Noname manuscript No.
(will be inserted by the editor)

An All–optical Soliton FFT Computational Arrangement in the 3NLSE–domain

Anastasios G. Bakaoukas

Received: date / Accepted: date

Abstract In this paper an all–optical soliton method for calculating the FFT (Fast Fourier Transform) algorithm is presented. The method comes as an extension of the calculation methods (soliton gates) as they become possible in the Cubic Non–linear Schrödinger Equation (3NLSE) domain, and provides a further proof of the computational abilities of the scheme. The method involves collisions entirely between first order solitons in optical fibers whose propagation evolution is described by the Cubic Non–linear Schrödinger Equation. The main building block of the arrangement is the half–adder processor. Expanding around the half–adder processor, the "Butterfly" calculation process is demonstrated using first order solitons, leading eventually to the realisation of an equivalent to a full Radix–2 FFT calculation algorithm.

Keywords Solitons, 3NLSE domain, All–optical FFT, Cubic Non–linear Schrödinger Equation, Soliton collisions, Soliton computational schemes.

1 Introduction

Computational systems based on soliton collisions for transferring and processing data continues to be a topic which stands at the forefront of scientific research (Jakubowski, Steiglitz, & Squier, 2001), (Jakubowski, Steiglitz, & Squier, 1996), (Bakaoukas & Edwards, 2009b), (Jakubowski, Steiglitz, & Squier, 1997), (Steiglitz, 2000).

Within this framework, an earlier version of this paper originally appeared in (Bakaoukas, 2016). The current paper has augmented this original work by including an extensive discussion of the optical solitons background theory as presented in the remainder of this section, a full explanation of all the basic concepts and parameters involved in the formulation of the computational system proposed for

Anastasios G. Bakaoukas
Computing & Immersive Technologies Department,
University of Northampton, St. Georges Avenue, Northampton, NN2 6JB, UK.
E-mail: Anastasios.Bakaoukas@northampton.ac.uk

the 3NLSE–domain (section 2), and a detailed analysis of all the numerical methods currently available for the simulation of optical solitons propagation down an optical fibre when the propagation parameters applicable to the system are those of the 3NLSE–domain (section 3). The Half–adder Processor Scheme proposed and discussed in the remainder of the paper, as well as individual soliton arrangements in the overall computational system, have been extensively tested and successfully verified using all these numerical methods.

Generally there are two different types of solitons, delineating the areas of general interest to create computational systems from. The, so called, "Spatial Solitons" and the "Temporal Solitons", respectively defining the "Spatial Solitons Computational Systems" and the "Temporal Solitons Computational Systems". Solitons owe their existence to the physical alignment between a phenomenon known as "Kerr Non–linearity" (Self–phase Modulation) and the phenomenon of "Chromatic Dispersion" in optical fibres, which is the primary material allowing for solitons generation and propagation.

Temporal Solitons in optical fibres can be described very accurately by the "Cubic Non–linear Schrödinger Equation" or "3NLS Equation" for sort, while Spatial Solitons can be described very accurately by the "General NLS Equation" which, although describing a system which is not generally integrable, never the less enables us to calculate more accurately more complex phenomena. Solitons in both the Temporal and the Spatial domain can either "Interact" with each other (Soliton Interactions) or "Collide" with each other (Soliton Collisions).

Soliton systems which are using fast digital logic gates and carry and process information through soliton collisions have been proposed for some time now (Jakubowski et al., 2001), (Jakubowski et al., 1996), (Bakaoukas & Edwards, 2009a), (Jakubowski et al., 1997), (Steiglitz, 2000), (Blair, 1998). The logic gates arrangements used by these systems take advantage of the phase difference or the frequency difference emerging after each collision between orthogonally polarised solitons to represent a bit as we know it from classical computational theory. Other systems of logic gates arrangements have also been proposed, which make use of the time difference ("Soliton Trapping") or of the position difference ("Soliton Dragging") emerging between the solitons involved after they have clash. Particle machines, which perform calculations using soliton collisions, have been also proposed (Steiglitz, 2000) and extensive theoretical research on finding a universal computer system in all these domains which to conform to the standard Turing system ("Computationally Universal System") is currently underway. The goal here is to create a global computational system, which will generally use logic gates of the type initially introduced by Toffoli in (Toffoli, 1980) or alternatively of the type initially introduced by Fredkin in (Fredkin & Toffoli, 1981) for processing and storing data very close to the classical.

Most of the major studies been carried out so far in the direction of computing with solitons are mostly at a purely theoretical level, especially considering collisions between first order solitons. Independent and more or less equally effective numerical methods like: the "Finite Difference Runge–Kutta Technique" (FDRKT), the "Split–Step Fourier Transform" (SSFT), the "Fourier Series Analysis Technique" (FSAT), and the " Fuzzy Mesh Analysis Technique" (FMAT) have been successfully applied to provide simulations of solitons propagation down the optical fibre in the 3NLS Equation domain and in general have been extensively used in theoretical research on colliding solitons. These methods give the possibility not only of theoretical analysis and simulation of collisions between solitons but also of monitoring the stability and the general dynamics of Non–integrable and Non–linear models alike.

There is a number of studies in which the use of solitonic optical pulses for the purposes of carrying out computations has been investigated (Jakubowski et al., 2001), (Jakubowski et al., 1996), (Bakaoukas & Edwards, 2009a), (Blair, 1998). In this present paper only temporal solitons (involving a balance between Kerr type Non–linearities and Dispersive effects in glass fibres) are concerned. At this early point the fact that the interactions between solitons of this type can be a relatively long–range phenomenon need to be emphasised, because the Kerr Non–linearity is a relatively weak effect.

In what follows in this introduction section, the discussion is focusing on the 3NLS Equation domain. For a more extensive and thorough discussion the reader is referred to (Bakaoukas & Edwards, 2009b), (Bakaoukas, 2013), (Bakaoukas & Edwards, 2013), (Bakaoukas, 2014) and (Bakaoukas, 2015) where the application of first order and second order solitons, following the Toffoli gates prototype as well as others, has been presented and verified regarding their computational abilities in terms of logic gates formations.

A positive value for "Dispersion" parameter describes the formation of bright optical solitons whilst a negative value leads to the formation of dark solitons. The 3NLS Equation in general, describes a modulated wave packet propagating through a non–linear dispersive medium with a constant velocity. For certain initial pulse shapes (the "Reflectionless Potentials"), the 3NLS Equation is completely integrable and the evolution of a soliton can be found in closed form by means of the Inverse Scattering Transform (IST) (Ablowitz & Segur, Philadephia, 1981). Solitons arising out of a balance between Dispersive and Kerr Non–linearity effects possess dominant characteristic features one of which is the elastic collisions between them. Solutions described by Non–integrable Non–linear Wave Equations on the other hand are usually referred to as "Solitary Waves" and collisions between solitary waves are inelastic and more complex in character. A solution of the integrable 3NLS Equation applicable to pulse propagation in optical fibres is the Hyperbolic Secant where an arbitrary positive number representing the soliton order, the distance along the fibre, and time, all in normalised dimensionless units, are the main parameters forming the initial soliton propagation envelope. By coupling pulses in and out of a fibre at appropriate points (of distance and time), useful computation could be possible based on collisions between solitons within the fibre.

The material presented in (Bakaoukas & Edwards, 2009b), in particular, shows that in situations where optical solitons are formed within optical fibres (simulations have been carried out using all the above mentioned numerical techniques), with appropriate practical arrangements, computationally universal systems based on collisions between first order solitons are possible using logical gates based on the "Controlled" type of gates originally proposed by Toffoli and Fredkin (Toffoli, 1980), (Fredkin & Toffoli, 1981). As an extension to what presented in the above mentioned papers, in this present paper, the numerical study of collisions between first order solitons is expanded leading towards an all–optical FFT (Fast Fourier Transform) calculation. The CN and CCN soliton gates continue to be the essential ingredient of the computational model.

2 Soliton Collisions And Computational Scheme In The 3NLS Equation Domain

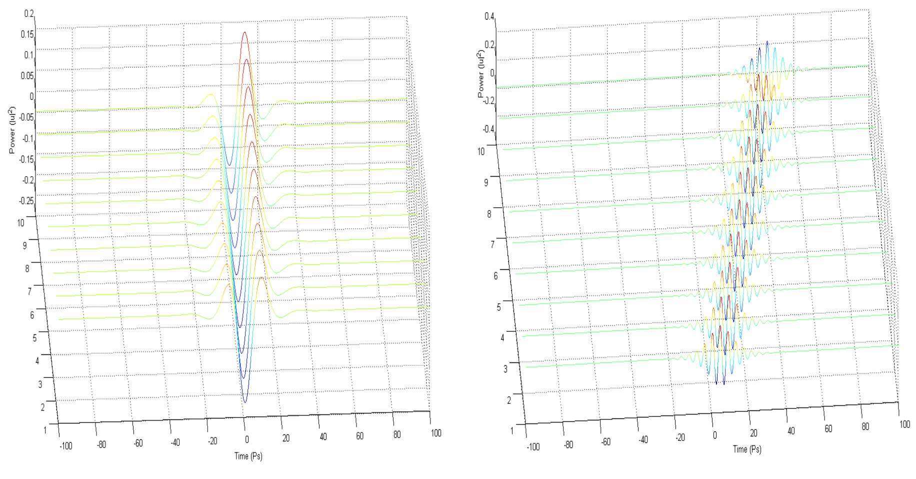
To be able to present and analyse the properties and the basic features of a soliton computational system in a domain described by the 3NLS Equation (1), we need to present first the basic requirements for computation, which include: cascadability, fanout, and Boolean completeness. In general terms, cascadability requires that the output of one device can serve as input to another; fanout refers to the ability of a logic gate to drive at least two similar gates; and Boolean completeness makes it possible to perform arbitrary computation. The 3NLS Equation domain system can be characterised as an “oblivious” system (one that is governed by totally elastic collisions). As we are about to see in what follows, “oblivious” soliton systems under certain conditions, can perform useful computations by a direct simulation of Toffoli logic gates.

$$\frac{\partial u}{\partial z} = -\frac{j}{2} sgn(\beta_2)\frac{\partial^2 u}{\partial T^2} + B\frac{\partial^3 u}{\partial T^3} + j\gamma|u^2|u - \Gamma u \tag{1}$$

Mathematically, a couple of solitonic pulses, which commences propagating down an optical fibre and possesses all the properties required to end up in a collision, can be described by the following equation:

$$\begin{aligned} u(0,\tau) = {} & r\ sech(r(\tau - q_0))e^{j\theta}e^{jv\tau} + \\ & + r\ sech(r(\tau + q_0))e^{j\theta}e^{jv\tau} \end{aligned} \tag{2}$$

where, r represents the amplitude of the solitons, θ is the relative phase value, and q_0 is the initial displacement between the two solitonic pulses.

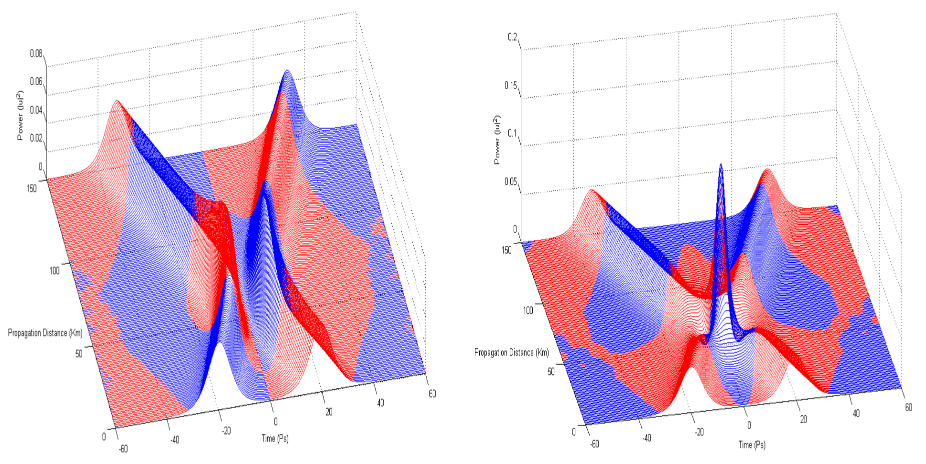


(a) (b)

Fig. 1 (a) Solitonic pulse carrier (the soliton velocity is equal to 0.3) and, (b) Solitonic pulse carrier (the soliton velocity is equal to 1.5).

By solving the basic soliton equation using the IST we come to know that the speed of a solitonic pulse depends on the frequency of the modulated carrier. So if we vary the frequency of the carrier we simultaneously alter the speed of the solitonic pulse itself. The second exponential term in (2) does just that; alters the frequency of the carrier by a rate corresponding to the value given to its variable, and simultaneously alters that way the speed of the pulse itself by the desired amount (Fig. 1).

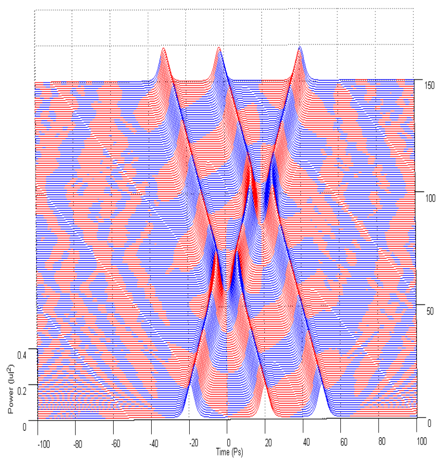
For the purposes of this paper only collision situations between two and three solitons (Fig. 2 & Fig. 3) will be discussed and presented in details, since these are the fundamental building blocks out of which the 3NLS Equation domain computational system is consisting of (interactions between solitons have been proved unable to offer computationally useful properties and soliton arrangements (Bakaoukas, 2014)). In this scheme, two situations can be distinguished: a) the solitons collide and are in phase, or b) the solitons collide and are out of phase. In physical terms this can be translated as between the two solitons emerging an attractive or a repulsive force, respectively (Fig. 2). The presence and the strength of the repulsive or attractive force depends on the relative phase values of the two pulses.



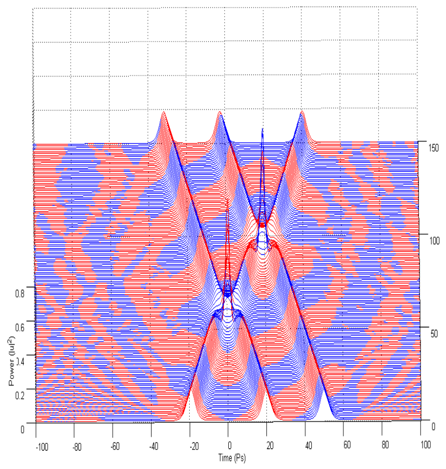
(a) (b)

Fig. 2 Collision between two first order solitons: (a) the solitons are out of phase and the repulsive force causes them to change their propagation direction (the soliton speeds are +0.3 and -0.3) and, (b) the solitons are in phase and the attractive force causes them to pass through each other maintaining their propagation directions (the soliton speeds are again +0.3 and -0.3). Maximum magnitude for the soliton envelope exactly at the point of collision is typical for this type of collision.

As a first step into starting describing the fundamentals of the computational system itself now, the encoding rules for representing bits into the computation system will be set by initially admitting the existence of only two solitons in it. One soliton with a phase value of π and one soliton with a phase value of 0. This



(a)



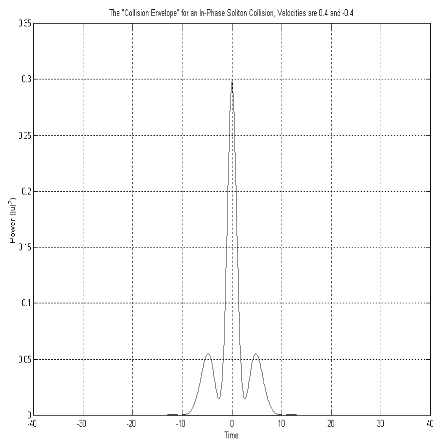
(b)

Fig. 3 Collision between three first order solitons: (a) the first and the third soliton possess the same phase value, while the second soliton has a phase difference of π in relation to the other two and, (b) all three solitons possess the same phase value.

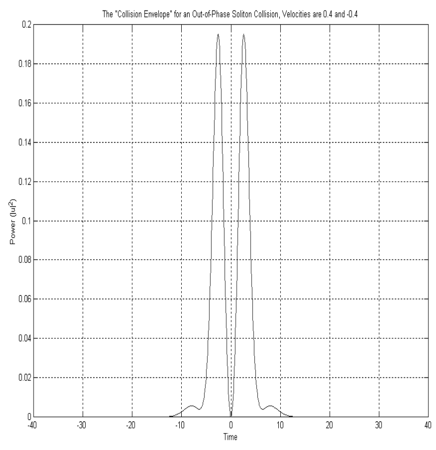
way there can only strictly exist two types of collisions between solitons in the computational system: a) two solitons collide and are in phase or, b) two solitons collide and are out of phase (Fig. 2). Now the solitons can be directly used as input values to a solitons defined type of logic gate. The most important fact of all is that these two types of collisions inherently possess the property of sequencing, so they can be cascaded.

By using these rules things can be stretched a bit further and into safely considering collisions between solitons as the inner process of a solitons defined logic gate, with the initial solitons formulating the input to the logic gate values and the two solitons recovered with their original state intact after the collision, as formulating the output values of the logic gate. So, basically, we split the whole process of a collision into two important parts: a) the logic gate length, bounded between the point at which solitons begin to propagate through the medium and that at which the two solitons emerge intact from the collision with time positions in reverse order, and b) the point at which the two solitons collide, creating a characteristic for their phase values "Collision Envelope" (Fig. 3 & Fig. 4). Introducing a third soliton into the arrangement, to which we refer to as "Time–Gated" soliton (Fig. 5) and is a generated by the system soliton with a phase value also determined accordingly by it. Effectively what is achieved here is within a system using three instead of two solitons per collision (logic gate), and by using appropriate combinations of phase values, the input solitons to be converted to Control Specific solitons and Data Specific solitons, achieving performing that way useful calculations.

Let us now see how we can use this three solitons arrangement and its properties as a full computational element (logic gate). In order to achieve this we assign to each soliton participating in the arrangement either the binary value 0 or the

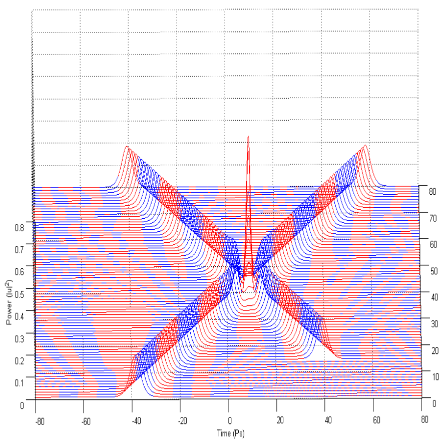


(a) The collision envelope for two solitons in phase.

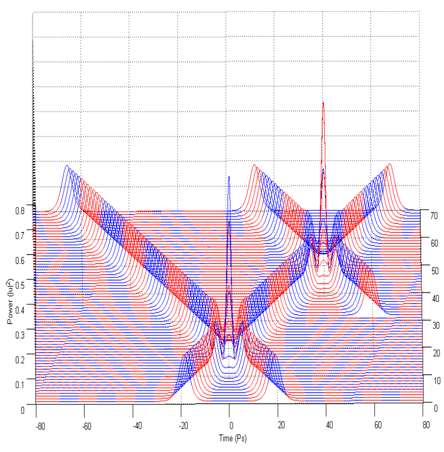


(b) The collision envelope for two solitons out of phase.

Fig. 4 The collision envelope for two solitons in and out of phase.



(a)



(b)

Fig. 5 (a) A collision between two solitons. The second soliton is a “Time–Gated” input soliton and, (b) Collision between three solitons. The third soliton taking part in the collision is a “Time–Gated” soliton in phase with the initial two.

binary value 1, the way previously explained. The next step is to assume that the third soliton is a “Time–Gated” soliton with a phase value determined accordingly by the system as the process progresses. This practically means that the “Time–Gated” soliton does not take part in the calculation process from the start but is created and begins to propagate down the fibre at another specific point in time as a result of a decision (command) initiated by the system. The right point in time for this “Time–Gated” soliton to start propagating is defined as the one immediately following the collision between the two initial to the logic gate arrangement solitons. This forces the third ("Time–Gated") soliton to play the role of the "Controlled" by the system soliton. Within the boundaries of the computational system presented here, the phase value of the created ("Time–Gated") soliton and its propagation timing are decided dynamically by the system after certain conditions have been taken into consideration, which in our case are the phase values of the two initial solitons. Now the arrangement can successfully simulate the CN and CCN Toffoli logic gates.

In setting the rules under which the system will be able to distinguish between conditions calling for the generation of a "Time–Gated" soliton with a phase value of 0 and those calling for the generation of a "Time–Gated" soliton with a phase value of pi, we assign a Boolean value to each one of the two envelope types resulting after a collision has taken place, and located at exactly the midpoint of the whole collision length (Fig. 4). The collision envelope occurring when two solitons collide and are in phase, is assigned the binary value 0, while the collision envelope occurring when two solitons collide and are out of phase is assigned the binary value 1. Thus, a system able to "read" the collision envelope within the mathematically determined collision point of the two initial solitons (input solitons to the logic gate arrangement), is also capable of recognising the nature of the collision, i.e. to recognise immediately whether the two solitons participating in the collision were in phase or out of phase. Finally, using these assumptions, in conjunction with the "Time–Gated" solitons concept, we can successfully simulate the operation of a CN Toffoli logic gate in the 3NLS Equation domain and build a truth table for this type of gates (Bakaoukas & Edwards, 2009a).

3 Numerical Methods For Soliton Propagation Simulation

In this section, the numerical techniques used for obtaining the simulation results are briefly described and example outputs are presented of using them. The recognition that the mathematical complexity of soliton solutions arises only because of the dependency of the Refractive Index n on spatially varying intensity could eradicate the complexity if, instead, n could be defined as depended not on a spatially varying intensity but on the total beam power. This can be achieved in a heuristic model by assuming that the medium has a non–local response with a correlation length much larger than the beam diameter. The non–linear wave equation then becomes linear and readily solvable, but the solution for solitons still contains the essential characteristic features. Physically, the model transforms the problem into a simple case of linear propagation of thin beams in a waveguide. With the assumption that the beams always stay close to the axis, the refractive index makes the wave equation identical to the Time–Dependent Schrödinger Equation (TDSE) for a linear harmonic oscillator, the solution of which is well known to all physicists. The physics of solitons can then be readily appreciated.

3.1 Finite Difference Runge–Kutta Technique (FDRKT)

Despite the complicated mathematical analysis and the advanced mathematical techniques (such as IST) someone can employ numerical techniques as well in order to analytically obtain solutions to the 3NLS Equation. The equation itself is but a Partial Differential Equation, which, by applying appropriate initial and boundary conditions can be solved by one of the available numerical techniques. The most popular, mainly because of the high accuracy of the Runge–Kutta Technique involved, is the Finite Difference Runge–Kutta Technique. To apply the Finite Difference Runge–Kutta Technique we need first to express the derivatives as a set of values representative of the continuous function:

$$\left\{ \begin{array}{l} f(x+\Delta x) = f(x) + \frac{\Delta x}{1!}\frac{\partial f(x)}{\partial x} + \frac{\Delta x^2}{2!}\frac{\partial^2 f(x)}{\partial x2} \\ f(x-\Delta x) = f(x) - \frac{\Delta x}{1!}\frac{\partial f(x)}{\partial x} + \frac{\Delta x^2}{2!}\frac{\partial^2 f(x)}{\partial x2} \end{array} \right\} \Rightarrow$$

$$\Rightarrow \frac{\partial^2 f(x)}{\partial x^2} = \frac{f(x+\Delta x) - 2f(x) + f(x-\Delta x)}{\Delta x^2} \tag{3}$$

In the Cubic Non–Linear Schrödinger Equation we substitute the second–order Dispersion driven second derivative (and/or the third–order Dispersion driven third derivative) with its Finite Difference equivalent and we obtain a form of the equation directly solvable by means of the Runge–Kutta Technique (Fig.6).

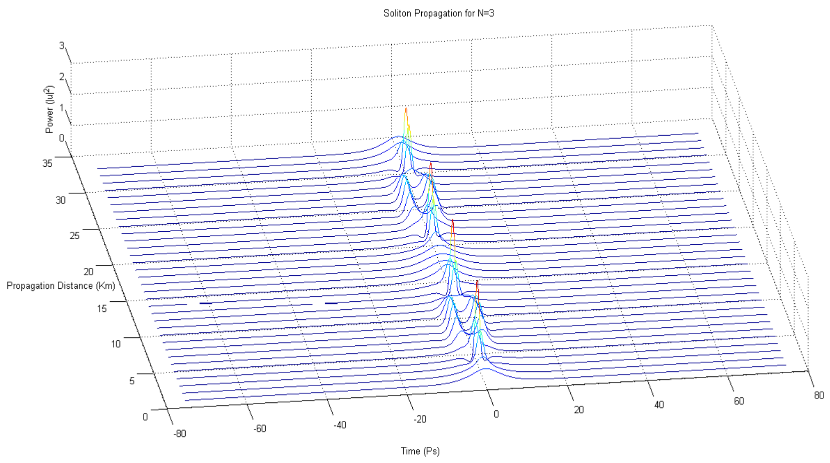


Fig. 6 Third–order soliton propagating down a length of fibre (Finite Difference Runge–Kutta Method).

3.2 Split–Step Fourier Transform (SSFT)

The Split–Step Fourier Transform (SSFT) is considered, well up to these days, the technique of choice for solving the 3NLS Equation and simulating the propagation of optical solutions for a variety of physical parameters and conditions. However, needs to be emphasised that the SSFT Method will not work well for simulating situations where there exists a forward and backward propagating wave.

A closer look in (1) reveals that the Dispersion and the Non–linearity are decomposed. Taking advantage of this fact the 3NLS Equation can be solved relatively easily through the SSFT Method. In fact, the technique has taken its name exactly because of this separation between Dispersion and Non–linearity. The first step is to make use of the operators $\hat{D}$ and $\hat{N}$ to correspond to the dispersive and non–linear terms respectively, so (1) can take the form:

$$\frac{\partial U}{\partial z} = (\hat{D} + \hat{N})U \tag{4}$$

Assuming now that only each one of them operating we can obtain:

$$\hat{D} = U_1(z,t) =$$
$$= IFFT(exp((\frac{j}{2}\beta_2\omega^2 - \frac{j}{6}\beta_3\omega^3 - \frac{a}{2})z)\ FFT(U(0,t))) \tag{5}$$

$$\hat{N} = U(z,t) = U_1(z,t)exp(j\gamma P_0|U_1(z,t)|^2 z) \tag{6}$$

At this point we need to note that the $\hat{N}$ operator multiplies the field solution and is a function of the solution $U(z,t)$, while the operator $\hat{D}$ is a differential operator expressed in terms of time derivatives that operate on $U(z,t)$. In order to make the computation more efficient the calculation of $\hat{D}$ is performed in the frequency domain with the result of transforming the derivatives in the time domain to a simple multiplication in the frequency domain.

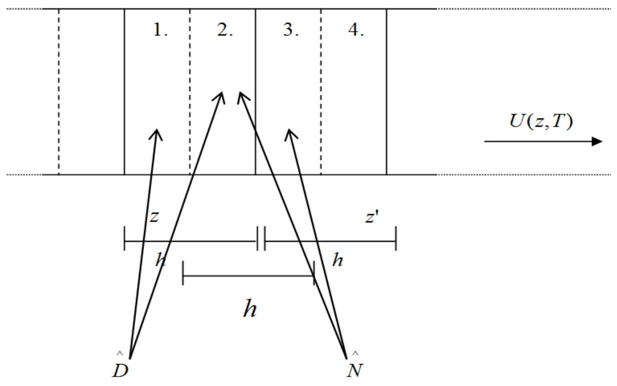


Fig. 7 The SSFT Method for one iteration (the length of every step is h).

The SSFT Method is an iterative method that determines the field solution for special steps of h. This is performed in a step–by–step approach and lasts for the entire length of the fibre. A length of optical fibre L is broken down into $S_L = \frac{L}{h}$ steps of length h (Fig. 7). So, applying the SSFT Method requires the following iterative procedure:

1. The total fibre length is divided into a number of segments of length h.
2. At the beginning of each of them we compute the FFT of the every time initial condition $U(0,t)$.
3. The pulse is propagated in the frequency domain for a distance $\frac{h}{2}$ under the effect of Dispersion only.

4. At the middle of the segment, the $IFFT$ is applied in order to come back to the time domain and calculate the contribution of Non–Linearity (and if desired fibre loss) in the whole segment.
5. In the last step another FFT is evaluated to return to the frequency domain and propagate the field through the remaining distance $\frac{h}{2}$, with Dispersion only.
6. The results obtained are used as the initial condition for the following segment and the process is repeated until the total fibre distance is achieved.

The initial condition or pulse shape is necessary to start the calculation. After propagation over a distance of h in both the linear and non–linear parts, the results can be used as the initial condition for a further propagation distance of h. This process is repeated until the required overall propagation distance is achieved.

Considering the initial fundamental optical fibre parameters: $h = 0.01Km$, $\beta_2 = -3\frac{ps^2}{Km}$, $\gamma = 2\frac{1}{\frac{W}{Km}}$, $Full\ Soliton\ Width\ (FSW) = 10ps$, $Soliton\ order = 1$, soliton propagation down the fibre using the SSFT Method is achieved (Fig. 8).

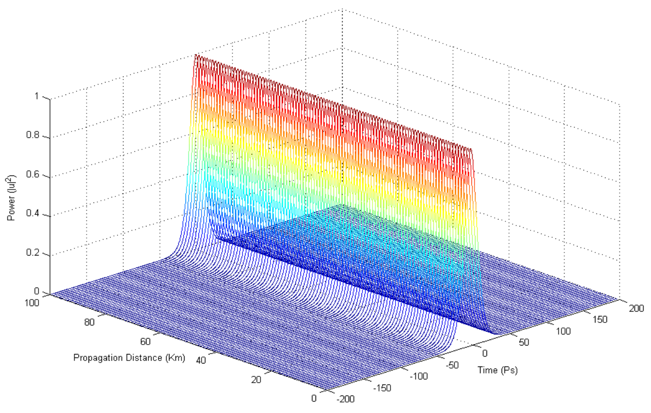


Fig. 8 A first–order soliton pulse in the 3NLS Equation domain. The Chromatic Dispersion effect blends perfectly with the Kerr Non–linearity effect and the result is the creation and propagation of a solitonic pulse. For simulating the soliton propagation the SSFT Method was used.

As a final remark the FFT algorithm imposes restrictions on the sample array format of a nature that cannot be ignored. The sample array of $U(z,t)$ for each value of z must have $N = 2^n$ points as required by the FFT algorithm. The initial array $U(0,t)$ must sample the initial pulse Δt with adequate temporal resolution and be temporally wide enough to prevent aliasing and wrapping errors. Of course, as is the case for all similar situations the sampling rate is to be given by the Nyquist Theorem.

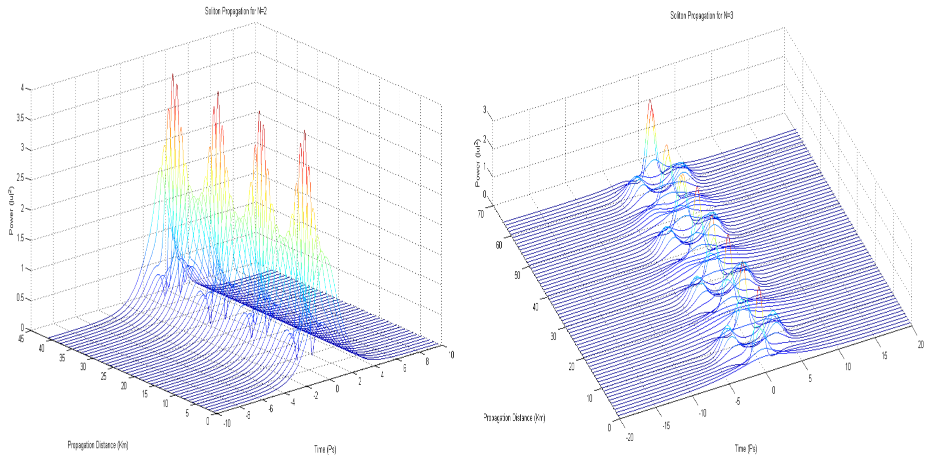
In contrast to what said above in regards to Δt, there is no strict mathematical restriction on the step–size h for the $SSFT$ Method other than choosing a very small h will result in a very accurate but very computationally time demanding simulation. Also, we need to have in mind that choosing h smaller than the carrier wavelength λ_0 is physically meaningless. On the other hand choosing h to be too large does not conserve spectral energy. The maximum choice of the step–size depends on the specific dispersive and non–linear properties.

3.3 Fourier Series Analysis Technique (FSAT)

One of the commonly used set of techniques for solving Partial Differential Equation (PDE) systems is the "Separation of Variables" set, of which the "Eigenfunction Expansion" technique is the most representative example, in which a solution of the following form is assumed:

$$u(x,t) = \sum_{n=1}^{\infty} a_n(t)\phi_n(x) \tag{7}$$

where, $\phi_n(x)$ are an orthogonal set of eigenfunctions and we have assumed that the PDE is now described by a scalar quantity $u(x,t)$. The $\phi_n(x)$ can be any orthogonal set of functions in which δ_{jk} is the Dirac function and $(\phi_j, \phi_k) = \int \phi_j \phi_k^* \partial x$ gives the inner product.



(a) A second–order soliton propagating down a length of fibre (FSAT using the MATLAB^{TM} ODE45() function).

(b) Third–order soliton propagating down a length of fibre (FSAT using the Runge–Kutta Method).

Fig. 9 Second–order and Third-order solitons as simulated using the FSAT numerical method.

Depending on the physical problem at hand, the two most common eigenfunctions used are the Fourier series and the Chebyshev polynomials, mainly because of their good correlation with the majority of physical phenomena, their spectral accuracy properties, and computational speed.

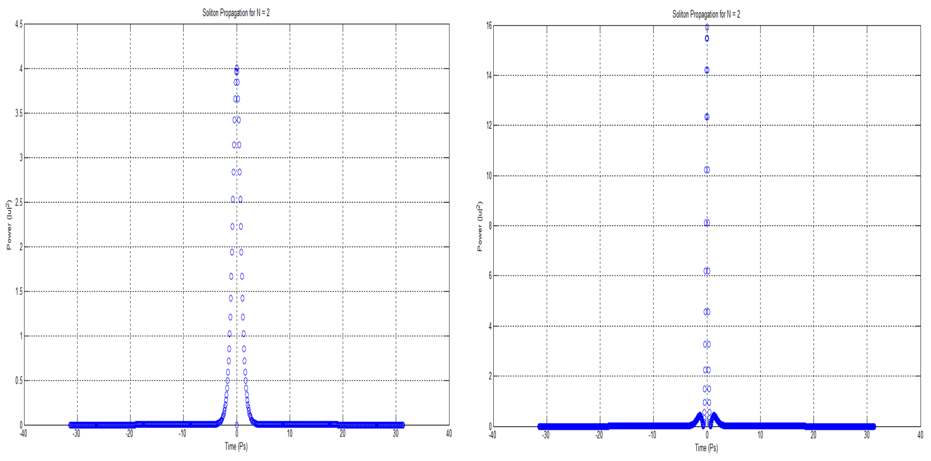
In order to solve the 3NLS Equation computationally, a Fourier Series expansion is used, thus, use can be made of the standard FFT technique. The Fourier Series Analysis Technique initially expresses the pulse envelope function in terms of a Fourier series in which $\hat{u}_n(x)$ is the Fourier amplitude coefficient, and ϵ the fundamental frequency. This procedure yields:

$$\frac{\partial \hat{u}_n(x)}{\partial x} = [-j\sigma(n) - \Gamma]\hat{u}_n(x) + jN^2 \sum\nolimits_{\forall \mu-\nu+\lambda=n} \hat{u}_\mu(x)\hat{u}^*_\nu(x)\hat{u}_\lambda(x) \qquad (8)$$

where n is an integer $-M \leq n \leq M$ and $\sigma(n) = n^2\frac{\epsilon^2}{2} + Bn^3\epsilon^3$. The equation represents a set of $2M+1$ First-order Partial Differential Equations (FPDE) of complex variable, which can be separated into its real and imaginary parts, and solved by a Fourth-order Runge-Kutta Method. The $2M+1$ initial conditions can be obtained for $x = 0$. The final solution is in the time domain. Significant parameters of this method include the time window ΔT, within which the signal is sampled for the Fourier series representation, the step length for integration Δx, and the integer M, related to the total number of samples $(2M+1)$.

3.4 Fuzzy Mesh Analysis Technique (FMAT)

The main advantage of this technique is its ability to allow for variation of the mesh size with the shape of the soliton pulse along the propagation distance, such that: a) the calculation efficiency can be enhanced, and b) the number of sampling points required can be greatly reduced. This technique requires for the soliton equation to be solved by first splitting it into two simpler parts which can be calculated easily, either analytically or numerically, in the time domain. In addition to that, the mesh size is controlled by the shape of the soliton pulse such that the number of sampling points used is in every propagation step minimised.



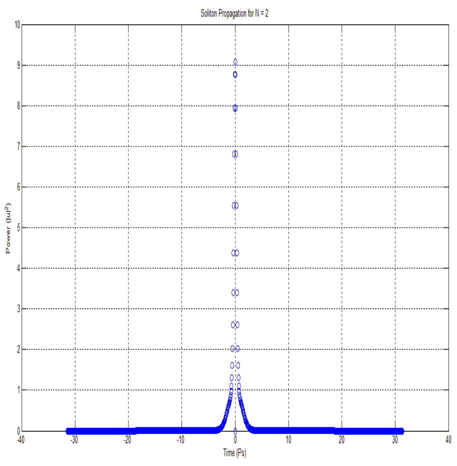
(a) Second–order soliton intial envelope.

(b) Second–order soliton envelope after propagating a quarter period down the length of fibre.

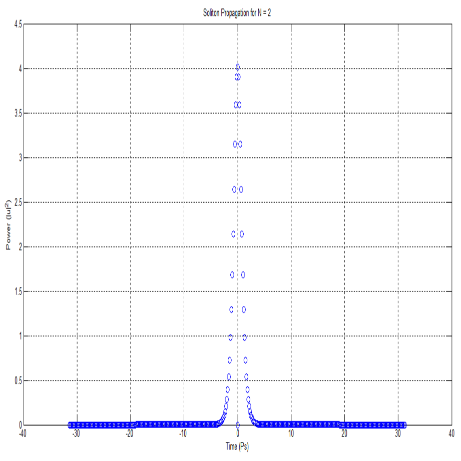
Fig. 10 The two initial stages of the soliton envelope as simulated using the FMAT method.

In order to implement the Fuzzy Mesh Analysis Technique, equation (1) is split into two parts, the Linear and Non–linear parts. It is obvious that any error arising as a result of the splitting operator is proportional to the choice of $\Delta(x)$, the propagation step. The Non–linear part can be solved analytically in the time domain, and the Linear part can be calculated by means of the Finite–Element Analysis Technique (Shum & Yu, 1998). Summarising, the technique in separate computational steps can be arranged as follows:

1. The initial input pulse shape is set for computation.
2. The Non–linear part is solved analytically in the time domain for a propagation distance of $\frac{\Delta(x)}{2}$.
3. $u(x+\frac{\Delta(x)}{2},T)$ as obtained from step (2) is used as the initial condition for solving the Linear part for another propagation distance of $\frac{\Delta(x)}{2}$.
4. Mesh control is adopted each time after the calculation of the Linear part such that the sampling profile can be optimised for the next calculation.
5. $u(x+\Delta(x),T)$ as obtained from step (3) is used as the initial condition for solving the Non–linear part of another propagation distance provided that the propagation distance is not reached.
6. Steps (2) to (5) are repeated until the required propagation distance is reached.



(a) Second–order soliton envelope after propagating half a period down the length of fibre (FMAT Method).



(b) Second–order soliton envelope after propagating for a full period down the length of fibre (FMAT Method).

Fig. 11 The two final stages of the soliton envelope as simulated using the FMAT method.

The computational procedure of the algorithm for the mesh control, in individual steps is as follows:

1. Using cubic shape functions, the temporal soliton pulse shape at a particular propagation distance is deduced as a function of T.

2. Based on the calculated cubic shape functions, both the temporal pulse shape and slope of $u(x,T)$ are obtained. Hence, the turning points ($\frac{\partial u(x,T)}{\partial T} = 0$) can be located by comparing the variation of the slope of $u(x,T)$.
3. The distribution of sampling points is defined within a sampling window along the axis. The left and right boundaries of this sampling window are defined as the magnitude of $u(x,T)$ just below 10^{-3}. Based on the location of turning points, new values of mesh sizes as well as a_i are assigned.
4. The procedures of assignment of mesh sizes can be described as follows: a) The number of turning points N_{tp} is counted within the sampling window, b) The sampling window is subdivided into $N_{tp} + 1$ regions with the turning points as the boundaries of each subdivided region, c) The sampling points within each subdivided region are equally spaced, but the number of sampling points N_j can be different between subdivided regions, d) If u_{pj} and u_{tj} are the boundaries of $u(x,T)$ within a subdivided region, N_j can be defined as $N_j = R_j(u_{pj}u_{tj})$, where R_j is a tuning factor. R_j is adjusted such that the summation of sampling points within each subdivided region is equal to N_{tp}, and e) If the total number of sampling points assigned to each subdivided region is more (or less) than the available sampling points of the sampling window, the above procedures are repeated for a different R_j until the optimal mesh size is achieved.
5. Hence, new values of $u(x,T)$ can be calculated using the optimised node distribution a_i.

Using this mesh control, more information of the soliton pulse can be obtained but without increasing the total number of sampling points (Fig. 10 & Fig. 11).

4 The Half–Adder Processor Scheme

The half–adder processor scheme, first introduced in (Bakaoukas & Edwards, 2009b), forms the essential central building block on which the overall FFT soliton computational scheme is wrapped around. The system reads the collision envelopes at distance and time specified points and uses this information to generate solitons with an appropriate phase value to represent the output of each "gate". The phase values of two of the output solitons determine the "sum" and "carry" outputs at the end of the computation process whilst all other solitons are superfluous to this calculation. By definition the half–adder (the sum implementation) is given by:

$$\overline{\overline{(X \cdot \bar{Y})} \cdot \overline{(\bar{X} \cdot Y)}} \tag{9}$$

In Fig. 12 the equivalent soliton scheme, originally presented in (Bakaoukas & Edwards, 2009b), is reproduced here for convenience. The points highlighted in this schematic representation by means of a bold circle indicate functional points at which a soliton collision, part of a gate, takes place; while, X and Y denote the initial input data. Full "gate" arrangements have been named and numbered (e.g. NAND (*), indicates the first NAND in the computational arrangement, NAND (**) the second, etc.).

In Figs 13 through to 16, the schematic representation of Fig. 12 is reflected on actual soliton collision simulations. Each individual gate–soliton collision is presented in a separate figure for clarity and comparison purposes. The simulation figures are to be followed in a top–to–bottom approach in the schematic representation of Fig. 12.

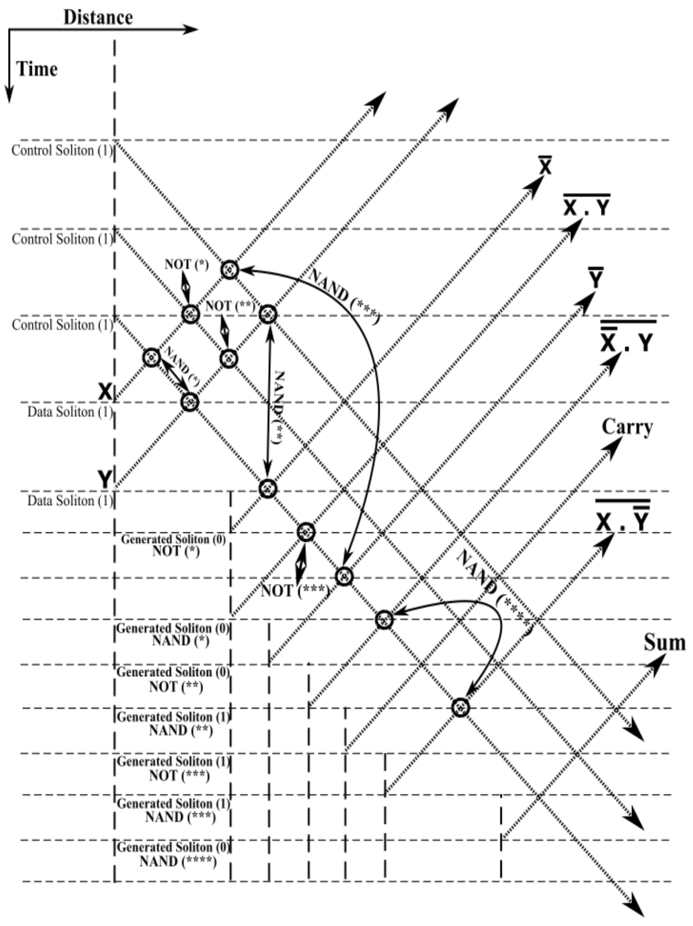
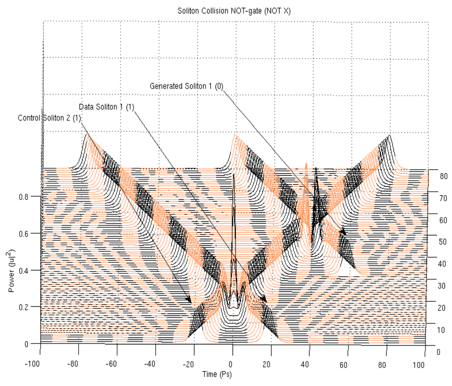

Distance
Time
Control Soliton (1)
Control Soliton (1)
Control Soliton (1)
X
Data Soliton (1)
Y
Data Soliton (1)
NOT (*)
NOT (**)
NAND (*)
NAND (**)
NAND (***)
NOT (***)
NAND (****)
$\overline{X}$
$\overline{X . Y}$
$\overline{Y}$
$\overline{\overline{X} . Y}$
Carry
$\overline{X . \overline{Y}}$
Sum
Generated Soliton (0) NOT (*)
Generated Soliton (0) NAND (*)
Generated Soliton (0) NOT (**)
Generated Soliton (1) NAND (**)
Generated Soliton (1) NOT (***)
Generated Soliton (1) NAND (***)
Generated Soliton (0) NAND (****)

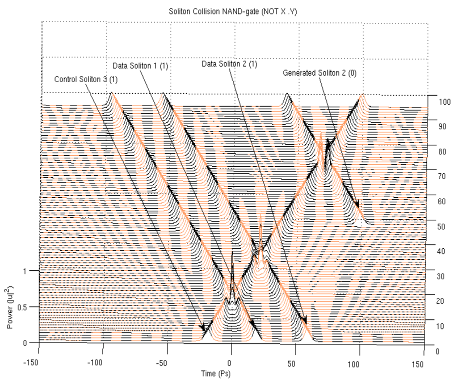
Fig. 12 The half–adder processor.

In all the figures the input–output "gate" sequence follows the soliton propagation direction. The point at which the soliton propagation begins (point 0 in the propagation scale across the depth of the figure) also reflects the input side of the "gate" and respectively, the point at which the soliton propagation ends (point 100 in the propagation scale across the depth of the figure) reflects the output side of the "gate".

At this point and for the approach used for the presentation of the material to follow in this paper to become clear, we need to stretch–out the fact that the computational complexities involved are extensively simplified if can become apparent that the scheme is flexible enough to be gradually get "packed" in fixed–purpose calculation lengths. This approach doesn't suppress the system from its generalisation properties, as the fixed reading points (as these have been identified and introduced in (Bakaoukas & Edwards, 2009b) and (Bakaoukas & Edwards, 2009a)) still hold their properties and continue to provide the system with all the capabilities initially identified as inherently characteristic of the computational system at hand.

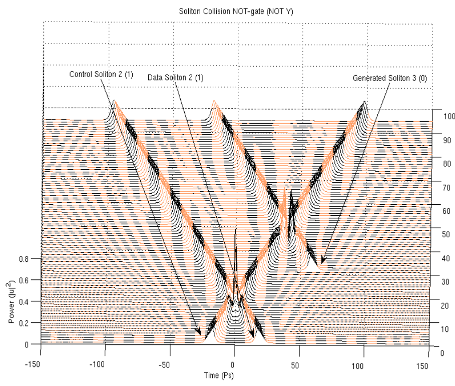


(a)

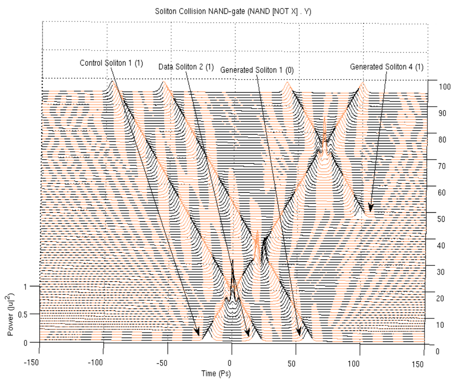


(b)

Fig. 13 a) The soliton "gate" NOT(*). The number in the brackets next to each soliton description is the bit value carried by the soliton and b) The soliton "gate" NAND(*). The number in the brackets next to each soliton description is the bit value carried by the soliton.

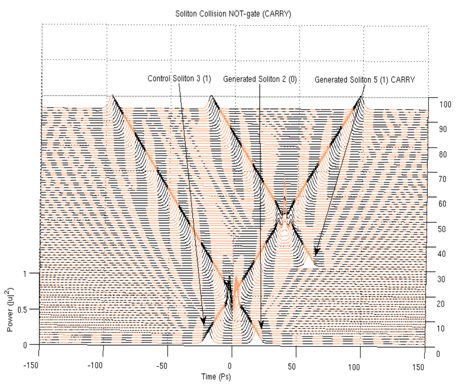


(a)

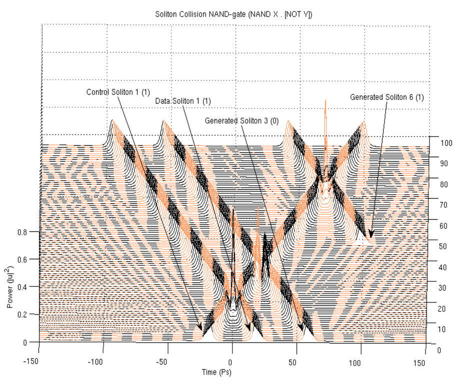


(b)

Fig. 14 a) The soliton "gate" NOT(**). The number in the brackets next to each soliton description is the bit value carried by the soliton and b) The soliton "gate" NAND(**). The number in the brackets next to each soliton description is the bit value carried by the soliton.



(a)



(b)

Fig. 15 a) The soliton "gate" NOT(***). The number in the brackets next to each soliton description is the bit value carried by the soliton and b) The soliton "gate" NAND(***). The number in the brackets next to each soliton description is the bit value carried by the soliton.

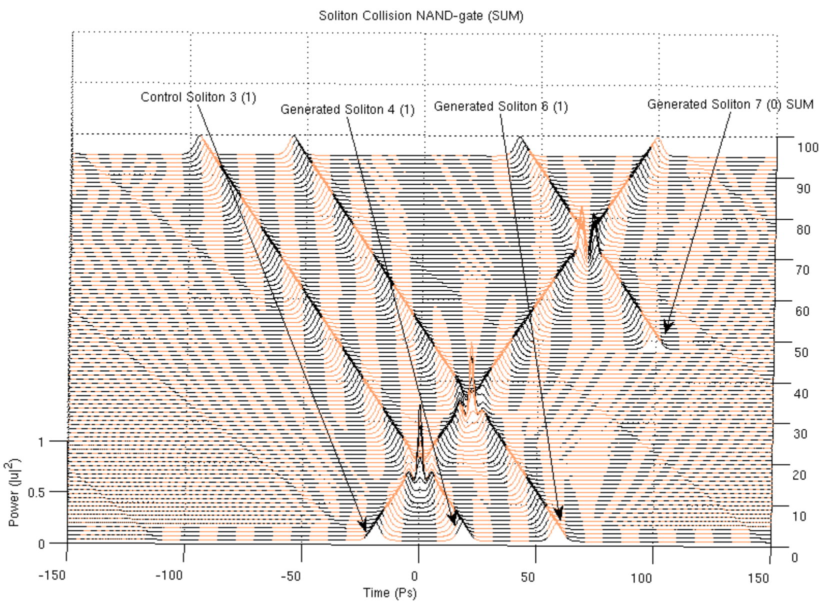


Fig. 16 The soliton "gate" NAND(****). The number in the brackets next to each soliton description is the bit value carried by the soliton.

5 The Two 2-bit Numbers Multiplier

In this section we present the "Two 2–bit Numbers Multiplier", which involves a half–adder as its lying–in–its–heart functional unit ("Three–bit Adder Arrangement"). The particular arrangement forms the compact small-scale equivalent of the "Two maximum–number–of–bits Numbers Multiplier", which for general purpose calculations must involve full–adders as well as half–adders in its arrangement. The reason behind choosing the Two 2–bit Numbers Multiplier is only the fact that the particular arrangement possesses all the functionalities and properties need to be demonstrated, while at the same time gives us the ability to keep the material presented at a minimum of extension and complexity in this paper.

Starting from the half–adder arrangement, if we now take a closer look in Fig. 12 we will notice that all the output solitons need to be ignored after reading and only the output soliton representing the "carry" value is to be allowed to propagate further on and enter the cascading second half–adder arrangement. Is exactly this soliton–bit that is required for the arrangement to complete the Three–bit Adder Arrangement output calculation as presented in a conventional block diagram in Fig. 17.

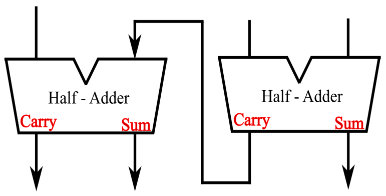


Fig. 17 The three-bit adder.

This "Soliton Suppression" requirement at the very end of a computational arrangement is not characteristic only of the computational scheme here presented but rather a common characteristic requirement in soliton computational arrangements as, for example, of the one introduced in (Rand & Steiglitz, 2007), where the additional property of not intersecting (solitons crossing paths but not colliding) is also a vital system characteristic requirement. The usual formal term coined for such kind of solitons is "Garbage Solitons" and is chosen to emphasise the fact that these solitons are to play no active role in the cascading calculations following the output of an arrangement. The way this "Soliton Suppression" can be physically achieved is, in general terms, a technicality, requiring some hands–on experimental work, in order for different methods and their corresponding effects on the overall computational arrangement to be properly studied. For these reasons we postpone, at this point, the explanation of how this "Soliton Suppression" can be accomplished.

In order to present a complete picture of the soliton arrangements as well as the almost unlimited flexibility possessed by the computational system (another reason is that in the view of the author the concept of "Garbage Solitons" is neither entirely satisfactory nor properly defined in its physical terms), in Fig. 18 an alternative soliton arrangement is presented which doesn't need "Soliton Suppression" any more in order for the cascading half–adder arrangement to commence calculation. In this new arrangement the general soliton pattern remains the same as in the original version, with the only difference that now the third control soliton is starting propagation at a time position shifted to the left (top) by four time slots (in Fig. 18 the original third control soliton propagation route has been maintained as well for comparison purposes). The order in which the individual gates are presenting their results is slightly changed as well. Shifting the third control soliton by four time slots to the left (top) of the arrangement has as a

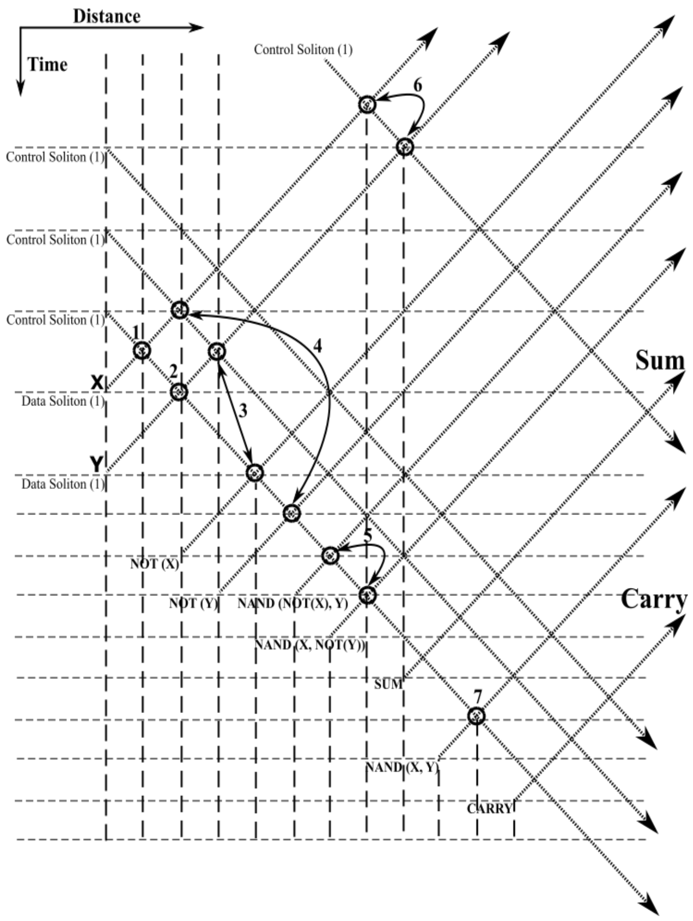


Fig. 18 The alternative half-adder arrangement. [Logic gates: (1) NOT, (2) NOT, (3) NAND, (4) NAND, (5) NAND, (6) NAND, (7) NOT].

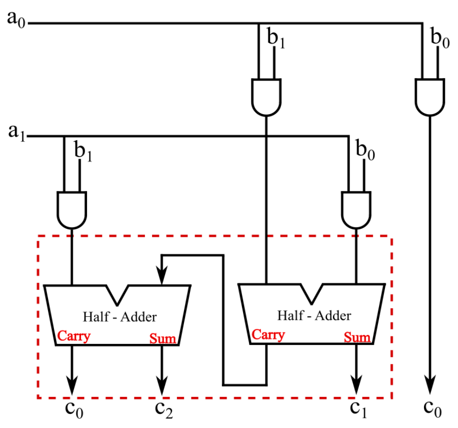


Fig. 19 The "Two 2-bit Numbers Multiplier".

result for the soliton carrying the “carry” value to appear at the end (bottom) of the output soliton order. So, this soliton can now be taken as the first input soliton of the new half–adder arrangement (literally, as it possesses the same propagation angle as the original input solitons to the half–adder arrangement) which, by use of a second appropriate input soliton and three control solitons, as required by the scheme, can provide us with the final computational result, without the need to include any kind of “Soliton Suppression” procedure.

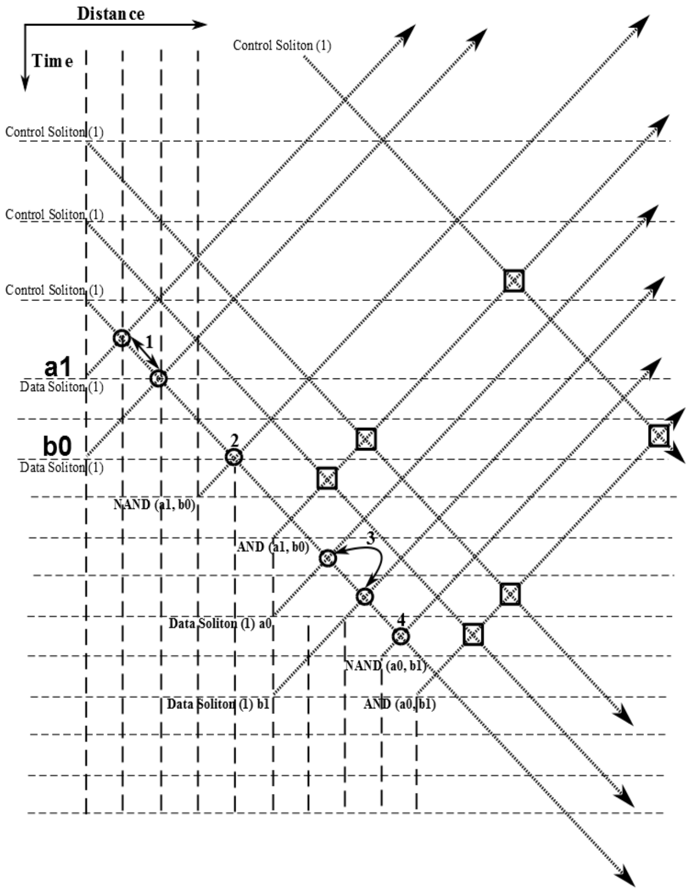


Fig. 20 Part of the “Two 2-bit Numbers Multiplier” (including two of the initial AND gates and the half-adder arrangement without the corresponding generated solitons). [Logic gates: (1) NAND, (2) NOT, (3) NAND, (4) NOT].

Having established and demonstrated the Three–bit Adder Arrangement, we can now build around it the full Two 2–bit Numbers Multiplier. The overall arrangement requires the addition of another four AND gates, to accommodate initial bit multiplications. The conventional diagram arrangement for the multiplier is as presented in Fig. 19.

In Fig. 20 part of the Two 2–bit Numbers Multiplier is presented. For illustration purposes Generated Solitons in Fig. 20 are shown to be closer together than they should be in an actual computational arrangement without loosing in computational properties or upsetting the result. Circular soliton collision points indicate collisions taking place during the initial AND gates calculations, while square soliton collision points indicate collisions taking place as part of the half–adder calculation process. The arrangement in Fig. 20 illustrates a certain degree of parallelism in the calculation process, which contributes significantly in increasing the overall computational speed of the arrangement. It comes without saying that the Two 2–bit Numbers Multiplier arrangement illustrated can be extended to cover any bit length required for the multiplication between two individual numbers. Again, the purpose here was to keep the length of the illustration to a minimum.

6 The "Butterfly" Soliton Arrangement

For the remaining part of the "Butterfly" calculation process, we need a soliton arrangement to convert a positive bit–number to a negative one. In order to achieve this we adopt the method of complementing each digit in a bit–number in turn (change 1 for 0 and 0 for 1) and then add 1 to the result. That way, the bit–number taken out of the procedure corresponds to a bit–number representing the negative equivalent of the initial bit–number.

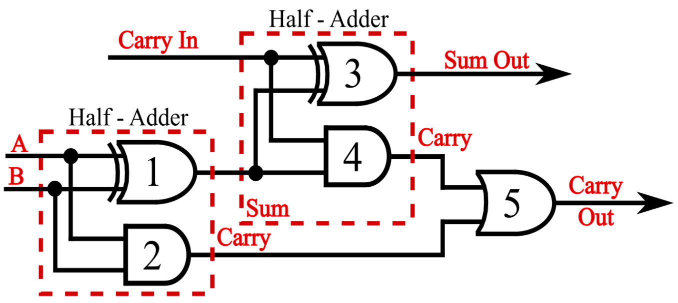


Fig. 21 The full-adder (conventional logic arrangement).

A series of collisions between the solitons carrying the bit–number values and a single control soliton with a phase value opposite to the one possessed by the control soliton that generated the initial bit–number, is enough to produce the bit–number complement. Since all the control solitons used so far in the computational arrangements presented had a phase value of π, corresponding to a bit value of 1, the appropriate control soliton to achieve the complement calculation must possess a phase value of 0, in turn corresponding to a bit value of 0. The addition of 1 to the complement can be easily achieved by means of full–adder arrangements internally consisting of two interconnecting half–adder arrangements and an OR gate, according to the conventional logic scheme presented in Fig. 21.

After the complement of a bit–number has been calculated, subtracting it from another bit–number requires the addition between the complement calculated and

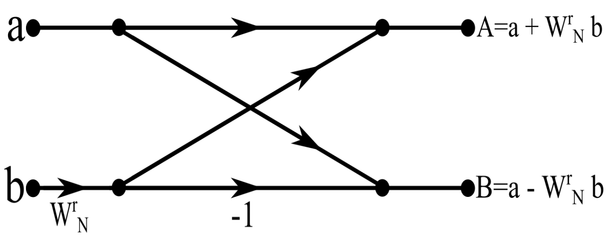


Fig. 22 Basic "Butterfly" computation in the decimation–in–time FFT algorithm.

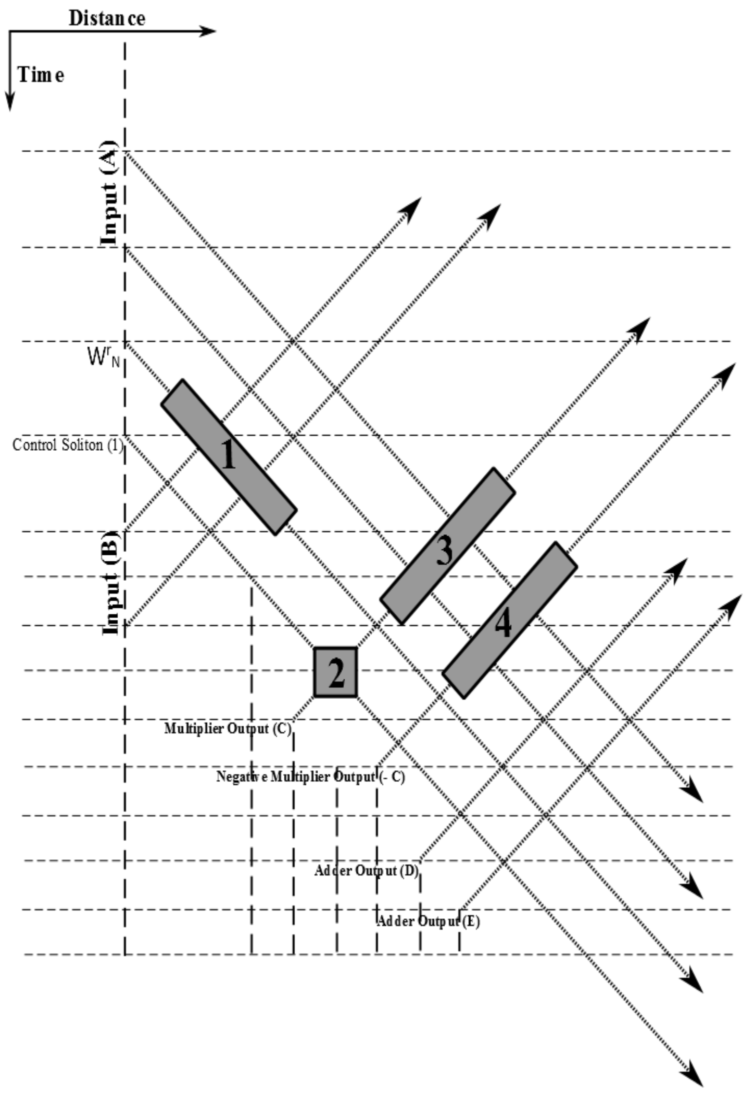


Fig. 23 The "Butterfly" soliton arrangement. [(1) Multiplier arrangement, (2) Negation arrangement, (3) Addition arrangement, (4) Addition arrangement].

the second bit–number. That way only half–adder and full–adder arrangements are required for the realisation of all the calculations involved in the "Butterfly" arrangement. Addition and subtraction calculations appear at the final stages of the "Butterfly" (Fig. 22), those that actually are giving the result and passing the values calculated to the next processing stage of the overall FFT calculation arrangement.

Having completed the presentation of the individual parts out of which the soliton "Butterfly" arrangement consists of, we can now present the schematic of the overall arrangement required. Fig. 23 presents the soliton "Butterfly" arrangement to full extend omitting, by means of a "black box" representation, those parts of the arrangement which have been previously analysed and illustrated. "Adder Output" (D) and "Adder Output" (E) appear at the end of the arrangement as required for the cascading "Butterfly" arrangements to continue further processing the data. All the output soliton propagation routes shown are indicative, since in an actual calculation of bit–numbers more than one solitons will represent the output bit–number of each block of calculation. As it is the case with the conventional Radix–2 FFT algorithm the first and the second decimation process results in a "shuffling" of the input data sequence, which has a well–defined order.

7 Conclusions and Further Research Directions

In this paper we surveyed the possibilities of an all–optical soliton FFT calculation and shown how this can become possible within the boundaries of the optical soliton 3NLSE domain. The outcome of this investigation is leading the way towards a fast all–optical soliton FFT calculation with the FFT phasors (roots of unity) to be represented directly by solitons of corresponding phase values, currently under extensive research by the author. In such a scheme the 8–point FFT phasors, for example, can be directly represented as:

$$\begin{array}{c} W_8^0 \rightarrow \textit{Soliton phase value} = 2\pi \\ W_8^1 \rightarrow \textit{Soliton phase value} = \frac{\pi}{4} \\ W_8^2 \rightarrow \textit{Soliton phase value} = \frac{\pi}{2} \\ W_8^3 \rightarrow \textit{Soliton phase value} = \frac{3\pi}{4} \end{array}$$

$$W_8^4 \rightarrow \textit{Soliton phase value} = -2\pi$$

$$\begin{array}{c} W_8^5 \rightarrow \textit{Soliton phase value} = \frac{5\pi}{4} \\ W_8^6 \rightarrow \textit{Soliton phase value} = \frac{6\pi}{4} \\ W_8^7 \rightarrow \textit{Soliton phase value} = \frac{7\pi}{4} \\ W_8^8 \rightarrow \textit{Soliton phase value} = 2\pi \end{array}$$

while the soliton phase values of π and 0 remain reserved to represent digit 1 and digit 0 respectively for the control and data solitons involved. This additional ability, when properly specified, will provide the overall computational scheme with a separate, well defined, and of a smaller fixed length FFT calculation arrangement without the need for it to consist of individual calculation arrangements based on the scheme's "gates".

References

Ablowitz, M. J., & Segur, H. (Philadephia, 1981). Solitons and the inverse scattering transform. *SIAM*.

Bakaoukas, A. G. (2013). Towards an all–optical soliton fft in the 3nls–domain. *Unconventional Computation and Natural Computation Conference*, *5*(6), 250–251. (UCNC 2013, LNCS 7956, Milano Italy, Springer–Verlag Berlin Heidelberg)

Bakaoukas, A. G. (2014). Computational capabilities of interactions between optical solitons. *International Journal of Information Science and Intelligent System*, *3*(2), 13–32. (Martin Science Publishing)

Bakaoukas, A. G. (2015). Computational capabilities of collisions between optical solitons. *International Journal of Information Science and Intelligent System*, *4*(1). (Martin Science Publishing)

Bakaoukas, A. G. (2016). An all–optical soliton fft computational arrangement in the 3nlse–domain. *Unconventional Computation and Natural Computation*. (15th International Conference, UCNC 2016, Manchester, UK, Proceedings, DOI: 10.1007/978–3–319–41312–9, Print ISBN: 978–3–319–41311–2, Springer International Publishing)

Bakaoukas, A. G., & Edwards, J. (2009a). Computation in the 3nls domain using first and second order solitons. *International Journal of Unconventional Computing (IJUC)*, *5*(6). (ISSN: 1548–7199)

Bakaoukas, A. G., & Edwards, J. (2009b). Computing in the 3nls domain using first order solitons. *International Journal of Unconventional Computing (IJUC)*, *5*(6). (ISSN: 1548–7199)

Bakaoukas, A. G., & Edwards, J. (2013). Quantum notation in the analysis of soliton gates interaction (part i)–the cn/ccn gates paradigm. *International Journal of Information Science and Intelligent System*, *2*(2), 1–50. (Martin Science Publishing)

Blair, S. (1998). Optical soliton–based logic gates. *PhD Thesis*. (University of Colorado at Boulder)

Fredkin, E., & Toffoli, T. (1981). Conservative logic. *International Journal of Theoretical Physics*, *21*, 219–253.

Jakubowski, M. H., Steiglitz, K., & Squier, R. K. (1996). When can solitons compute? *Complex Systems*, *10*(1).

Jakubowski, M. H., Steiglitz, K., & Squier, R. K. (1997). Information transfer between solitary waves in the saturable schrödinger equation. *Physical Review E*, *56*(6).

Jakubowski, M. H., Steiglitz, K., & Squier, R. K. (2001). Computing with solitons – a review. *Multi–valued Logic (Special Issue on Collision Based Computing)*.

Rand, D., & Steiglitz, K. (2007). Computing with solitons.
(July 1)

Shum, P., & Yu, S. F. (1998). Numerical analysis of non–linear soliton propagation phenomena using the fuzzy mesh analysis technique. *IEEE Journal of Quantum Electrinics*, *34*(10).

Steiglitz, K. (2000). Time–gated manakov spatial solitons are computationally universal. *Physical Review E*, *63*.

Toffoli, T. (1980). Reversible computing. automata, languages, and programming. *J. de Bakker, Ed. Springer–Verlag, New York*.