from the main peaks can be seen: broad, small amplitude, oscillatory structures develop in front of and behind the main pulses. This is perhaps largely radiative, indicating that we have strayed from soliton states of the CNLS, should they exist. This simulation was run for roughly four times the distance required for the $v$ pulse to reach the $u$ pulse, given the initial temporal separation (i.e., $4 \times$ (initial separation in $s$) / $\sigma$).

What happens if we further decrease $\sigma$? A mere reduction to $\sigma = 0.6$ leads to yet more interesting behavior. Figures 4.4 through 4.7 show a few samples of the evolution of the scattered pulses, revealing the possible existence of a "breather," or oscillatory bound state. The scattering is inelastic. In fact, Randa Seif Hartemann first observed evidence of a breather in 1987 in a simulation with $\sigma = 0.5$. The latter stages of my simulation revealed small peaks emerging to the front and back of the main pulses, but it is unclear whether these are physical, or if they are byproducts of hard boundary conditions and/or numerical roundoff error. Reflections of the broad, small amplitude pedestal restrict the useful length of any
Figure 4.3: Slow collision, equal pulses, $\sigma=0.625$: (a) initial, (b) post-collision.
such simulation. Also, very sharp derivatives arise in the breather. Regardless of subsequent behavior, however, there is no denying the oscillatory pulse energy exchange.

Three related simulations are certainly worth mentioning, although accompanying figures have not been provided. Under conditions identical to those in the $\sigma=0.6$ simulation discussed above, variants of the unit amplitude, unit width sech's were used. If one of the pulses is a unit amplitude, unit width sech and the other is subsolitonic, with unit width, but with amplitude 0.9, then no bound state oscillation is observed. The subsolitonic pulse has not dispersed very significantly prior to the collision, and the collisional energy exchange appears to give rise to two stable structures, in spite of the fact that one of the two initial pulses in isolation would have completely dispersed.

What is more surprising is that if we use two of the 0.9 amplitude pulses as our initial conditions, we again see the creation of what appear to be two stable structures - pulses with energy distributed roughly 2:1 between the two channels, not unlike those seen in the $\sigma = 0.625$ case above. Bear in mind that the stability has not been tested with any rigor, for the author was merely looking for breathers. Breathers were not observed.

Finally, if we choose a 1.1 amplitude, 1.1 width sech and a 0.9 amplitude, 0.9 width sech (both NLS $N = 1$ solitons), we find capture but no oscillation. At least for $\sigma=0.6$, the breather appears to be a rather tenuous structure. Smaller slip values have not yet been thoroughly investigated.
Figure 4.4: Slow collision, equal pulses, $\sigma=0.6$ : (a) initial, (b) $t = 33.75$. 
Figure 4.5: Slow collision, equal pulses, $\sigma=0.6$ : (a) $t = 51.75$ , (b) $t = 63.00$ .
Figure 4.6: Slow collision, equal pulses, $\sigma=0.6$: (a) $t=67.50$, (b) $t=69.75$. 
Figure 4.7: Slow collision, equal pulses, $\sigma=0.6$: (a) $t = 81.00$, (b) $t = 87.75$. 
4.4 Comparison with perturbation theory

Of course, the cases discussed above for which \( \sigma < 2 \) are totally impractical for our proposed all-optical switching technique. The reader was forewarned of this, however. Other simulations, not discussed above because they reveal nothing new, were run for \( \sigma > 2 \) as well. The comparison with the simple perturbation theory is presented graphically in figure 4.8.

We see that the perturbation theory provides reasonable parameter estimates for slip values in excess of 2. For smaller slip, the pulse distortion is in fact beyond that which would be acceptable for switching applications.

It is clear that a \( \pi \)-phaseshift could not be achieved from a single collision of equal unit amplitude linearly polarized NLS \( N = 1 \) pulses in linearly birefringent polarization maintaining fiber. One solution is to collide the signal pulse with several \( N = 1 \)'s in succession, and thereby accumulate sufficient phase. This is inelegant, is more difficult to implement, and would require far greater fiber lengths.

4.5 Alternate approaches to switching

From the perturbative phaseshift expression at the end of Chapter 3 we might try increasing the value of \( \eta_2 \); that is, use a taller, narrower pulse in the second channel. If we could collide these pulses over the same range of slip, we could achieve greater phaseshifts. Physically, the cross phase modulation is increased because of the larger intensity at the peak of the control pulse. Of course, the stronger coupling also leads to greater signal pulse distortion. Our attempt to
Figure 4.8: Comparison of predictions of simple perturbation analysis with collision simulations. Initial pulses unit amplitude, unit width sech's. (a) $\Delta \phi_1$, (b) $\Delta |c_1|$. 
increase $\eta_2/\sigma$ fails because an increase in $\eta_2$ requires at least a commensurate increase in $\sigma$.

How can we modulate the phase of the signal pulse without disturbing the envelope? One idea is to decrease the amplitude of the control pulse and take advantage of the reduced signal distortion at smaller slip. In fact, we can operate in a regime in which the natural birefringence is sufficiently small that the pulse envelopes essentially copropagate for the duration of the interaction ($\sigma \to 0$).

The key is, considering equation (2.15), to choose $\delta$ small while keeping $R\delta_b \gg 1$. In words, we require small group velocity slip and large phase velocity slip. Mathematically, an easy way to achieve this is to choose $\delta_b = \delta_\omega$; i.e., use pulses of different frequencies. In this case we would actually have zero group velocity slip, with several available parameters to vary if necessary in order to force the coherence terms to oscillate rapidly, making them negligible. From a practical standpoint, however, two-color schemes are not so attractive. At present, two lasers would be required, and a two-color logic scheme would have to be implemented.

Without resorting to pulses of different carrier frequencies, we can still select our parameters such that $\delta$ is small and $R\delta_b \gg 1$. Let us consider the following situation, many of the values for which are taken from Menyuk [6]. First, assume that the walkoff is roughly $10^{-7}$, typical of the fluctuations in nonpolarization preserving fiber:

$$\frac{1}{k^{(i)}} - \frac{1}{k^{(\bar{i})}} \approx 10^{-7} \frac{c}{n}.$$  

Take $D = 6.5 \times 10^{-3}$, $\lambda = 1.55\mu m$ and a pulsewidth of 5 ps. This gives $\delta \approx 0.04$
and \( R\delta = R\delta_b \approx 200 \). Note that

\[
R\delta \propto \tau^2
\]

so for 500 fs pulses, \( R\delta \) is reduced to roughly 2, which is getting uncomfortably small. One could improve \( R\delta \) through changing the wavelength or dispersion in the fiber:

\[
R\delta \propto D^{-1} \lambda^{-2}.
\]

A fiber with anomalous dispersion (and low loss) in the visible would permit the use of pulses 2 - 4 times shorter than at 1.55\( \mu \)m. Before we go searching for exotic fibers, however, consider the fact that it is beyond our present capabilities to produce a fiber with consistent \( 10^{-7} \) walkoff over any significant distance. We have to work with fiber whose walkoff fluctuates. This may make copropagation impractical.

Perhaps we need not concern ourselves with phase velocity slip. Suppose we operate with circularly polarized signal and control. The governing equations in this case lack coherence terms. In addition, the coupling coefficient is three times larger than for linear polarizations. One of my next endeavors, following the completion of this thesis, will be to prove whether or not a polarization instability exists for circularly polarized solitons. I intend to investigate the cross phase modulation in this environment, looking for improved switching. In particular, since coherence terms were the bane of copropagation, and copropagation provided good phaseshifting without excessive signal distortion, copropagation of circularly polarized pulses looks promising.

Out of curiosity, I tried something analogous to the so-called "vector solitons" of Christodoulides and Joseph [11]. In particular, I used a sech signal (NLS \( N = \)
1), with a sech-tanh control pulse. I found that my antisymmetric control pulse increased its area (my I.C.'s were somewhat off from the "vector solitons" and the system evolved to a preferred state), but that the signal pulse remained essentially the same. The signal had the same phase variations as the corresponding NLS \( N = 1 \), so this particular combination was useless for switching. Perhaps some other "vector soliton" solution would be appropriate, but this approach doesn't look promising, particularly if very special pulse shapes are required.

Other possibilities exist. One exotic scheme would require one channel in the normal dispersion regime and the other in the anomalous dispersion regime. Trillo et.al. [18] simulated propagation of a bright pulse in the anomalous channel and a dark pulse in the normal channel, finding that this combination was stable until the dark pulse dispersed, but they did not discuss phase behavior. Our code would be easily modified to accommodate this set of conditions.

I will conclude this chapter by further discussing copropagation. Numerically, I have demonstrated that switching can be readily achieved in roughly two soliton periods with linearly polarized pulses, as long as we can maintain \( \delta \) small and \( R\delta \gg 1 \). For a unit amplitude and width NLS \( N = 1 \) signal, one could use a 0.3 amplitude and width NLS \( N = 1 \) as the control pulse, in order to achieve the above result. As I pointed out earlier, this is a nice mathematical result, but is rather impractical.

Circular polarization relieves us of our phase and group velocity restrictions, and we can achieve comparable switching. It is useful to assess distortion qualitatively in both the time and frequency domains. To demonstrate the effects of copropagation upon a unit amplitude, unit width sech signal, we first present

47
figures 4.9 and 4.10, for which the control pulse is $0.25 \text{ sech} s$. Figure 4.11 tracks the signal phase. We can achieve cleaner pulses, but with slower phase accumulation if we instead choose a control pulse of $0.2 \text{ sech} s$, as exhibited in figures 4.12 through 4.14. In both cases, the $u$ pulse is distorted only somewhat more than the $u$ pulse, manifesting small wings, but essentially maintaining its shape. This is quite different from linear polarization copropagation with nonnegligible coherence terms, for which the $v$ pulse undergoes major changes.
Figure 4.9: Initial (a) and π-phaseshifted (b) signal pulse shapes. Circular polarization. \( \eta_1 = 0.5 \), initial pulse 0.25 sech \( s \).
Figure 4.10: Initial (a) and π-phaseshifted (b) signal pulse Fourier transforms. Circular polarization. \( \eta_1 = 0.5 \), initial \( v \) pulse 0.25 sech \( s \).

50
Figure 4.11: Signal phase vs $\xi$. Circular polarization. $\eta_1 = 0.5$, initial $\nu$ pulse 0.25 sech $s$. 
Figure 4.12: Initial (a) and \( \pi \)-phaseshifted (b) signal pulse shapes. Circular polarization. \( \eta_1 = 0.5 \), initial \( v \) pulse 0.2 sech \( s \).
Figure 4.13: Initial (a) and $\pi$-phaseshifted (b) signal pulse Fourier transforms. Circular polarization. $\eta_1 = 0.5$, initial pulse 0.2 sech $s$. 
Figure 4.14: Signal phase vs $\xi$. Circular polarization. $\eta_1 = 0.5$, initial pulse 0.2 sech $s$. 

54
Chapter 5

Discussion

The coupled NLS's have been derived with coherence terms and allowing for the waves in the two channels to differ in frequency. A perturbation theory for the NLS due to Gordon, Haus, and Islam [15] was presented. With simplifying approximations, this approach was applied to the problem of estimating effects of scattering in the coupled NLS's. FORTRAN code has been developed for simulating the scattering. This code associates soliton parameters with the scattered solitary waves for comparison with the perturbation theory. Many simulations were executed, and sample results of a few of these have been presented. Several schemes for all-optical switching with solitary waves have been suggested.

The perturbation theory was shown to provide good results for sufficiently great walkoff. Simulations revealed that the discrepancy for small walkoff is related to "capture" of pulse energy.

The collision of linearly polarized solitary waves in linearly birefringent (polarization maintaining) fiber was found to be a poor means of switching. Alternatives have been suggested.
Copropagation of circularly polarized signal and control pulses provided the best results amongst the alternatives analyzed in this work. The phaseshifting can be achieved in roughly two soliton periods with minor signal pulse distortion. With circular polarization, soliton self-frequency shift sets the lower limit for useful pulsewidths. The pulses required are relatively easy to generate experimentally.
Appendix A

Simulation code

This appendix contains the complete simulation source code as of November 15, 1988. Chapter 4 contains a brief description of the program. In addition, comments appear throughout the code. The author strongly recommends that the interested reader peruse the input datafile (Appendix B) in conjunction with the source code.
@PROCESS DIRECTIVE('*COW')
c NOTE, the code has been modified presently so that the +/-slipbi
c du/dx terms are ignored. This is on physical grounds - HAH

c claims that the group velocity change couldn't be of the same

c order as the phase velocity change.

---

c IHALFFLAG has therefore been set to 1

c
---

Positive slip corresponds to the u channel as the fast axis.

---

original code: Sumanth Kaushik 1986

c modified by Randa Saif Hartemann 1987

c and substantially revised and augmented by John D. Moores 1987-1988

c MIKE includes a coherence term--energy exchange.

c Note: to run simulations without Raman, set taun < 1.d-25.

c BUT NEVER set tau0=0.

This program computes the time evolution of solitary wave

c pulses in a birefringent optical fiber.

c The required parameters are given below. They are defined in

c the data file 'soliton3.dat'. The program can operate in either of

c two fashions:

C (1) It can generate an initial profile for both polarizations based
on the height and width of the pulse supplied as parameters.

C (2) It can use an initial profile from a supplied data file (file= 
C <name2>). This initial profile can be either:

C (i) The last profile from a previous run. That is, each time
C program finishes, it produces a LAS file which contains al
C the necessary info to continue the run after the program's
C
C (ii) Initial profile generated by another program
C (say the ritz.f program). If using another program to gen
C initial profiles, make sure that they use the same parame
C ters as those supplied by 'soliton3.dat'.

C

---

block data shear
    common /filename,name,name2,ipnum(3),puname,pvname,aname
    common /polar/u(-1000:1000,0.1),v(-1000:1000,0.1)
    character*8 name,name2,puname,pvname,aname
    complex*16 u,v
    data u/4002*(1.d-15,0.d0)/,v/4002*(1.d-15,0.d0)/
    data ipnum/0,0,0/
program blast
implicit real*8(a-h,o,q-z)

common /filname/, name, name2, iipnum(3), puname, pvname, aname
common /polar/, u(-1000:1000,0:1), v(-1000:1000,0:1)
common /grid/dx, dt, ngridminus, ngridplus, ngrid
common /derivative/ dcoeff(20), iorder
C
common /imp/jlocuinit, jlocvinit, uoriginrelv, voriginrelv,
C & beta, slip, dt, rlambda, xloc, itact, n, nminus1
common /param/alphav, alphav2, alphav, alphau2
common /origin/voriginrelv, uoriginrelv, jlocuinit, jlocvinit

external errset

cmplx*16 u, v, uinit(-1000:1000)

complex*16 rpot1, rpot2, i, unj, vnj, dsqdx2, dsqvdx2
complex*16 vinit(-1000:1000), evphase, dt2i
complex*16 coh1, coh2, raman1, raman2, ecoh1, ecoh2, aacoh

character*8 name, name2, puname, pvname, aname, bname, p1name
1 , p2name, p3name, p4name, p5name, p6name, p7name, p8name

logical reducectime

real*8 upperv, x, dx, dx2i, beta, dt, umag(2001), vmag

1 , energyu, energyv, TotalEnergy, alphau2, alphau
2 , alphav2, alphav, rlambda, areau, areav, etal1, cmagli
3 , voriginrelv, uoriginrelv, rl1, r2, xjlocu, xjlocv
4 , yul, yvl, darg, slip, sliplbi, dtreal, dt1, dtmax
5 , tau1, tauv1, dcoefficient, tau0, taun, ramanrat, voruinit
6 , window, tau, zindex, zlambar, disp, wbar, dw, t0, vgbar
7 , acoh, bcoh, fcoh1, fcoh2, gcoh, hcoh, evphase, spacing
8 , avx, factor, filter, center, peak, w1, w, t
9 , etal1, xii, cmag1, phil, oldcenter, R, u9, u0, u1, s9, s0, s1

real*8 betah, c, cohtag, energyninit, omegacap, onee
1 , pi, rmagusq, rmagvsq, theta, timel, xtest, xjlocuinit, xjlocvinit
2

real*8 peak, phil, phiref, pi, pi2, umag(2001), polariz

integer locmax, itact, j

data i/(0.d0,1.d0)/
call errset (207,1,1,2)

c 145 format(1x,a18,e17.10)
147 format(1x,a6,' dw/w',e13.7,'c',i1,' h',i1,'x
148 & s',e11.4,' sb',f9.5,' R',f6.1/1x
2 , 'dt',e8.3,'u',f9.6,'v',f9.6/1x
3 , 'ng',i5,' vph',f6.4)
148 format(1x,3(1x,e17.10))
149 format(1x,2(1x,e17.10))

c A variable with one of the dumbest names you've ever seen, but which
c saves CPU cycles.
one=1.d0/180.d0
c Some universal constant (perhaps) in m/s.
c=2.997925d8
c Another universal constant, but this one is dimensionless.
pi=3.1415926535897932384626d0
pi2=2.d0*pi
c Initialize.
ilastrun=0

c Set an upper limit on the number of spatial iterations.
itactmax=200000

c extraphi is to make sure phases are positive so the stupid DMOD
c function will work properly.
extraphi=5.d0*pi2

C------------- REQUIRED PARAMETERS -------------

open(unit=8,file='soliton3')

read(8,*)iboundary
read(8,*)isymmetric
if ((isymmetric.ne.0).and.(isymmetric.ne.1)) then
   print *, 'invalid isymmetric'
goto 999
endif
read(8,*)ngrid
read(8,*)dt
read(8,*)dtmax
read(8,*)voriginrelu
read(8,*)window
cc
voriginrelu=window*.5d0*spacing

c
ctrofmass=.5d0*voriginrelu

c The center of mass of two equal pulses is simply voriginrelu/2.
read(8,*)tau
t0=.568d0*tau
read(8,*)zindex

c Note that Manyuk's vgbar, used here, is defined as the inverse of
c the arithmetic mean of the inverse group velocities in the
c two channels. I'm approximating it by c/n.
vgbar=c/zindex
read(8,*)zlambardisp
read(8,*)disp

c the "mean" freespace angular frequency wbar is determined by
c zlambardisp.
read(8,*)iasym
wbar=2.d0*pi*c/zlambardisp
read(8,*)omegacap
dw=omegacap*wbar
acoh=2.d0*wbar*t0
R=2.d0*acoh
bcch=2.d0*pi*c*omegacap*t0/(vgbar*zlambar*disp)
read(8,*)alphau2
read(8,*)alphau

c These initial soliton parameters will be N=1 only for c alphau2=alphau.

etalii=.5d0*alphau2
cmgl=alphau2
read(8,*)alphav2
read(8,*)alphav
read(8,*)vphase
evphase=dcos(vphase)+i*d*sin(vphase)
read(8,*)beta
betah=.5*beta
read(8,*)cohflag
if (cohflag.eq.0.d0) then
  icohflag=0
else
  icohflag=1
  if (isymmetric.eq.1) then
    print *, 'isymmetric and icohflag are both on.'
    print *, 'You will have to turn at least one off.'
    goto 999
  endif
endif
coh1=0.d0
coh2=0.d0
read(8,*)tau0
if (tau0.le.0) goto 999
read(8,*)taun
if (taun.lt.0.d0) goto 999
raman1=0.d0
raman2=0.d0
read(8,*)rlambda
read(8,*)slipbi
slip=slipbi-t0*dw
read(8,*)ihalfflag
if (ihalfflag.ne.0) ihalfflag=1
if (ihalfflag.eq.1) then
  slip=0.d0
  slipbi=0.d0
endif
if ((slip*voriginreul).lt.0) then
  print *, 'no collision - change slip/voriginreul sign.'
  goto 999
endif
read(8,*)irunlength
if ((irunlength.ne.0).and.(irunlength.ne.1)) then
  print *, 'invalid irunlength'
  goto 999
endif
read(8,*)factor
read(8,*)itmax
if (irunlength.eq.1) then
  if (ihalfflag.eq.1) then
    print *, 'both irunlength and ihalfflag=1-incompatible'
goto 999
  endif
  itmax=abs(idnint(voriginrelu*(1.d0+factor))/(slip*dt)))
endif
print *, 'itmax=', itmax
if (itmax.gt.itactmax) then
  print *, 'itactmax = ', itactmax
  goto 999
endif
iareaint=itmax/999+1
read(8,*)filter
if (filter.lt.0.d0) then
  iflipbeta=1
  filter=-filter
endif
itfilter=idnint(filter*itmax)
  print *, 'itfilter', itfilter
read(8,*)iprint
read(8,*)itrack
read(8,*)rm
read(8,142)name
  print *, 'name as read = ', name
read(8,*)model
read(8,142)puname
read(8,142)pvname
read(8,142)aname
  bname=aname
  p1name=aname
  p2name=aname
  p3name=aname
  p4name=aname
  p5name=aname
  p6name=aname
  p7name=aname
  p8name=aname
  bname(2:2)='A'
p1name(2:3)='P1'
p2name(2:3)='P2'
p3name(2:3)='P3'
p4name(2:3)='P4'
p5name(2:3)='PH'
p6name(2:3)='PC'
p7name(2:3)='PD'
p8name(2:3)='PL'
write(*,*)aname,bname
read(8,*)mode2
read(8,142)name2

close(8)

C------------------------------------------------------------------------
call xuflow(0)

itact=0
it=0
n=1
nminus1=0
ng1=ngrid/2
ngrid=ng1*2
zng1=1.0d0/float(ngrid)

C---- Differencing is done to 6th order (errors are in 8th order) ----
oorder=6
ngridminus=-ngrid/2
ngridplus=ngrid/2

C--- Determine Grid Spacing ---
dx=window*zng1
dx1=1.0d0/dx
dx2i=dx1*dx1

ramanrat=taun/(2.0d0*dx*tau0)

name(2:4)=’000’
puname(2:4)=’000’
pvname(2:4)=’000’
cc aname(2:4)=’000’

C------------------------------------------------------------------------
C Determine Relative Origins of the polarized pulses
C
C In the case of solitary wave collision,
c the U-polarization travels from right
to left and the V-polarization travels from left to right.
C
C Variable Definitions:
C
voriginrelu: Center of V-polarization at time t with the x=0
            the center of the U-polarization
uoriginrelv: Center of U-polarization at time t with the x=0
            the center of the V-polarization
jlocuinit: The grid number of the U-polarization that corr
           to the x=-xmax for the U-polarization
jlocvinit: The grid number of the U-polarization that corr
           to the x=-xmax for the V-polarization

63
C Illustrative Sketch: Take two pulses (given as two triangles below) with the left one being the V-polarization and the right one being the U-polarization.

```
  /\ /\  \
 / \ V/ \  \
\  \ /  \  \\
```

X axis --> x=0 x=uoriginrelv

x=voriginrelu x=0

C-----------------------------------------------

if (model.eq. 1) then
  voriginrelu=0.d0
  uoriginrelv=0.d0
  jlocuinit=ngridminus
  jlocvinit=ngridminus
else
  c SPACING is the fraction of (ngrid/2) separating the initial U and V c pulses.
  voriginrelu=window*.5d0*spacing
  voruininit=voriginrelu
  jxloc=voriginrelu*dx1
  voriginrelu=dx*float(jxloc)
  uoriginrelv=-voriginrelu
  jlocuinit=ngridminus-jxloc
  jlocvinit=ngridminus+jxloc
endif

cccc open(unit=8,file=name)

if (mode2 .eq. 1) then

C-------- GENERATE INITIAL PROFILE -------------------------------

  TotalEnergy=0.
  areau=0.d0
  areav=0.d0
  energyu=0.d0
  energyv=0.d0
  ngm3=ngridminus-iorder/2-1
  ngp3=ngridplus+iorder/2+1
  do 18 j=ngm3,ngp3
    jm=j+1001
    x=dx*float(j)
    avx=alphav*x
    if (avx.gt.175.d0) avx=175.d0
    if (avx.lt.-175.d0) avx=-175.d0
    u(j,n)=dcmplx(alphau2/dcosh(alphau*x),0.d0)
    if (iasymmm.ne.1) then

64
v(j,n)=dcmplx(alphav2/dcosh(avx), 0.d0)*evphase
else
v(j,n)=dcmplx(alphav2*dsinh(avx)/(dcosh(avx)*
dcosh(avx)), 1.d-15)*evphase
endif
umag(jm)=cdabs(u(j,n))
vmag=cdabs(v(j,n))
rmagusq=umag(jm)*umag(jm)
rmagvsq=vmag*vmag
areau=areau+umag(jm)
areav=areav+vmag
energyu=energyu+rmagusq
energyv=energyv+rmagvsq

cc
TotalEnergy=TotalEnergy + energyu+energyv

18 continue
c
Energy and area, + OPEN's for "soliton parameters."
c
energyu=energyu*dx
energyv=energyv*dx
areau=areau*dx
areav=areav*dx
c
open(4, file=aname)
open(5, file=bnname)
c
write(4,147)'Energy ', omegacap, icohflag, ihalflflag,
slip,slipbi,R, dt, alphau2, alphav2, ngrid, vphase
write(5,147)'Area ', omegacap, icohflag, ihalflflag,
slip,slipbi,R, dt, alphau2, alphav2, ngrid, vphase
if (itrack.eq.1) then
open(13, file=p1name)
open(14, file=p2name)
open(15, file=p3name)
open(16, file=p4name)
open(17, file=p5name)
open(19, file=p6name)
open(20, file=p7name)
open(21, file=p8name)
write(13,147)'xii ', omegacap, icohflag, ihalflflag,
slip,slipbi,R, dt, alphau2, alphav2, ngrid, vphase
write(14,147)'eta1 ', omegacap, icohflag, ihalflflag,
slip,slipbi,R, dt, alphau2, alphav2, ngrid, vphase
write(15,147)'cmag1 ', omegacap, icohflag, ihalflflag,
slip,slipbi,R, dt, alphau2, alphav2, ngrid, vphase
write(16,147)'phi1-ref', omegacap, icohflag, ihalflflag,
slip,slipbi,R, dt, alphau2, alphav2, ngrid, vphase
write(17,147)'phase-', omegacap, icohflag, ihalflflag,
slip,slipbi,R, dt, alphau2, alphav2, ngrid, vphase
write(19,147)'uscenter', omegacap, icohflag, ihalfflag,
        slip, slip1, R, dt, alphau2, alphav2, ngrid, vphase
write(20,147)'final va', omegacap, icohflag, ihalfflag,
        slip, slip1, R, dt, alphau2, alphav2, ngrid, vphase
write(21,147)'u pk pol', omegacap, icohflag, ihalfflag,
        slip, slip1, R, dt, alphau2, alphav2, ngrid, vphase
endif

c
write(4,148) 0.d0, energyu, energyv
write(5,148) 0.d0, areau, areav
cc
TotalEnergy=TotalEnergy*dx
TotalEnergy=energyu+energyv
call fillnd(itact, TotalEnergy, n, slip, slip1, slip, ipint,
        omegacap, icohflag, vphase, alphau2, alphav2, ilastrun, R
        , ihalfflag, isymmetric)
else

C-------- MODE 2: READ FROM <name2> FILE --------

c
I never use this. If you want to use it, check to see which
file it reads, how many comment lines it expects at the beginning
c of this file, etc.
c
name2(2:4)= 'LAS'
open(unit=9, file=name2)
read(9,79) itact, ipnum(3), ipnum(2), ipnum(1)
ngridminus = -ngrid/2
ngridplus = ngrid/2
77 read(9,78, END=100) j, u(j,n), v(j,n)
go to 77
78 format(I4,3x, (' ', E25.18E2, ',', E25.18E2, ','), 1x,
    (' ', E25.18E2, ',', E25.18E2, ',')
79 format(I4,2x,I1,2x,I1,2x,I1)
endif
100 close(9)

energyinit=TotalEnergy

C*****************************************************************************
C*****************************************************************************
C*****************************************************************************
C*****************************************************************************

C Note: The leap frog technique center differences the time derivative.
C Hence, U(n+1,j) requires information from U(n-1,j) and U(n,j).
C At t=0, only information for U(0,j) is known (this is the
C initial profile). Thus, to compute for U(3,j), we need to
C somehow generate U(2,j). This can be done by solving for
U(2,j), V(2,j) using an UNSTABLE backward difference explicit algorithm.

If giant time steps are needed, the initial unstable explicit method causes serious instabilities (i.e., picket fence behavior is observed in energy). Thus, we must find a more stable way of computing the second profile needed to do leap frog. This is accomplished by breaking up the big time steps into smaller time steps. The t=0 profiles are stored as UINIT and VINIT. We then compute the profiles using the same algorithm as we would if the time steps were not too large. Then, once we get to t=dtreal (the large time steps desired), we use the uinit,vinit profile at t=0 and use the computed profile at t=dtreal as the second profile and then revert to the big time steps <dtreal>. This way, we get a stable second profile.

Variable Definitions:

- **dtreal:** The actual big time steps desired
- **dtmax:** The criteria by which the time step is judged to be big or not. This is determined by trial and error. All that is required of dtmax is that if the time step increment was selected to be dtmax, then the computation must be stable dtmax is smaller than dtreal.

```fortran
if (dt .gt. dtmax) then
  reducedtime=.true.
dtreal=dt
dt1=dtreal/dtmax
nredtsteps=int(dt1)
dt=dtreal/float(nredtsteps)
dt2i=2.d0*i*dt
slipdt=slip*dt
    do 711 j=ngm3,ngp3
        uinit(j)=u(j,n)
        vinit(j)=v(j,n)
    enddo
```

---

67
2 \text{continue}
   \text{it}=\text{it}+1
   \text{itact}=\text{itact}+1
\text{cc} \quad \text{if (reducedtime) t}=\text{dtmax}*\text{it}
\text{c} 
\text{I've set a max "itact" in case the program didn't terminate}
\text{c} \quad \text{for some reason, like a user not understanding how to properly}
\text{c} \quad \text{establish run length.}
\text{c} 
\text{if (itact.gt.77777) goto 999}
\text{if (reducedtime .and. (itact .gt. nredtsteps)) then}
\quad \text{reducedtime=.false.}
\text{c} 
\quad \text{nredtsteps}=0
\quad \text{dt}=\text{dtreal}
\quad \text{dt2i}=2.d0*\text{i}*\text{dt}
\quad \text{slipdt}=\text{slip}*\text{dt}
\quad \text{do 712 } j=\text{ngm3},\text{ngp3}
\quad \quad \text{u}(j,\text{mminus1})=u\text{init}(j)
\quad \quad \text{v}(j,\text{mminus1})=v\text{init}(j)
\quad \text{712 continue}
\quad \text{itact}=2
\text{endif}
\text{cc} \quad \text{if ((itact .gt. itmax) .and. (.not. reducedtime)) goto 8}
\text{if (itact.gt.itmax) goto 8}
\text{cclast} \quad \text{if (itact.eq.itmax) ilastrun=1}
\text{t}=\text{dfloat(itact)}*\text{dt}
\text{c} \quad \text{Are we unstable ?}
\quad \text{if ((energyu+energyv).gt.(energyinit*1.1d0)) goto 998}
\quad \text{energyu}=0.d0
\quad \text{energyv}=0.d0
\quad \text{areau}=0.d0
\quad \text{areav}=0.d0
\quad \text{iright}=0
\text{c} 
\quad \text{center}=0.d0
\quad \text{if (itrack.ne.1) goto 249}
\quad \text{peak}=0.d0
\quad \text{s9}=-1000.d0
\quad \text{s0}=-1000.d0
\quad \text{s1}=-1000.d0
\quad \text{u9}=0.d0
\quad \text{u0}=0.d0
\quad \text{u1}=0.d0
\quad \text{249} \quad jlocu=jlocuinit
\quad \text{jlocv}=jlocvinit
\text{c} 
\text{C Coherence term components.}
if (iCOHflag.eq.0) goto 241
  gcoh=slip*omegacap*.5d0
  fcohl=acoh*t*(slip-bcoh-gcoh)
  fcoh2=acoh*t*(slip-bcoh+gcoh)
  ecoh1=dcos(fcohl)-i*dsin(fcohl)
  ecoh2=dcos(fcoh2)+i*dsin(fcoh2)

C---------------------------------------------------------------------
C------------------------ TIME LOOP -------------------------------
C---------------------------------------------------------------------

241    do 4 j=ngridminus, ngridplus
       jm=j+1001

C In pulse V frame, compute the corresponding value of U polarization at
C node j. By V frame, it is meant that j=0 implies x=0 for pulse V. Th
C the center of pulse U in the V frame would be x=uooriginrelv
C
C But first, make sure we aren't going to do unnecessary work.
C
C Equal pulses in the two channels?
C
  if (isymmetric.eq.0) goto 267
  if iright already = 1, vnj already = (1.d-15,0)
     if (iRIGHT.eq.1) goto 269
     vflip=originrelv*dx-j
     if (vflip.ge.0.d0) then
       jvleft=int(vflip)
     else
       jvleft=int(vflip-1.d0)
     endif
     if (((jvleft.ge.ngridplus).or.(jvleft lt ngridminus)) then
       iRIGHT=1
       vnj=(1.d-15,0.d0)
       goto 269
     endif
     jvright=jvleft+1
     dvflip=vflip-dx*jvleft
     vnj=v(jvleft,n)+dvflip*dx1*(v(jvright,n)-v(jvleft,n))
     goto 269

C
C Copropagating?
C
267    if ((j.gt.0).and.(IHALFflag.eq.1)) then
u(j,nminus1)=u(-j,nminus1)
v(j,nminus1)=v(-j,nminus1)
if ((itrack.eq.1).and.(j.lt.20)) umag(1001+j)=umag(1001-j)
goto 4
endif

c
On with the loop:
c
x=dfloat(j)*dx
r1=uoriginrelv + x
xjlocu=float(jlocu)*dx
101 if ((xjlocu .le. r1) .and. (jlocu .lt. 1000)) then
    jlocu=jlocu+1
    xjlocu=float(jlocu)*dx
    goto 101
endif

C In pulse U frame, compute the corresponding value of V polarization a
C node j. By U frame, it is meant that j=0 implies x=0 for pulse U. Thu
C the center of pulse V in the U frame would be  x=voriginrelv

r2=voriginrelu + x
xjlocv=float(jlocv)*dx
102 if ((xjlocv .le. r2) .and. (jlocv .lt. 1000)) then
    jlocv=jlocv+1
    xjlocv=float(jlocv)*dx
    goto 102
endif

C Perform Interpolation to Compute Total Potential at Node j. In the U
C frame, the potential would be a function
C of U(n,j) and the corresponding
C value of V at the node j, taking into account the relative position o
C U and V as determined above

if ((jlocv .gt. -1000) .and. (jlocv .lt. 1000)) then
    unj=u(jlocv-1,n) + (u(jlocv,n)-u(jlocv-1,n))*(r2-xjlocv+dx)
    *dx
else
    unj=0.d0
endif

if ((jlocu .gt. -1000) .and. (jlocu .lt. 1000)) then
    vnj=v(jlocu-1,n) + (v(jlocu,n)-v(jlocu-1,n))*(r1-xjlocu+dx)
    *dx
else
    vnj=0.d0

70
endif

c Compute the Profile.
c
269 continue

c----------Coherence:  
   if (icohflag.eq.0) goto 246
   hcoh=acoh*omegacap*x
   aacch=dcos(hcoh)+i*dsin(hcoh)
   cohl=(1.0d0-rlambda)*vnj*vnj*dconjg(u(j,n))*ecohl*aacoh
   if (isymmetric.eq.1) goto 246
   coh2=(1.0d0-rlambda)*unj*unj*dconjg(v(j,n))*dconjg(ecohl*aacoh)

c----------Cubic term, with coupling:  
246  rpot1=(u(j,n)*dconjg(u(j,n)) + vnj*dconjg(vnj)*rlambda)*u(j,n)
   if (isymmetric.eq.1) goto 271
   rpot2=(v(j,n)*dconjg(v(j,n)) + unj*dconjg(unj)*rlambda)*v(j,n)

c----------Raman:  
271  if (dabs(taun).le.1.d-25) goto 272
   raman1=ramanrat*(u(j+1,n)*u(j+1,n)*dconjg(u(j+1,n))-
&       u(j-1,n)*u(j-1,n)*dconjg(u(j-1,n)))
   if (isymmetric.eq.1) goto 272
   raman2=ramanrat*(v(j+1,n)*v(j+1,n)*dconjg(v(j+1,n))-
&       v(j-1,n)*v(j-1,n)*dconjg(v(j-1,n)))

c----------GVD:  
272  dsqdudx2=(2.d0*(u(j+3,n)+u(j-3,n))-27.d0*(u(j+2,n)+u(j-2,n))+
&       270.d0*(u(j+1,n)+u(j-1,n)) - 490.d0*u(j,n))^onee*dx21
   if (isymmetric.eq.1) goto 273
   dsqvdv2=(2.d0*(v(j+3,n)+v(j-3,n))-27.d0*(v(j+2,n)+v(j-2,n))+
&       270.d0*(v(j+1,n)+v(j-1,n)) - 490.d0*v(j,n))^onee*dx2i
C  dsqduvx2=(u(j+1,n)+u(j-1,n)-2.d0*u(j,n))*dx2i
C  dsqvdvx2=(v(j+1,n)+v(j-1,n)-2.d0*v(j,n))*dx2i

C------------------------------------------------------------------

C At t=1, we use the unstable explicit method where
C
C   dU
C   -- = (U(n,j)-U(n-1,j))/dt
C   dt
C
C instead of the leapfrog method where
C
C   dU
C   -- = (U(n+1,j)-U(n-1,j))/2*dt
C   dt
C
C------------------------------------------------------------------

273  if (itact.gt. 1) then
\[ u(j,n+1) = (u(j,n) + \Delta t2i^{*} (\text{beta}h^{*}\text{dsqxdz2} + \Delta t) + r\text{pot1} + \text{coh1} + \text{raman1}) ] \\
\text{if (isymmetric.eq.0)} v(j,n+1) = v(j,n) \\
\text{else} \\
u(j,n+1) = u(j,n) + \Delta t (\text{beta}h^{*}\text{dsqxdz2} + \Delta t) + r\text{pot1} + \text{coh1} + \text{raman1} ] \\
\text{if (isymmetric eq.0)} v(j,n+1) = v(j,n) + \Delta t \\
\text{else} \\
\text{endif}
\]

This is where we can apply a polarizer to the fiber to eliminate the v pulse.

\[ \text{if (itact.eq.itfilter)} \text{then} \]
\[ \text{if (iflipbeta.eq.1)} \text{then} \]
\[ \text{beta} = -\text{beta} \]
\[ \text{beta}h = -\text{beta}h \]
\[ \text{else} \]
\[ v(j,n+1) = (1.0 - 15.0, 0.0) \]
\[ \text{endif} \]
\[ \text{endif} \] 

Energy, area, and "soliton parameters"

\[ \text{if (ihalfflag.ne.0).and.(itrack.eq.0)} \text{goto 4} \]
\[ \text{continue} \]
\[ \text{if (cdabs(u(j,n+1)).lt.1.0)} \text{then} \]
\[ \text{umag jm} = 0.0 \]
\[ \text{goto 328} \]
\[ \text{endif} \]
\[ \text{umag jm} = \text{cdabs(u(j,n+1))} \]
\[ \text{if (ihalfflag.ne.0).and.(itrack.eq.1)} \text{goto 4} \]
\[ \text{darea} = \text{dx*umag jm} \]
\[ \text{area} = \text{area + darea} \]
\[ \text{energy} = \text{energy} + \text{darea*umag jm} \]
\[ \text{if (isymmetric.eq.1)} \text{goto 4} \]
\[ \text{if (cdabs(v(j,n+1)).lt.1.0)} \text{then} \]
\[ \text{vmag} = 0.0 \]
\[ \text{goto 45} \]
\[ \text{endif} \]
\[ \text{vmag} = \text{cdabs(v(j,n+1))} \]
\[ \text{darea} = \text{dx*vmag} \]
\[ \text{area} = \text{area} + \text{darea} \]
\[ \text{energy} = \text{energy} + \text{darea*vmag} \]
\[ \text{if (j.eq.0).and.(ihalfflag.eq.1)} \text{then} \]
\[ \text{energy} = \text{energy} - \text{darea*umag jm}^{*} \text{.5d}0 \]
\[ \text{energy} = \text{energy} - \text{darea*vmag}^{*} \text{.5d}0 \]
\[ 72 \]
areau=areau-dareau*.5d0  
areav=areav-dareav*.5d0  
endif  

4  continue  

---------------------------------------------------------------------  
END OF TIME LOOP  
---------------------------------------------------------------------  

if (ihalfflag.eq.1) then  
  energyu=2.d0*energyu  
  energyv=2.d0*energyv  
  areau=2.d0*areau  
  areav=2.d0*areav  
endif  

---------------------------------------------------------------------ENERGY, AREA OUTPUT + SOLITON PARAMETER EXTRACTION-----------------------  
---------------------------------------------------------------------  

if ((.not. reducedtime).and.(mod(itact,iareaint).eq.0)) then  
  write(4,148) t,energyu,energyv  
  write(5,148) t,areau,areav  
  if (itrack.eq.1) then  
    locmax=idmax(2001,umag,1)  
    u9=umag(locmax-1)  
    u0=umag(locmax)  
    u1=umag(locmax+1)  
    s0=dsfloat(locmax-1001)*dx  
    s9=s0-dx  
    s1=s0+dx  
    u11=umag(locmax-5)  
    url=umag(locmax+5)  
    s11=s0-5.d0*dx  
    srl=s0+5.d0*dx  
  vupkmax is the magnitude of the field in the  
v channel at (roughly) the location  
of the peak magnitude of the u channel. I haven't bothered  
to interpolate to the accuracy of Kaushik, since I'm just estimating  
amplitude ratio, and there is no accumulation of error.  
  vupkmax=cdabs(v(locmax-1001  
1 -idnint(voriginrelu*dx1),nminus1))  
  c quadratic peak fit: see red notebook #1, p.215  
  c see p.219 for a sech fit  
  c N.B. There should be conditionals with alternate methods of  
  c approximating 'center' and 'peak' to avoid very small denominators.  
  c oldcenter=center  
  deno1=2.d0*u0-u9-u1  
  if (dabs(deno1).gt.1.d-60) center=s0+.5d0  
1 *dx*(u1-u9)/(2.d0*u0-u9-u1)  
  w1=s0-center  
  w=(w1/(w1+dx))**2
\begin{verbatim}
wm1=1.d0-w
if (dabs(wm1).gt.1.d-60) peak=(u0-w*u1)/wm1

   c End of quadratic fit.
   c Now search and interpolate to approximate pulse width. This is
   c one-sided: it assumes that the pulse is symmetric.
   c Presently, this is only calculated at the last spacetime.

   if (itact.lt.(itmax-1areaint)) goto 24
   halfmax=.5d0*dsqrt(2.d0)*peak
   j=locmax+1
   c As I define sfwhm below, it is actually half the FWHM.

   23     j=j+1
   if (umag(j).lt.halfmax) then
       sfwhm=dx*(dfloat(j-1001)-(halfmax-umag(j)))/
       (umag(j-1)-umag(j)))
       goto 24
   endif
   goto 23

   24     continue

   c Now we can extract.

   c xi

   xil=-.5d0*(center-oldcenter)/(dt*dfloat(iareaint))
  ccc
   xil=-.5d0*(center-oldcenter)/(t-oldt)
   theta9=atan2(dimag(u(locmax-1002,mminus1)),
   1       dreal(u(locmax-1002,mminus1)))
   thetal=atan2(dimag(u(locmax-1000,mminus1)),
   1       dreal(u(locmax-1000,mminus1)))
   if ((thetal-theta9).gt.4.d0) theta9=theta9+pi2
   if ((theta9-thetal).gt.4.d0) thetal=thetal+pi2
   thetarl=atan2(dimag(u(locmax-1006,mminus1)),
   1       dreal(u(locmax-1006,mminus1)))
   thetar1=atan2(dimag(u(locmax-996,mminus1)),
   1       dreal(u(locmax-996,mminus1)))
   oldt=t
   xil=.5d0*(.25d0*(theta9-thetal)+.05d0*(thetalall-thetal1))*dxi
  c eta -  I'm using amplitude as the eta-determining factor
  c rather than width. In doing this, I am demonstrating
  c Hamilton's principle of least work.
   etal=.5d0*peak

   c cmag
   cmag1=peak*dexp(peak*(center+2.d0*x11*t))

   c center is frequency independent. If x11=0, the center of the
   c pulse is at s=scenter (x in code notation)
   scenter=dlog(peak/cmag1)/peak

   c phi - note this is the phase of the residue, not the phase of
   c the wave. It's sign is a matter of convention.
\end{verbatim}
theta = atan2(dimag(u(loccounty-1001,nminus1)),
       dreal(u(loccounty-1001,nminus1)))
phii = theta - 2.0d0*(xil*center - (xil*xil - etai*etai)*t) + extraphi

if (phii.lt.0.0d0) then
   extraphi = extraphi + 5.0d0*pi2
   phii = phii + extraphi
   endif
if (phii.gt.0.0d0) then
   extraphi = extraphi + 5.0d0*pi2
   phii = phii - extraphi
endif
phii = dmod(phii,pi2)
phiref = dmod(phiref,pi2)

!pol:normal - printed in degrees, not radians.
!Note that this really doesn't tell you much about
!the polarization state: it is the arctangent of the
!amplitude ratio, but ignores the phase variation.

polariz = 180.0d0 * datan2(vupkmag, peak)/pi

dump
write(13,149)t, xil
write(14,149)t, etai - etai
write(15,149)t, cmag1 - cmag1
write(16,149)t, phii
write(17,149)t, phiref
write(19,149)t, scenter
write(21,149)t, polariz
endif
endif

!----------------------------------------Boundary Conditions----------------------------------------
!
!Three possible ways to take care of boundary conditions:
!boundary=1: Fit exponential curve to tail to provide values outside boundary (i.e. for |t| > ngridplus)
!boundary=2: Set all values outside boundary to value at boundary
!boundary=2: All values outside boundary are zero.

if (iboundary .eq. 1) then
   do 524 j7 = 1, 2
      j5 = ngridplus
      j6 = ngridplus - 1
      if (j7 .eq. 2) then
         j5 = -j5
         j6 = -j6
      endif
      yu1 = cdbas(u(j5,nminus1))/cdbas(u(j5,nminus1))
      yv1 = cdbas(v(j6,nminus1))/cdbas(v(j5,nminus1))
      endif
524 continue
\[ \tau_{uji} = \text{dlog}(y_{j1}) \times dx \]
\[ \tau_{uvi} = \text{dlog}(y_{v1}) \times dx \]
\[ \text{n terms} = \text{order}/2 \]
\[ \text{do} 523 \ j3 = 1, \text{n terms} \]
\[ j4 = \text{ngrid plus} + j3 \]
\[ \text{if} (j7 \ .eq. 2) \ j4 = -j4 \]
\[ \text{darg} = \text{float}(j3) \times dx \]
\[ u(j4, nminus1) = u(j5, nminus1) \times \exp(-\text{darg} \times \tau_{uji}) \]
\[ v(j4, nminus1) = v(j5, nminus1) \times \exp(-\text{darg} \times \tau_{uvi}) \]
\[ \text{continue} \]
\[ 523 \]
\[ \text{continue} \]
\[ 524 \]
\[ \text{else} \]
\[ \text{if} (\text{iboundary} .eq. 2) \text{ then} \]
\[ \text{n terms} = \text{order}/2 \]
\[ \text{do} 525 \ j3 = 1, \text{n terms} \]
\[ u(\text{ngrid plus} + j3, nminus1) = u(\text{ngrid plus}, nminus1) \]
\[ v(\text{ngrid plus} + j3, nminus1) = v(\text{ngrid plus}, nminus1) \]
\[ u(\text{ngrid minus} - j3, nminus1) = u(\text{ngrid minus} - j3, nminus1) \]
\[ v(\text{ngrid minus} - j3, nminus1) = v(\text{ngrid minus} - j3, nminus1) \]
\[ 525 \]
\[ \text{continue} \]
\[ \text{endif} \]
\[ \text{endif} \]

c

C------ Adjust Coordinate Frame -----------------------------------------------
c

c  *************** No group velocity difference ****************************
\[ \text{if} (\text{icoh flag} .eq. 1) \text{ then} \]
\[ \text{if} (\text{slip} .ne. 3.2d17) \text{ goto } 171 \]
\[ \text{endif} \]
\[ \text{if} (\text{slip} .eq. 0.d0) \text{ goto } 171 \]
c  Check that the below corresponds to the coherence terms. 
c  Note however that for coherent pulses the group velocity 
c is negligible compared to the phase velocity. 
\[ \text{vorigin rel u} = \text{vorigin rel u} - \text{slip dt} \]
\[ \text{uorigin rel v} = \text{uorigin rel v} + \text{slip dt} \]
\[ \text{x test} = \text{vorigin rel u} + \text{float(\text{ngrid minus})} \times dx \]
\[ \text{xj loc init} = \text{float(\text{j loc init})} \times dx \]
\[ 104 \text{ if } ((\text{xj loc init} .lt. \text{x test}) \ \text{and.} (\text{j loc init} .lt. 1000)) \text{ then} \]
\[ \text{j loc init} = \text{j loc init} + 1 \]
\[ \text{xj loc init} = \text{float(\text{j loc init})} \times dx \]
\[ \text{goto } 104 \]
\[ \text{endif} \]
\[ \text{x test} = \text{uorigin rel v} + \text{float(\text{ngrid minus})} \times dx \]
\[ \text{xj loc init} = \text{float(\text{j loc init})} \times dx \]
\[ 103 \text{ if } ((\text{x test} .le. \text{xj loc init}) \ \text{and.} (\text{j loc init} .gt. -1000)) \text{ then} \]
\[ \text{j loc init} = \text{j loc init} - 1 \]
\[ \text{xj loc init} = \text{float(\text{j loc init})} \times dx \]

76
goto 103
endif

C****************************************************************************** Compute Total Energy of Two Pulses ******************************************************************************
C****************************************************************************** Total energy must be conserved ******************************************************************************
C******************************************************************************

171  TotalEnergy=energyu+energyv
c cc  energyv=0
c c  rmagu=cdabs(u(0,n))
c c  rmagv=cdabs(v(0,n))
c c  uov=uoriginrelv
c c  vou=voriginrelu
if (.not. reducedtime) then
  n3=(ipint)*((itact)/ipint)
c time1=itact*dt
  if (n3 .eq. itact) call filhnd(itact,1,TotalEnergy,n,slipbi,slip,ipint,omegacap,icohflag
  vphase,alphau2,alphav2,ilastrun,R,ihalfflag,isymmetric)
c cc  if (name(2:4).eq.'70') goto 8
endif

n=abs(n-1)
nminus1=abs(nminus1-1)
goto 2
c
C******************************************************************************************** END OF SPACE LOOP ********************************************************************************************
8  continue
c ccc close(9)

C****************************************************************************************** Detailed Output of Last Time Step **********************************************************************************

ilastrun=1
cc call filhnd(itact,TotalEnergy,n,slipbi,slip,ipint,1
  omegacap,icohflag,vphase,alphau2,alphav2,ilastrun,R
  2 ,ihalfflag,isymmetric)
C
 name2(2:4)='LAS'
C open(unit=9, file=name2)
C write(9,500) itact,ipnum(3),ipnum(2),ipnum(1)
C 500  format(I4,2x,I1,2x,I1,2x,I1)
C do 343 j=ngridminus,ngridplus
C write(9,501) j,u(j,nminus1),v(j,nminus1)
C write(9,* )u(j,nminus1)
C 501  format(I4,3x,'(E25.18E2',E25.18E2')',1x,
C ' (E25.18E2',E25.18E2')
C print *,u(j,nminus1)
C 526  format(2D25.18)
C 343  continue
C close(9)

142 format(a8)
close(4)
close(5)
if (itrack.eq.1) then
close(13)
close(14)
close(15)
close(16)
close(17)
close(19)
close(21)
write(20,*)
write(20,*)'t = ',t
write(20,*)
write(20,*)
write(20,145)'xil ',xil
write(20,145)'etal change ',etal-etal1
write(20,145)'cmagl change ',cmagl-cmagli
write(20,145)'phil ',phil
write(20,145)'pulse phase (rad) ',phiref

c Again remember that scenter is not the value of s (x in the code)
c corresponding to the pulse peak.
write(20,145)'center shift ',scenter
write(20,145)'FWHM est. ',2.d0*sffwhm
write(20,145)'u pk. polariz. (deg)',polariz
close(20)
endif
goto 999

998 print *, 'energy too large'
print *, 'energy = ',(energyu+energyv)
print *, 'energyinit = ',energyinit
print *, 'itact = ',itact
print *, 'j = ',j
print *, 'ngridminus = ',ngridminus

999 stop

END

-------------------------------------------------------------
C<<<<<<<<<<<END OF MAIN PROGRAM>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
-------------------------------------------------------------
C
C---------------------------------------------------------- FUNCTION PHASE---------
cc function phase(z)
c implicit real*8(a)
c complex*16 z
c
c y=dimag(z)
c x=dreal(z)
c arc=atan2(y,x)
c angle = arc/3.14159265358979323846d0
c phase = angle

c return

c end

C--------------------------------------------- FUNCTION DACOSH-----

c Inverse hyperbolic cosine.
double precision function dacosh(x1)
real*8 x1, d1,d2, cow

cow = 1.d60

c The lack of dabs is intentional.
if (x1 .lt. 1.d0) x1 = cow*cow

if (x1 .lt. 1.d4) then
d1 = dabs(dlog(x1-dsqrt(x1**2. - 1.d0)))
else
d2 = 2.d0*x1
d1 = dlog(d2)
endif

dacosh = d1

return
end

C--------------------------------------------- SUBROUTINE FILHND --

subroutine filhnd(itact, TotalEnergy, n, slipbi, slip, ipint, omeacal
1 , icohflag, vphase, alphau2, alphaavo2, ilastrun, R, ihalfflag
2 , isymmetric)
implicit real*8(a-h, o-z)
common / polar/ u(-1000:1000, 0:1), v(-1000:1000, 0:1)

common / grid / dx, dt, ngridminus, ngridplus, ngrid

common / origin/ voriginrelu, uoriginrelv, jlocuinit, jlocvinit
common / filename/ name, name2, ipnum(3), puname, pvname, aname

complex*16 u, v, uu(0:2047), uf(0:2047)
character*3 suffix
character*8 name, name2, puname, pvname, aname, pure, pvri
character*10 numbers
real*8 phase1(-1000:1000), phase2(-1000:1000), rmagv(-1000:1000)

1 , rmagv (-1000:1000), angle1(-1000:1000), vangle(-1000:1000)
2 , freq(2048), amp(2048), alphau2, alphavo2, R

real*8 voriginrelu, uoriginrelv, dx, dt, aux1(5959), aux2(4096)

1 , uarea, varea, slip
2 , slipbi, omeacap, pi, pii

integer nstep, ngrid, ngridminus, ngridplus, icohflag
1 , ilastrun
data numbers/ '0123456789' /

pi = 3.14159265358979323846

79
\[ \pi = 1.0 / \pi \]

BEGIN OLD PRINTING

This is from the way we used to print:

This printout business is a mess. An example should help. If \( \text{iprint} = 20 \) and \( \text{dt} = 0.001 \), then \( 1.0 / \text{dt} = 1000 \), \( \text{int}(1.0 / \text{dt}) / 3 = 333 \), \( 333 / \text{iprint} = 16 \), so \( \text{nprint} = 320 \). The output file names will be in multiples of 16 \( \text{(320/20=16)} \). Isn't that convenient? Use the long conditional which may be presently commented out because it can be nicer to use a simpler IF.

\[ \text{nprint} = \text{iprint} * (3 * \text{int}(1.0 / \text{dt}) / \text{iprint}) \]

not sacred.

\[ \text{nprint} = \text{nprint} / 8 \]

END OF OLD PRINTING.

The time below is true iff \( \text{reducedtime} = \text{false} \). - see main program.

The point is, we don't usually print during the reducedtime period.

\[ \text{time} = \text{float} \left( \text{itact} \right) * \text{dt} \]

"pd" is the ordinary oscillatory phase of an \( N=1 \). The code was written originally so that an \( N=1 \) of unit amp. was in channel one. In fact, there was only one pd.

I discovered this when propagating a .5 amp. \( N=1 \).

\[ \alpha = 2 \text{ etal}. \ N=1 \text{ phase goes as } \exp \left( 2 \text{ etal} \times \text{etal} \ t \right). \]

Recall that for whatever reason, the phase range is \( (0,2) \), so we must divide by \( \pi \).

It is important to recognize that this factor of \( \text{pd#} \) is actually intended for an interferometric application. It is the cyclic phase change of an equal pulse propagating alone in a fiber: i.e., it is the reference arm phase variation.

\[ \text{pd1} = \text{time1} \times 0.5 \alpha \times \alpha \pi \]
\[ \text{pd2} = \text{time1} \times 0.5 \alpha \times \alpha \pi \]

if\( \{ (\text{time1} .lt. 0.) \) or \( \{ \text{mod} \left( \text{itact}, \text{nprint} \right) \} \).ne.0) \} goto 143

if\( \{ (\text{time1} .lt. 0.) \) or \( \{ \text{mod} \left( \text{itact}, 1280 \right) \} \).ne.0) \} goto 143

\[ \text{if} \{ \text{ilastrun} \).eq.1 \} \) goto 581

\[ \{ \text{if} \{ \text{ipnum} \}.eq.0 \} \) \) and \( \{ \text{name} \}.eq.'000' \} \) and.

For slip less than 1, comment out the 1 extension below.

\[ 1 \{ \text{ipnum} \}.eq.5 \} \) and.

\[ 2 \{ \text{name} \}.eq.'005' \} \) goto 143

18 = u mag + phase, 3 = v same, 2 = FFT u

581 open(unit=18, file=\text{name})

if \( \{ \text{ilastrun} \).eq.1 \) goto 584

if \( \{ \text{mod} \left( \text{ipnum}(1) + \text{ipnum}(2) + \text{ipnum}(3) \right), 3 \} \).ne.0) goto 582

584 continue

if \( \{ \text{isymmetric} \).eq.0 \} \) open(unit=3, file=\text{pdbname})

open(unit=2, file=\text{purname})

80
if (ilastrun.eq.1) then
  puri=name
  pvri=pvname
  puri(6:7)='RI'
  pvri(6:7)='RI'
  open(unit=21,file=puri)
  open(unit=22,file=pvri)
  write(21,140)'URI',dt,omegacap,slip,slipbi,R,voriginre
  ,time1,ichflag,TotalEnergy,ngrid,vphase,ihalfflag
  if (isymmetric.eq.0)
  1 write(22,140)'VRI',dt,omegacap,slip,slipbi,R,vorin
  1 ,time1,ichflag,TotalEnergy,ngrid,vphase,ihalfflag
endif

write(18,140)'U ',dt,omegacap,slip,slipbi,R,vorin
  ,time1,ichflag,TotalEnergy,ngrid,vphase,ihalfflag
if (ilastrun.eq.1) goto 585

print 'V and FFT only every thirtieth output filename update: if (mod((ipnum(1)+ipnum(2)+ipnum(3)),3).ne.0) goto 583
write(2,140)'UFT',dt,omegacap,slip,slipbi,R,vorin
  ,time1,ichflag,TotalEnergy,ngrid,vphase,ihalfflag
if (isymmetric.eq.0)
  1 write(3,140)'V ',dt,omegacap,slip,slipbi,R,vorin
  1 ,time1,ichflag,TotalEnergy,ngrid,vphase,ihalfflag

-----------------------------------------------------------------
583 rmagumax=0.
rmagvmax=0.

c*COW PREFER VECTOR
cc do 92 jj=-ngridminus,ngridplus,nstep
    uarea=0.d0
  92 varea=0.d0
    nstep=ngrid/999+1
ccccc cow assume count (128)
C*COW PREFER VECTOR
    do 7 j=ngridminus,ngridplus,nstep
        rmagu(j)=cdabs(u(j,n))
    c
        rmagv(j)=cdabs(v(j,n))
    c
        uarea=uarea+rmagu(j)
    c
        varea=varea+rmagv(j)
    c
    c Note the ESSL routine locmax=idmax(#elems in vector,vector,stride)
    c rmagumax=rmagu(locmax)+v
C------------------------------------------------------------------
247 angle(j)=atan2(dimag(u(j,n)),dreal(u(j,n)))*pii
    c
    7 continue

q=0.d0
    do 76 j=ngridminus,ngridplus,nstep
cc    q=int(float(itime+6)*.08333333333333333)


```c
if (isymmetric.eq.1) goto 767

C* COW PREFER VECTOR
    do 74 j=ngridminus,ngridplus,nstep
        rmagv(j)=dabs(v(j,n))
        varea=varea+rmagv(j)
    c Note the ESSL routine locmax=Idmax(#elems in vector,vector,stride)
    c rmagumax=rmagu(locmax)+ v

74   vangle(j)=atan2(dimag(v(j,n)),dreal(v(j,n)))*pi2

    do 75 j=ngridminus,ngridplus,nstep
        phase2(j)=vangle(j)-pd2+80.d0
    c

767   continue

    do 78 j=ngridminus,ngridplus,nstep
        if (phase1(j).lt.0.d0) phase1(j)=phase1(j)+1
        if (phase2(j).lt.0.d0) phase2(j)=phase2(j)+1
        phase1(j)=mod(phase1(j),2.d0)
        phase2(j)=mod(phase2(j),2.d0)

78   xu=float(j)*dx
    if (ilastrun.eq.1)
      1 write(21,142) xu,dreal(u(j,n)),dimag(u(j,n))
    c Below I artificially set the phase to zero if the magnitude c is less than a certain value. This cleans up plots.
        if (rmagu(j).lt.1.d-3) phase1(j)=0.d0
    8 write(18,142) xu,rmagu(j),phase1(j)

    if (ilastrun.eq.1) goto 777
    if (mod((ipnum(1)+ipnum(2)+ipnum(3)),3).ne.0) goto 71
    if (isymmetric.eq.1) goto 71

777   do 9 j=ngridminus,ngridplus,nstep
        xv=float(j)*dx+voriginrelu
    if (ilastrun.eq.1)
      1 write(22,142) xv,dreal(v(j,n)),dimag(v(j,n))
    c Below I artificially set the phase to zero if the magnitude c is less than a certain value. This cleans up plots.
        if (rmagv(j).lt.1.d-3) phase2(j)=0.d0
    9 write(3,142) xv,rmagv(j),phase2(j)

71   close(18)
    if (ilastrun.eq.1) then
```
close(21)
close(22)
goto 787
def
if (mod((ipnum(1)+ipnum(2)+ipnum(3)),3).ne.0) goto 143
787    close(3)
c
c------------------------------------------FFT section------------------------------------------
c
cc isayso=1
cc if (isayso.eq.1) goto 143
do 18 j=0,2047
    ju=j-1023
    if (abs(ju).le.(ngrid/2)) then
        uu(j)=u(ju,n)
    else
        uu(j)=0.d0
    endif
18 continue
xmax= float(ngrid)*dx
period=2.d0*xmax*2048./2047.
fundamental=2.d0*pi/period
call dftci(1,uu,1,2048,uf,1,2048,2048,1,1,1.,aux1,5959,aux2,4096
call dftci(0,uu,1,2048,uf,1,2048,2048,1,1,1.,aux1,5959,aux2,4096
do 20 j=1251,2047,2
    freq(j)=float(j-2048)*fundamental
    amp(j)=cdabs(uf(j))
20    continue
do 19 j=0,798,2
    freq(j)=float(j)*fundamental
    amp(j)=cdabs(uf(j))
19    continue
write(2,144) freq(j),amp(j)
do 22 j=0,798,2
22 write(2,144) freq(j),amp(j)
close(2)

C-------------------------------------------------------------
C
C Update the Name of File to Store the Next Frame of Data
C
143    continue
    ipnum(1)=ipnum(1)+1
do 876 loop=2,3
    if (ipnum(loop-1) .eq. 10) then
        ipnum(loop)=ipnum(loop)+1
        ipnum(loop-1)=0
    endif
876    continue

do 871 loop=1,3
    11=ipnum(4-loop)+1
suffix(loop:loop)=numbers(11:11)
name(2:4)=suffix(1:3)
puname(2:4)=suffix(1:3)
pvname(2:4)=suffix(1:3)
c
cc
aname(2:4)=suffix(1:3) don't use this for areas !!!!!!!!

140 format(1x,a3,' dt',e8.3,' dw/w',e12.6/1x
 &   ,'s',e11.4,' b',f9.4, 'R',f7.1/1x
 2   ,'vctr',f12.7,' t',f11.7,' c',i1
 3   ;/1x,'Et',f10.5,' ng',i5,' vp',f6.4,' h',i1)
c
Below is an old format:
c 943 format(1x,'VRI dt=',e8.3,' dw/w=',e9.3/1x
 &   ,'s',f6.3,' sb',f7.4,' t',f11.7,' c',i1/1x
 c 2   ,'vctr=',f16.11,' ng=',
 c 3   i5,/1x,'Et=',f17.12,' vph=',f6.4)
c U, V, UFT, URI, VRI

142 format(1x,3(1x,e17.10))
144 format(1x,2(1x,e17.10))
return
end
Appendix B

Sample input datafile

This appendix contains an example of an input datafile for the simulation code of Appendix A. The example, which is extensively commented, should be particularly useful to the reader who wishes to understand the operation of the simulation program.
Further comments:

1. The center of the u pulse is at x=0 in the u frame. The center of the initial v pulse is (in the u frame) voriginrelu=.5*window*spacing, where 'spacing' could be negative.

2. If "slip" is positive, u is the fast channel, so "spacing" had better be positive as well. One has to be very careful in the case of incoherent pulses (pulses of different frequencies) because the total slip could be of opposite sign from "slipbi," which is the slip due to birefringence alone.
3. The source code statement, "$b(3:3)='A'$" assumes that "aname" is a two-character string. It is not necessary, but it is nice.

4. Likewise, the source code was written anticipating that "name" and "pvname" would be 5-character strings.

5. The user is given the choice of specifying the length of the program run to be either an explicit number of timesteps or in terms of the naively anticipated post-collision relative pulse displacement. The former is straightforward, but the latter clearly warrants further explanation. At the very beginning of the simulation, the center of the \( v \) pulse is displaced from the center of the \( u \) pulse by an amount "voriginrelu" (see 1. above). The source code calculates the slip, from which it can determine the expected number of timesteps for the \( v \) pulse to be displaced from the \( u \) pulse by an amount "voriginrelu." This calculation of course neglects any scattering-induced shifts, but it is a good approximation for reasonable values of slip (e.g. slip\(\geq1\)). The user can specify "factor," which, again neglecting scattering effects, would terminate the program when the \( v \) pulse is displaced from the \( u \) pulse by an amount \"factor\*voriginrelu\". Choosing factor\(=0\) or less would probably not be terribly useful, since factor\(=0\) corresponds roughly to mid-collision. Choosing factor\(>1\) is perfectly reasonable.

6. Choose "beta" > 0 for anomalous dispersion, < 0 for normal.

iboundary - 1=exponential tails out of bounds
          2=values outside=values at boundaries
          3=zero outside

I always use 3, and it is generally quite suitable.

With collisions, when strange profiles result, one cannot easily determine reflectionless boundary conditions

isymmetric - a flag which allows one to cut the execution time almost in half. If the pulses in the two channels are identical, and if the coherence terms are set to zero (see icohere below), then we need not calculate \( v \) at each spatial increment, for it is simply \( "u" \) flipped in space about the center-of-mass.

Certain aspects of parameter tracking may not work properly for isymmetric\(=1\).

ngrid - the number of x gridpts. NOTE that this number cannot exceed the max dimension in array \( u \), i.e.
        if \( u \) is dimensioned \( u(-1000:1000,0:1) \), don't try to set ngrid=2002 or more. If you change the dimensions of \( u \) and \( v \), make sure you change them in the subroutine fihnd as well.

dt - use around .005 for no coherence, slip between 1 and 10.

You must balance this with an appropriate \( dx \), determined by ngrid and window. I like ngrid=530,window=100.

For \( s=100 \), I've used \( dt=.0005 \), ngrid=1980, window=100.

dtmax is traditionally chosen to be 5\( dt \) - see source code for explanation of the distinction.

spacing: The center of the \( v \) pulse is initially at
voriginrelu (see 1. above), a function of spacing.

window: In actual x units, is the truncated region over which we solve the PDE. We cannot use the correct -\infty to +\infty.

tau actual pulsewidth in engineering units, not normalized

zindex "mean" index of refraction in the fiber, again in real units.

zlambda "mean" wavelength in real units.

disp a dimensionless measure of dispersion, see Menyuk.

iasymm the v-profile is sech tanh instead of sech if iasymm=1.

omegacap (frequency difference of pulses in u and v channels) divided by (wbar).

alphau2, ... The initial u profile is alphau2 sech (alphau x)

with the v profile analogous and shifted by voriginrelu.

vphase rather than begin with u and v sech's of zero phase, vphase allows one to put some other phase on the v pulse.

beta is twice the coefficient of the second derivative (the GVD=group velocity dispersion) term, in normalized units. It is positive for anomalous dispersion.

coherence flag must be real*8 - it is converted to an integer flag, but it 1. convenient to introduce it as real.

If it is 0.0, coherence terms are neglected.

If it is ANYTHING ELSE, coherence effects are included.

tau0 is an x scale needed when Raman effects are included.

tauun Raman relaxation time constant, real units.

rlambda is very important. It is the coupling coefficient for the equations. It indicates the strength of the cross phase modulation. Use 2/3 for linear polarization or 2 for circular.

slipbi normalized units. Intrinsic birefringence.

The source code calculates the actual slip, including that due to the frequency difference between the two pulses.

ihalfflag Setting ihalfflag=1 tells the code to ignore the relative motion of the u and v pulse envelopes. This allows us to take advantage of the symmetry in the problem to cut the execution time by around 40%. Only use this if you are certain that the relative envelope motion is truly negligible.

iruntime see 5. above. It is very convenient to specify the length of a simulation via "factor," unless the pulse envelopes copropagate.

factor see 5. above.

itmax you only need to specify this if you choose iruntime=0.

filter this isn't calculated the same way factor is. Recall factor measures from the naively predicted mid-collision. filter is the fraction of the TOTAL SIMULATION t at which the v channel disappears.

As an added surprise, you can use a negative value of filter. The absolute value of filter is then used, but the fact that it is negative means that instead of
filtering the v pulse, the medium jumps from one
dispersion regime to the other. That is, the coefficient
of the second derivative term changes sign.

ipint see source for details, but a large ipint means that you
won't get magnitude/phase and F.T. datafiles very often.
I usually use from 10 to 80. If you only care about
the cumulative files, and want only the initial and
final profiles, you can set ipint very large.

itrack =1 if you want output files giving the evolution of the
"soliton parameters" for the solitary wave in channel 1,
the u pulse. It isn't a true soliton, but this allows
a fair comparison with perturbation theory, as in my
thesis. See CZMSI2N FORTRAN for the perturbation
software.

rm God only knows (well maybe Sumanth or Randa knows)
name Name of u magnitude/phase output files, where the 2nd,
3rd, and 4th characters are arbitrary - they get replaced
by numbers in subroutine fillnd of the source code.
I have no idea why a suffix of 'D' was chosen for channel 1
model another relic. I always use 2. It looks like 1 was
intended for single pulse propagation. It might save
a nanosecond or so of run time to use it for copropagation
but I haven't bothered to check.
puname as name, but for FFT file.
pyname same but for v pulse.
aname for the energy tracking output file.
mode2 I always use 1 and have sech input pulses Setting =2
allows one to read the profile(s?) from a datafile.
I've never used 2.

name2 For mode2=2, filename.
Bibliography


induced by cross-phase modulation,” Optics Letters, vol. 13, pp. 871–873, 
October 1988.