Soliton computing in the Toda Lattice

Controllable delay and logic gates

Mischa Cusveller

Master of Philosophy

University of York

Computer Science

August 2020

Abstract

Most people take the technological revolution from the past two centuries for granted and expect that this revolution will not slow down. In the recent century, a major presence in most people's lives has become electronic computers in one form or another. A new path for technology innovations needs to be set out if the revolution is to continue at the current pacing. One such promising path are optical computers using solitons as information carriers. Solitons have favourable properties and one under-explored soliton system for its computation capabilities is the Toda lattice, which has been used to model DNA and can be transformed into optical fibre models. By expanding the possible logic gate designs in this lattice, steps are made to bring us closer to realize a fully functional optical computer.

In the one-dimensional Toda lattice, it is possible to create a delay in the solitons' travels that can be controlled. The lattice has been used to create logic gates for computation, however, the delay mechanism has not been incorporated in those designs so far. With this controllable delay, an OR and XOR gate can be designed. The delay for a travelling soliton is created by incorporating a lattice made of harmonic oscillators between two Toda lattices. The duration of the delay can be controlled by changing the time difference of two solitons scattering against the harmonic oscillators. If the duration is too short, there are only reflections, however, when the duration between the two soltions' scatterings is long enough, transmission is possible. Both presented logic gates apply the controllable delay mechanism.

This thesis contains the following contributions, the first investigation of interaction of solitons in impurity, a new XOR and OR gate design, and code for simulating the Toda lattice and the mentioned contributions.

Table of Contents

List of Figures

Acknowledgement

My first expression of gratitude goes to the person I love the most in life, my dear wife Martina (Tina) Paulina Maria Iping-Cusveller. If it was not for you, I would not have been able to complete this work and not been able to make the many sacrifices that were needed. You have always supported me in any way you could, I hope that acknowledging you here for your many support given, will bring a smile on your lovely face. Next, I would like to thank my supervisor Dr Simon O'Keefe, who made it possible to undergo this process and end up with the thesis in front of you, I have improved much in the way of communicating scientific concepts and ideas. Your countless feedback on this thesis has improved the quality enormous and I am more than grateful for your efforts. I would also like to thank the Computer Science department and again my supervisor for helping to fund my time at this university. Through the teaching scholarship, I was able to learn much about teaching, finish the York Learning and Teaching Award and becoming an Associate Fellow of the Higher Education Academy. My thanks go out to the Computer Science and the York Cross-disciplinary Centre for Systems Analysis department for their support and guidance, they helped me grow as an academic. As last but certainly not least in my heart, are my family members and friends on the other side of the English Channel, who kept contact and were able to visit us in this country. I have wished many times that the distance between us were smaller so that we could visit each other much more frequently.

Declaration

I declare that this thesis is a presentation of original work and I am the sole author. This work has not previously been presented for an award at this, or any other, University. All sources are acknowledged as References.

Chapter 1

Introduction

Most people presume that technology progresses linearly however, a read through history will show a nonlinear trend of progression, as different options were explored and considered first before a new paradigm of technology emerges. We live in a period where the general public expects quick technological advancement in the area of computing and to deliver on this demand will require a new technological paradigm. One such promising candidate is optical computing done with the information carriers called solitons. Because of the attractive physical properties solitons posses, such as their robustness for carrying information for long-distance communication, much research has been done in investigating their computation possibilities. One model which allows solitons to exist is the Toda lattice and in this model, it has been shown that a few logical operations can be performed on travelling solitons [91]. In this thesis, we will demonstrate how an additional mechanism, such as the presented mechanism for controlling the delay of a soliton travel, can be used for performing computation inside the lattice. Further, the thesis outlines how this research can contribute to the realization of a functional physical computer that runs on optical solitons.

1.1 Approaching the limits of standard computing

When integrated circuits were introduced, Moore in 1965 [102] discovered a trend that is now famously known as Moore's law. In his original work, it stated that *on a chip the number of components would double every year*. It is this phenomenological relation that has been used as a measuring tool to gauge the success of standard computing, also called conventional computing. Ten years later he altered his prediction to *the number of components doubles every two years* [103]. He did this because from around 1980 onwards, packing more components in chips was becoming more difficult, which is one of the parameters in his law. In 1974 Dennard and his team [47] observed that the power density stayed constant while the number of transistors located in a given area grew. This is because the power that a transistor requires is proportional to the size of the transistor. Like Moore's law, this was another gauging tool used for measuring technology success. This observation is called Dennard's scaling rule and it ran alongside Moore's law until ten years ago, when it became too difficult to maintain this rule for companies due to the increased electric current leakages [24, 111]. The leakages of today's transistors are

caused by the imperfect insulation layer separating the gate from the other electrode of the transistor. At 7nm scale decreasing the size of the transistor will increases the flow of unwanted electrons even further, at this scale the insulation layer in the transistor becomes insufficient due to quantum tunnelling and other quantum effects [118]. However, Moore's law has not ended as of writing this thesis. Companies are still trying to continue the trend [94], but the trend however has slowed down [98]. There will probably be a point where Moore's law will break down as it is not a law of nature, but a phenomenological relation, which depends on the ingenuity of technology development. It has been proclaimed that when Moore's law breaks down [111], a new course in the development of computing hardware needs to be made, thus this law has become an indicator of the urgency for this new course. A reason for setting this new course might be found in the desire to extend the rapidness of technological progress we have experienced in the last century.

Let us take a step back in history. The idea of machines performing computations rather than humans is not a recent interest. For example the Stepped reckoner, also called the Leibniz Machine, in 1673 [27] invented by G.W. Leibniz, was a mechanical machine that could multiply numbers by repeating the addition operation [126]. It will not be until the twentieth century before a functioning electronic computing machine was developed. In 1946 [117] a group of computer designers came together in the Moore summer school lecturers to collaborate on the development of an electronic general purpose computer. Before the invention of the general purpose electrical computer, computers were general purpose but mechanical, or electrical and not general purpose. Doubt has been risen by Conrad and Zauner [41] if the general purpose electrical computer (the digital standard computer) is truly general purpose, as they argue that a realizable general purpose machines can only implement a portion of its input-output capabilities. For some instances, standard digital computers may not be the best solution, because it is slower, bigger, uses a lot more energy, becomes too warm or perhaps it cost more money than a computer made for the specific task would be. In the field of unconventional computing, some researchers are developing computers specialised for a specific task, which should perform that specific task better than the standard digital computer. Others are searching and developing what could become the next generation general purpose computer. This field is called unconventional computing mainly because the systems performing the computing have not been brought to the market (yet). Most of those researched systems only recently have been used for their computing capabilities. The unconventional computers now being developed could be decades away before they overtake the majority of tasks being performed by today's standard computers. However studying them now not only allows us to ask questions about the nature of computing, but it also allows us to study never seen before aspects of nature itself. There are questions which we can only hope to answer on an unconventional computer, one example is simulating quantum mechanics on a computing system. Feynman [59] argues that to simulate a large quantum system a quantum computer is needed, as nature is quantum mechanical and not classical, such as classical physics bounded standard computer. However, a sufficient simulation of some aspect of quantum mechanics can still be done on the standard computer, as a search on quantum mechanics simulation literature will show. An example of this is the paper *Sufficient conditions for efficient classical simulation of quantum optics* [114].

Moore's law is used as an indication of the necessity of new computing developments and some tasks require unconventional computing methods, but ultimately every computing system is faced by the limits of the physical world. One such limitation is argued by Landauer, he states that the minimum amount of energy needed for a switch in an ideal world is in the order of k_BT [93] where k_B is the Boltzmann constant and T the temperature. This amount of energy is the difference between the entropy of the binary states. A physical switch will always cost more energy than the amount to only change the state of an ideal two-level system. He also argues that a useful logical machine needs to operate on irreversible logic operators (example of an irreversible operation is an OR gate, which is irreversible if the only information you have is its output). The physical maximum amount of possible operations executed during a second, as argued by Lloyd [96], is given by $2\Delta E/\pi\hbar$, were ΔE is the available energy in the system and $\hbar$ the Planck's constant divided by 2π. This result is based on the time required to distinguish between the different quantum states. There are more limits set by physics for any real world system performing computations, but the examples from Landauer and Lloyd show what kind of limitations we have to expect when developing computing with unconventional physical systems. There are advantages when we couple one system to another system to form a hybrid system, combining the strengths of both systems, which may be the direction we need to go in the future. However, an example of the disadvantages of hybrid systems can be seen with standard computers connected by optical fibre networks, its an electronic system combined with an optical system, combining the strengths of silicon based computing to the fast communication properties of optical signals. But the connection between electronic and optics is causing what is called an *electronic bottleneck*, also called the *electro-optic bottleneck* [113]. The electronic components cannot access the data transport capabilities of the optical fibres, causing a bottleneck to the network. The electric signal travels only a fraction of the light speed, this also occurs when electronic computers communicate over the optical fibre network. One solution to this would be an all-optical computer, which is a computer that completely operates on light rather than electrons. Over the years different approaches for optical computing have been researched [9] and one of the promising options is computing done with solitons.

Important to this thesis is the concept of the solitons and most mediums that have soliton solutions behave classically. A soliton is a hump-shape wave localized in a certain region of space. The interest computing performed with solitons may have begun when they were able to carry information over a long distance in optical fibre without significant background interference [55]. To mention only a few examples, in 1995 Kawai, Iwatsuki and Nishi [81] published their work on the sending and receiving of soliton transmission over a distance of 30,000 km at a bit rate of 10Gb/s. In that same year Kubota and Nakazawa [88] published that they were able to send a soliton successfully over 2,000 km at 20 Gb/s and 2,500 at 10 Gb/s using an existing optical network in Tokyo. Amiri *et al.* [10] in 2017 was able to send and decode a bit pattern from 180 km away, using soliton for secure communication as an application of chaos theory. At this moment it is still crucial to restore the logic values of optical solitons during long-distance communication. One method is by using what is called adiabatic amplifiers [19, 63, 100]. The examples that use soliton for long-distance communication show that it is possible to receive and send signals from a great distance at a great speed. This is only possible because solitons

are robust against small perturbation of the medium, making them excellent carriers of information in optical fibres. As mentioned before, an all-optical computer would solve the electro-optic bottleneck problem and with the advantageous properties of soliton, a computer operates on soliton logic circuits, could pave the way for future technology.

Soliton does not only exits in optic media, but can also be found in shallow water and deep water, a tsunami is such an example, but also as cloud formations such as morning glory clouds and in many more mediums. J.S. Russell discovered this phenomenon in a canal in Scotland in 1834 and was able to reproduce those solitary waves in his lab [45]. A photo of a soliton on the aqueduct named after him, the Scott Russell Aqueduct can be found in [53, 107] The photo shows how the soliton he observed may have looked. However G. G. Stokes and G. B. Airy were not convinced about the existence of those waves and they derived wave formulas for nonlinear wave behaviour to show that soliton solutions were not possible. In the decades after the publication, it was shown that those derivations do not hold in general. In 1895 D. J. Korteweg and G. de Vries demonstrated mathematically that solitary waves for shallow water can exit. Interestingly they had rediscovered the equations that J. Boussinesq derived back in 1872 in France, however, Boussinesq was not aware of the discovery made by Russell. It was not until 1954 that the derivation of Korteweg and de Vries would be used again. In that year E. Fermi, J. Pasta and S. M. Ulam looked at a numerical model of a discrete nonlinear mass-spring system. They followed a suggestion of P. Debye for a nonlinear system and found an unexpected behaviour, which a decade later turned out to be solitons, by the works of N.J. Zabusky and M.D. Kruskal [87] and M. Toda [123]. Since then, research in solitons has been an ongoing process.

Figure 1.1: *A picture of the Scott Russell Aqueduct. This image is from [53] and can also be found in [107]. This image shows a soliton moving over a bridge named after the first person who studied the behaviour of solitons.*

1.2 Aim of this thesis

This thesis presents a structure for employing solitons in computing. We will do this in the Toda lattice, a medium that has soliton solutions and is made of sites that are connected by the nonlinear Toda potential. It has been shown that some varieties of the Toda lattice are equivalent to varieties of the nonlinear Schrödinger equations (NSE) [11, 83], an equation that describes soliton dynamic in optical fibres. The majority of literature on employing solitons for constructing logic gates, are focussed on optical systems and use primarily the NSE or one of its variations. Those research could help solve the electro-optic bottleneck problem by developing a fully optical computer. One such work that shows the equivalence between the NSE and the Toda lattice is from Arnold [11]. It was shown in his work that a complex variable extension of the Toda lattice is equivalent to a simplified version of the NSE. Another example is a set-up of the Toda lattice where the zeroth site is constrained in its movements, the soliton solution of this set-up is equivalent directly to the soliton solutions of the NSE [83]. The Toda lattice not only played an important role in the recent interest in solitons, as mentioned in the previous section, but this system is still being actively researched to this day [17, 108, 112]. The Toda lattice has been used to model solitons in DNA [105] and also applied to building logic gates [91]. It is the range of possible logic gates designs which this thesis will be expanding. This will be done by exploring a mechanism that allows control over the delay experienced by a soliton travelling inside the lattice and this option is subsequently applied in the design of a new set of binary logic gates in the lattice. This thesis will not be discussing the transformation required to go from the Toda lattice soliton solutions to those of the NSE. In theory, it is possible to transform the results described in this thesis to equivalent results in the NSE. Consequently this thesis indirectly provides an approach to designing logic gates within an optical fibre system, in the hope that it can be used in an all-optical computer. Recognize that in the past much research was required on the transistor before it became widely available, perhaps more research on soliton logic gates is required before they will see general applications in the area of computing.

1.2.1 Contribution to knowledge

The author claims that the following contributions to knowledge are made through this thesis:

- The first investigation of an interaction between a travelling soliton arriving at the impurity and a soliton already trapped inside the impurity. Through this investigation a method of controlling the period a soliton is trapped inside the impurity was discovered.
- A new design for a XOR and an OR logic gate in the Toda lattice is outlined. The gate designs uses the method of controlling the trapping period.
- A simulation coded in Mathematica has been created by the author from the ground up. This program simulates the Toda lattice, has the method for controllable trapping time of a travelling soliton incorporated and is able to simulate the XOR and OR gate designs.

1.3 Thesis outline

The remainder of the thesis is organized as follows:
Chapter 2: This chapter focusses mainly on the background information necessary to follow the main subject presented in this work. The first part of the chapter will focus on computing done by physical systems and the framework which makes discussion about this topic possible. This is followed by a general introduction into the physical phenomenon of solitons and we end this chapter by discussing how solitons have seen application in the area of computing.
Chapter 3: Introduces the theory behind the Toda lattice, a system that allows the existence of solitons. This chapter also treats the mechanic of delaying a soliton in its travel inside the lattice.
Chapter 4: In this chapter we are going over the measurements done in the Toda lattice simulations. This chapter contains the original contribution to knowledge made by the author. An approach is shown of controlling the duration of the delay for a soltion and how this new kind of delay mechanism is implemented in the creation of an OR and XOR gate in the lattice.
Chapter 5: The final chapter summarises the thesis and goes over some possibilities for future continuations of the presented work.
Appendix A: This appendix contains the computer code for simulating the Toda lattice and the code for generating the results from chapter 4. The first part provides a general outline of the programming code and how it works in principle. This is followed by the complete Wolfram Mathematica code itself.
Appendix B: In this appendix we introduce some alternatives to binary logic, such as fuzzy and multi-valued logic. This chapter is a complement to the suggestions brought up in chapter 5.

Chapter 2

Conventional and unconventional computing

In this chapter are introduced two central concepts of the thesis. The first concept is the notion of physical systems that are capable of performing computation. It is not only the human mind that is capable of doing computations, as mechanical and electronic machines have shown their capabilities in this day and age. But we also find in unexpected places physical systems capable of computation, such as the bulk of nuclear spins, where nuclear magnetic resonance is used to realize logic gates [18]. Second concept we are introducing here are solitons, we will go over the definition of solitons and the different types that exist. Both concepts are then combined into *soliton computing*, computing done with solitons. The approach of using soliton to carry information and logic gates that modify the variables of solitons will be discussed in this section, alongside a discussion of the current state of this field. This last section lays the ground for chapter 4, where the author's original contribution to science are discussed.

2.1 Computing on a physical system

2.1.1 Representing information

In this section, we go over what it means for a physical system to perform any kind of computation. In standard electronic computing, the logic values are represented by a voltage level on the wires. For most electronics, the value 1 is represented by a voltage level close to 5 V and the value 0 by a voltage close to 0 V. As an example, the SN74LS00 NAND gate from Texas Instruments [121] under recommended operating conditions will recognize a logic 1 if the voltage is $\geq$ 2 V and a 0 when the voltage is $\leq$ 0.8 V. The logic value that the chip outputs will be uncertain when the voltage value is between 0.8 and 2. The keyword in all of this is *representation*, the logical values are represented by a physical value, state or quantity in a real world system. The logical values are only abstract objects and when we want a physical system such as calculators to perform any computation, we need to have a physical value, state or quantity to represent this abstract object. Numbers are also abstract objects, having a certain amount of one type of object in one place represents that amount as an abstract number. When we perform calculations in our mind, we use the abstract numbers and perform abstract manipulations on those numbers to arrive at another abstract number. Even though for most of us we were

initially taught the other way around as children, to make calculations easier, we were taught that the abstract numbers represent that amount of a certain object. But this is perhaps more of a difference in how we view things, that goes back as far as the discussion between Plato and Aristotle over the nature of physical objects and the theory of ideal forms. Do we recognize physical objects for what they are due to our experience, or do we recognize them because we have some innate knowledge of their ideal forms? So too may the difference how we view things exist between a stereotypical mathematician that would argue that physical amount of the same object represents the abstract numbers and a stereotypical engineer that would view that abstract numbers represent objects in the physical world. Going back to the view that physical objects represent abstract numbers, in an abacus, the numbers are represented by beads on a rod, an abstract number is represented by the beads slid to a certain side of the rod. Another way to represent information is with a strip of compartments [60], such as ice trays, where the state of the compartment corresponds a binary value, the value 0 corresponds to a compartment being empty and 1 when it is filled, for example by a pebble in the compartment. We will later come back to this example and it might be used to perform calculations. In the Turing machine, information is represented by the symbols on the tape. From this point onwards in this thesis, the writing will be written from the point of view that abstract objects such as numbers are represented by a physical value, state or quantity in the physical world. So far the focus has been on numbers and how they are represented in the physical world, but numbers alone are not useful enough for what we want to achieve when it comes to computing. If someone gave you a number, for example three, then there is not enough information in that number until we know the context of the number for example, that a specific neighbour has three children. Therefore when we mention information, it is defined as a set of numbers where the context of those numbers are understood. There are a lot of different ways information can be represented, that is to say, encoded into a physical form, but it always needs to be in conjunction with the way how information is manipulated and decoded back to the user in some useful way.

2.1.2 Performing calculations

Due to the abstract nature of numbers, performing operations on those numbers will require abstraction. When we represent values as physical value, state or quantity, performing operations on those values will mean that there is a way to change those physical values or quantities in a predictable matter. How those values change predictably will determine which mathematical operations we can associate with the behaviour of the physical system. Going back to the standard electronic computing machine, when we are computing something, a series of mathematical operations are being performed such as comparing values, adding two values, subtracting, multiplying, etcetera. This approach can also be used in unconventional computing systems. In the earlier mentioned example of pebbles in a compartment, we can create an additional operation by placing two rows of plastic compartments underneath each other and some compartments containing pebbles. We can move the pebbles from the lower row into the row above, but when there already is a pebble in the above row, we remove that pebble and place one in the compartment left of the previously occupied compartment. See an example of this addition done in figure 2.1. While this operation can be done by us humans and moving pebbles to add

two numbers is certainly not a conventional way of doing additions, it is however also not automatic.

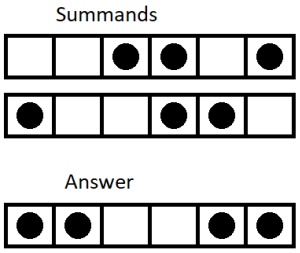


Figure 2.1: *The sum of two strips of compartments filled with pebbles. Below the two strips is a strip containing the sum of those strips. The addition is done in the same convention as is done when adding two binary numbers.*

It could very well be that earlier inventors of mechanical calculators such as Pascal, Leibniz, Monroe and Babbage [27] were motivated to having calculation performed automatically after the user has entered an input into the machine. The field of unconventional computing, among its other research topics, investigates different physical systems for their computing capabilities. An example of an unconventional physical system being used for computing can be found in the work of Assunção *et al.* [12]. They had a Y shaped lattice and the top part of the Y shape is used as the location for the inputs. The input signal for their AND, OR and XOR gates are harmonic waves and the presence of the wave represented 1 and its absence 0. Harmonic waves are additive in their amplitude, meaning that the amplitude when two waves are atop of each other is the sum of the amplitude of both waves. The value of the output was determined by the amplitude of the wave arriving at the bottom part of the Y shape, if the amplitude was below a certain threshold then that represented the output value 0 and if it was above or equal to the threshold, that represented 1.

A common approach of unconventional computing found in the literature is collision-based computing. As the name suggests, colliding objects are used to perform computation. An example of this can be found in Conway's Game of Life, which was used to prove that it is Turning complete among other things, this was done by colliding gliders [20]. One of the fundamental models of collision-based computing is the billiard ball model [130], named such because this model can be physically realized with billiard balls. The billiard ball model can not only be used to realized Boolean logic operations, but also realizing conservative logic operators [62]. It was Fredkin and Toffoli, the inventors of this computing model that introduced collision-based computing together with conservative logic. They described conservative logic as "a comprehensive model of computation which explicitly reflects a number of fundamental principles of physics, such as the reversibility of the dynamical laws and the conservation of certain additive quantities" [62]. This type of logic is still Boolean in nature because there are only two possible values, but the set of operators are a little bit different than those operations found in Boolean logic. Only Boolean valued operations that are reversible are considered in the set of conservative logic. The universal operator in conservative logic is the Fredkin gate and its function is defined in table 2.1.

u	x_1	x_2	y_1	y_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	1
0	1	1	1	1
1	0	0	0	0
1	0	1	0	1
1	1	0	1	0
1	1	1	1	1

Table 2.1: *The truth table of the conservative logic Fredkin gate. The control signal is u, the inputs are x_1 and x_2, the outputs are y_1 and y_2. The role of the control signal in this gate is to reverse the outputs of y_1 and y_2.*

Going back to the billiard ball model for a moment, the billiard balls in this model can undergo elastic collision that follows the same rules as the perfect gas molecules under classical kinetic theory. This is drawn in figure 2.2(b). The picture in 2.2(a) shows the two initial directions a ball is given and in (c) the balls are deflected by mirrors, just as perfect gas molecules are by the outer walls. Those mirrors allow the balls to be directed in another direction than that they initially were.

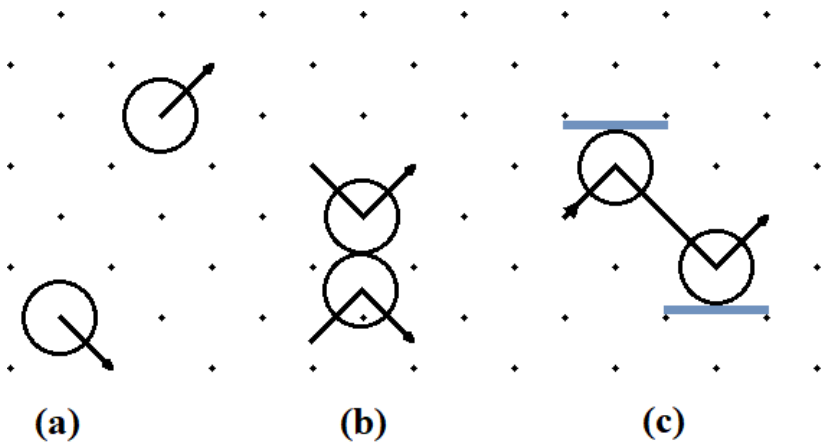


Figure 2.2: *Billiard ball model. (a) shows the two initial movements the ball are given, (b) two balls colliding and their momentum has been swapped, and (c) shows how a ball is deflected by a mirror. This image is inspired on a figure from Fredkin and Toffoli's work [62].*

A logic gate in conservative logic which can be constructed in the billiard ball model is the interaction gate, drawn in figure 2.3(a). The binary values 1 and 0 are represented by the presence and absence of a billiard ball respectively. The interaction gate has three different outcomes, pq, $p\overline{q}$ and $\overline{p}q$, the grey dashed lines shows the different paths the balls can take. Those paths are only taken when either input p or q is absent. Another logic gate that can be constructed in this type of logic is the switch gate, drawn in figure 2.3(b). The input p plays the role of the control signal, because the value of p determines the location were the input q arrives, thus this set-up acts as a switch for the signal q. The Fredkin gate can be build from a series of switching gates connected in a particular way, as shown in Fredkin and Toffoli's paper on conservative logic [62].

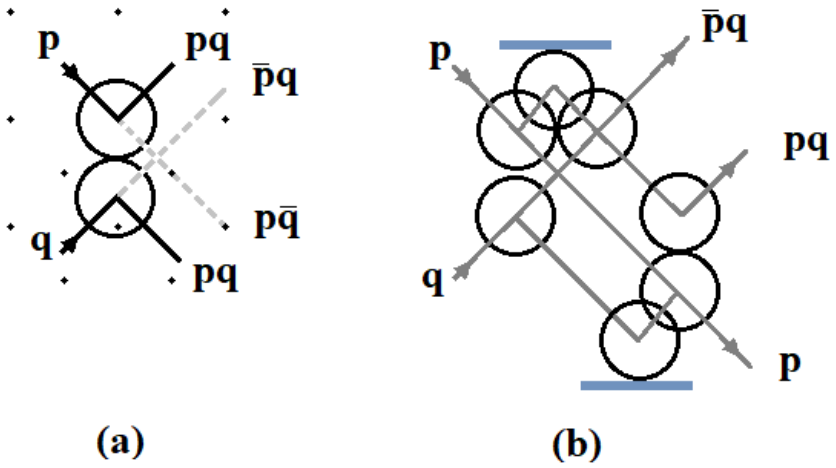


Figure 2.3: *Two gates in the billiard ball model. The inputs are p and q, the values* 1 *and* 0 *are represented by the presence and absence of a billiard ball respectively. The line above the letters p or q indicates the NOT p and NOT q, which is the absence of the input ball p or q. (a) shows the interaction gate and (b) shows the switch gate. This image is inspired on a figure from Fredkin and Toffoli's work [62].*

In this section, we have gone over different physical implementations of operators, from simple additions with pebbles and compartments to the billiard ball model. It is only when the physical values, states or quantities change predictably as argued by Horsman *et al.* [69], that we are allowed to speak of computation being performed by a physical system. We are going over this statement and the reasoning behind it.

2.1.3 Computing definition

Now that we are familiar with the notion of information being represented by a physical parameter, state or quantity, and both information and operations belonging in the realm of abstraction, we can now dive into what it really means for a physical system to perform computations. There are different definitions of computation [43], one such definition is given by B. MacLennan [99]: "Computation is a physical process the purpose of which is the abstract manipulation of abstract objects". This definition is in line with the approach of a physical system performing computing described so far in this chapter. A more precise definition is found in the framework developed by Horsman *et al.* [69], which sets out the criterion that needs to be met before we can categorize a physical system capable of performing any kind of computation. The paper gives a thorough treatment of the framework, but what follows in this section are pointers from the paper that are sufficient for the intention of this thesis. To get to the heart of this framework we need to take a step backwards and consider the relationship between physical experiments and models in the natural science fields. In those fields, it is through scientific models that we can make sense of experimental data. For example, Eratosthenes measured the length of a shadow in Alexandria and he was familiar with the distance between Syene and Alexandria. Just those numbers alone are not important on their own, but because he had a model of a spherical earth, he was able to calculate from his shadow length measurement the radius of the earth. However, the relation between physical experiments and models can also change our world view. Take for example the geocentric model, which places the

earth as the centre of our solar system, and the heliocentric model which places the sun as the solar system's centre. Both models were in the time of Galileo able to explain the known observations however, the heliocentric model was used for its ease when it came to performing calculations and the geocentric model for theological reasoning. It was the measurement of the parallax of nearby stars that could not be explained from the geocentric model, that broke the tie between both models. Since then the heliocentric model has become the dominant model for thinking about our planet's place in the solar system. This example illustrates that when we measure new observations, our models change and when our model changes, we sometimes change our world view with them. A model is as good as its predictive or explanatory powers are. Through a model are we able to describe our observations, a model is an abstract object which may be written in mathematical formulas or drawn pictorial. When creating or using a model, there has to be some connection or relation between the observations of a physical system and the abstract model that describes physical systems behaviour(s). Using the same notations as Horsman *et al.*, let us indicate this relation with the symbol $\mathcal{R}$, the physical system that observations are taken from with $\mathbf{p}$ and the abstract model of $\mathbf{p}$ with $m_{\mathbf{p}}$. The relation may be written as $\mathcal{R} : \mathbf{p} \to m_{\mathbf{p}}$, it is this relation that brings us from the physical system to the abstract world where we are describing the relevant qualities and parameters of the physical system that is being studied.

A model $m_{\mathbf{p}}$ is always theory dependent, for example, the motion of the planets can be described by either Newtonian mechanics or general relativity, but both theory make on some aspects different predictions. One example is the motion of Mercury observed from Earth, which can not be determined accurately by Newtonian mechanics, but can be better predicted by general relativity. While we might, therefore, want to enforce all calculations about gravitational effects to be made with general relativity, even if it is mathematically more difficult to do so. In practice for most calculations involving gravity on a planet's surface will Newtonian mechanics suffice. Another standard example that shows that the predictive power of a model needs to be viewed from the theory it is incorporating, is between classical mechanics and quantum mechanics. We know that at larger scales quantum effect disappears and at a very small scale classical predictions break down. When we look at the phenomena of light, in classical optics the theory describes light as waves, while the theory used in quantum computing often describes light as a particle. Now from quantum mechanics viewpoint, we know that every particle can be seen as a wave or particle, that both interpretations are equivalent. But the model predictions are depended on which of the two interpretations are used. Because models are theory depended, we indicate the relation $\mathcal{R}$ with $\mathcal{T}$ for the theory the relation and model is associated with, $\mathcal{R}_{\mathcal{T}} : \mathbf{p} \to m_{\mathbf{p}}$. The theory not only includes the experiment that is being tested, but also the theory that described the behaviour of the apparatus being used for measurements. At the moment we have a way of writing down how a static physical systems can be described by a static model. Of course, we want to have systems that evolve. Let us note the dynamics of the model by the relation $\mathcal{C}_{\mathcal{T}}$, that moves the model $m_{\mathbf{p}}$ over time into the new state of the model, written as $m'_{\mathbf{p}}$. Thus we have $\mathcal{C}_{\mathcal{T}} : m_{\mathbf{p}} \to m'_{\mathbf{p}}$. Our physical system will also evolve, let us use $\mathbf{H}(\mathbf{p})$ to describe the process that brings the physical $\mathbf{p}$ to the new state of the systems, to $\mathbf{p}'$. In physics, the dynamics of a physical system are often described by the Hamiltonian is often indicated

by the letter H. The evolution of the physical system can be written as $\mathbf{H}(\mathbf{p}) : \mathbf{p} \to \mathbf{p}'$.

Now that we have a relation that brings us from the physical system to the model, and we have a relation for the dynamics of both the model and physical system, we are now able to compare how faithful the evolved model is to the evolved physical system really is. We, therefore, want to bring $\mathbf{p} \to \mathbf{p}'$ and then use the relation $\mathcal{R}_{\mathcal{T}}$ to have a model describing this new state of the physical system, $m_{\mathbf{p}'}$. This can be written as $\mathcal{R}_{\mathcal{T}} : \Big[\mathbf{H}(\mathbf{p}) : \mathbf{p} \to \mathbf{p}'\Big] \to m_{\mathbf{p}'}$. When we evolved the model, we ended with the state of the model $m'_{\mathbf{p}}$ and now that we have $m_{\mathbf{p}'}$, we are going to compare if the outcome from both evolutions of the system is exactly same. If they are the same, we can say that $m'_{\mathbf{p}} = m_{\mathbf{p}'}$ and that it is a good model to represent the dynamics of this particular physical system. In practice it is not always possible to obtain this due to modelling or experimental limitations, we sometimes have to be satisfied with a model that is 'good enough', which is $|m_{\mathbf{p}} - m_{\mathbf{p}'}| < \epsilon$. The size of this error ϵ depends on the context of the usability of the model and the user their criteria. If $m_{\mathbf{p}} =_{\epsilon} m_{\mathbf{p}'}$, which means that the model agrees with one another within the accepted error ϵ, we can say that the theory has not been invalidated in this one specific experiment. This is summarized schematically in figure 2.4.

Perhaps rather than taking the route $\mathcal{R}_{\mathcal{T}} : \Big[\mathbf{H}(\mathbf{p}) : \mathbf{p} \to \mathbf{p}'\Big] \to m_{\mathbf{p}'}$, we would like to have an evolved model that can make new predictions or have explanatory power. Effectively such a path would be $\widetilde{\mathcal{R}}_{\mathcal{T}} : \Big[\mathcal{C}_{\mathcal{T}} : \Big[\mathcal{R}_{\mathcal{T}} : \mathbf{p} \to m_{\mathbf{p}}\Big] \to m'_{\mathbf{p}}\Big] \to \mathbf{p}'$. Where $\widetilde{\mathcal{R}}_{\mathcal{T}}$ brings us from the abstract domain to the physical domain, however, this is only possible when the theory has been robustly tested and we can extrapolate new predictions of the physical system dynamics from the model itself. To name only two famous cases of many where researchers were able to predict phenomenon from theory before any experimental confirmation. Paul Dirac created a theory that predicted the existence of positrons (anti-electrons) [56], even when his theories were tested and proven wrong, he kept believing in his theory. It later turned out that he was right and the existence of positrons was proven, as the other experiments that invalidated Dirac's theory were shown to not be correct and a new experiment was able to produce positrons. Another example is when Einstein predicted that certain stars would still be visible during a solar eclipse [72], even though those stars should be behind the sun, after the observation of those stars it gave evidence for Einstein's theory of general relativity.

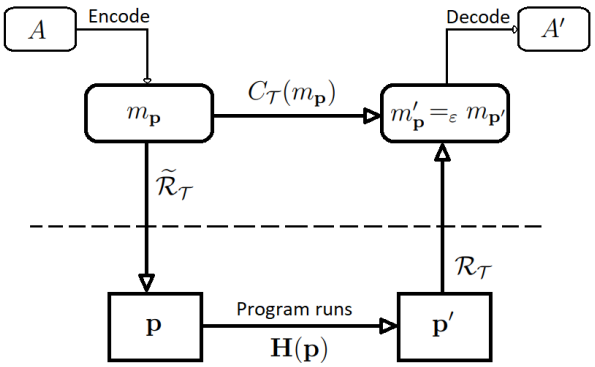


Figure 2.4: *Framework for categorizing how computation is performed by a physical system. Below the - - - - dashed line is the physical system that undergoes evolution. Above the line is the abstract model that changes after the rules for dynamic behaviour has been applied. This image is inspired on figure 3 in the paper [84].*

For a computer, which is a physical system, we want a way to provide it with the necessary information and that the system evolves such that at the end process the information has changed in a satisfying and deterministic fashion. The information lives in the realm of abstraction and while our model may be able to use the information directly, our physical system does not, because it does not know what a number is. Thus in our model, we have to include a way to encode our information into the physical system itself. In the section *representing information* we have seen some forms of encoding information. After the physical system has processed the information, we interpreted a quantity in the system, state or some parameter from the system as the processed information. Thus we use our model $m_{\mathbf{p}'}$ to decode the processed information in a way that makes sense to us, the end users of this computing system. The form we encode information and the form how we decode does not necessary needs to be in the same form, what is important is that we have a consistent way of encoding and decoding information throughout the experiments. The information outcome we obtain through the decoding after an experiment, does partly depend on the physical system. The system may undergo the same physical evolution, but still is being associated with a different mathematical operation, that is because the outcome also depends on the decoding scheme we are using. A simple example, let us say that we swap the decoding scheme for the representation of $0 \to 1$ and $1 \to 0$. We would then change the OR operation with NOR, AND with NAND, and XOR with XNOR, if our experiment only contained Boolean operations. As said before, the form the information encoding and decoding can be different, this was done in an NMR computing set-up [18]. The first input was encoded as a phase difference and the second as time delays in the initiating pulse signal, the output was decoded from the integrated spectral intensity obtained from the bulk molecules in the experiment sample. Going back to the framework, a physical system will evolve through time, even when it is not initiated by the user. Therefore to have a system that would be meaningful for performing computations it is important that there is intention behind the evolution from the user, together with some form of encoding and decoding information. Physical systems all around us evolve but viewed from this framework, only when information has been encoded and we can decode the information after the system has evolved in a predetermined way, we can argue that a physical system has performed any kind of calculation. When we have a theory of

the systems computation capabilities that has been robustly been tested and so far has not been invalided, we are then able to create building blocks for computing with this system. This is the argument by Horseman *et al.* For electronic computers, those building blocks are the electronic integrated circuit chips that are only able to perform Boolean operations. By combining chips, a more complex form of computing can take place. For those electronic chips robust theories do exits and $m_{\mathbf{p}} =_{\epsilon} m_{\mathbf{p}'}$, thus we can work with them in confidence and do not need to test those theories every time to check if figure 2.4 still holds. This is also the goal of creating novel ways of performing operations with a physical system. The physical system most concerning this thesis are the solitons that travel inside the Toda lattice. In the next chapter, we will be discussing the model for the Toda lattice, how encoding and decoding have been used to realize logic gates in this physical system. The next section will introduce the concept of solitons. It is after the coming section that we are reviewing the literature on logic gate implementation done in soliton supporting mediums. Horseman *et al.* [69] summarize their work with following definition of physical computing: "*the use of a physical system to predict the outcome of an abstract evolution*".

2.2 Solitons

Waves are everywhere in nature and we come in contact with them daily, such as in the form of sound or light. They transport energy from one location to another and two categories of waves are linear and nonlinear. An example of linear waves are the pitches on a musical instrument and a way to express linear waves are done with travelling harmonic waves

$$y(x,t) = A\,\cos\Big(\frac{2\pi}{\lambda}(x - vt) + \phi\Big), \tag{2.1}$$

where x and $y(x,t)$ are spatial coordinates, A the amplitudes, λ the wavelength, t is time, v the waves' velocity and ϕ is the off-set. Linear waves are characterized by the following properties:

- The sum of two linear waves is another linear wave.
- The waves' shape and velocity are independent of the waves' amplitude.

The first property can also be phrased in another way, the sum of two solutions of a linear wave equation, is another solution to that same equation. This is called the *superposition principle*. We have drawn in figure 2.5 the sum of two linear waves.

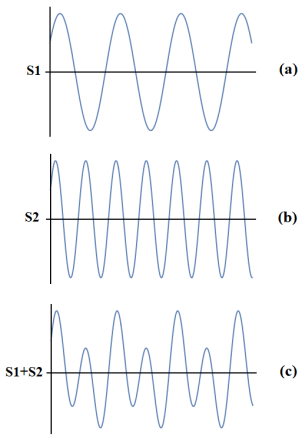


Figure 2.5: *Two linear waves and the sum of both waves. (a) cos* $(2\pi(1-t))$*, (b)* $2cos\ (2\pi(1-2t))$ *and (c) showing the sum of (a) and (b).*

The superposition principle does not generally hold for nonlinear waves and for many nonlinear wave equations this means that a unique method needs to be developed specifically for solving the individual equation equation. Unlike linear waves, nonlinear waves exhibit linear dispersion and nonlinear effects. Nonlinear effect is when the top part of a wave moves faster than its sides, creating the effect that can be seen in figure 2.6. Another effect that is seen in nonlinear waves is that the wave peak becomes more narrow over time. The linear dispersion, on the other hand, causes the wave to flatten its form over time, this is generated by the longer wavelength components that travel faster than shorter wavelength components, see again figure 2.6. It is when both dispersion and nonlinear effect are balancing out that we have a special class of waves, which is the focus of this thesis, the so called solitary waves, also known as a soliton.

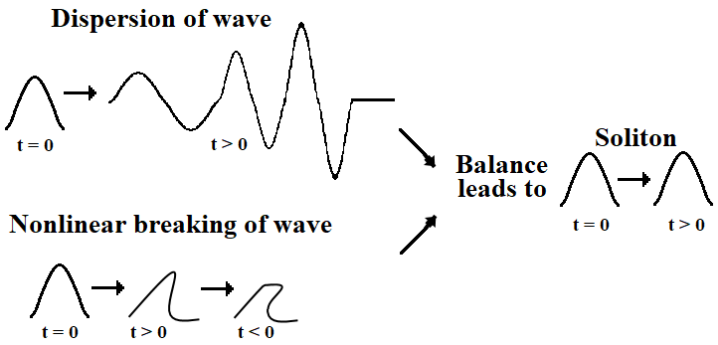


Figure 2.6: *Waves dispersion and nonlinear effect are shown and how they are balanced to maintain its shape over time in the solitary wave. Solitary waves are also called solitons due to their pseudo-particle behaviour. This image is inspired on a figure from Lomdahl's paper [97].*

Not all nonlinear equations have soliton of solutions, thus not all nonlinear systems allows for solitons to exist. The famous Korteweg-de Vries (KdV) equation [97, eqn.5] is one such nonlinear equation that has soliton solutions. The equation is given by

$$u_t + u_{xxx} + uu_x = 0. \tag{2.2}$$

were u is the amplitude of the wave, which depends on the time t and the spacial coordinate x. The shorthand u_t has been used for $\frac{\partial u}{\partial t}$ and $u_x = \frac{\partial u}{dx}$. When we separate the above equation into two parts, we have linear dispersion part $u_t + u_{xxx} = 0$ called the Airy equation, sometimes called the linear Airy equation to differentiate it from $u_x x \pm \mathrm{con}^2 xu = 0$, which is also called the Airy equation in the literature. And the second part is the nonlinear effect, $u_t + uu_x = 0$ which is called the nonlinear transport equation and also called the inviscid Burgers equation, to differentiate it from the Burgers equation $u_t + uu_x = \mathrm{con}\ u_x x$. The solution to the inviscid Burgers equation will show the breaking of waves as seen in figure 2.6. Those two parts together balance each other out to make soliton solutions possible. One soliton solution for the KdV equation is given by [97, eqn.6]

$$u(x,t) = 3c\ \mathrm{sech}^2\Big[\frac{\sqrt{c}}{2}(x - x_0 - ct)\Big] \tag{2.3}$$

were c is the velocity of the soliton and x_0 its initial location. x_0 has been included in the equation above to show that the difference between the current and initial location is used in the solution, which is implicitly done in [97, eqn.6]. The soliton is shown in figure 2.9(a), where the height is $u(x,t)$ and on the horizontal axes the spacial coordinate x. Unlike the waves that have been drawn in figure 2.5, a soliton is located in one region of space and their behaviour under collision is analogous to particles [97, 104], making them pseudo-particles. It is their pseudo-particles behaviour that makes them interesting, among other things, for their applicability as information carriers. Solitons are commonly characterised [104] by three properties: (1) localized within a region, (2) have a permanent form at every moment in time and (3) its shape stays unchanged after collision with another soliton, only the solitons phase is changed. In figure 2.7 we see the collision of two solitons. What is characteristic of solitons is that after a collision the current position is shifted somewhat compared to the location they would have been in

without any collision. In equation 2.3 if we had introduced a phase φ in the $\text{sech}^2[...]$ term, then after the collision the value of φ would have been changed. From the figure, the particle behaviour can already be seen and we will illustrate this better with an analogy of two tennis balls colliding [104], shown in figure 2.8.

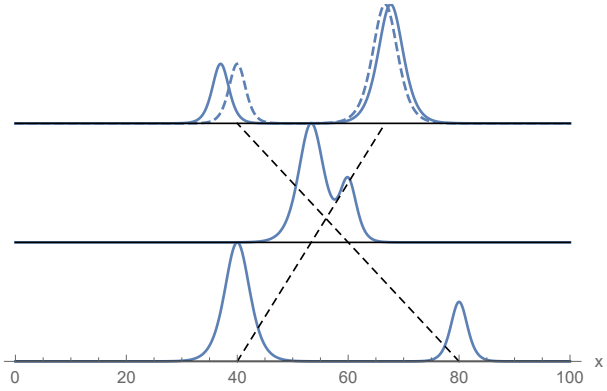


Figure 2.7: *Two solitons colliding. Three moments in time have been drawn, the - - - - dashed line indicates the original trajectory, the height of the soliton is the amplitude. This image is inspired on a figure from Andrei's book [104].*

One tennis ball moves with velocity $\mathbf{v}_1$ and another with $\mathbf{v}_2$ in figure 2.8. O indicates the position of the centre of mass for both balls, O_1 and O_2 are the position of the two tennis balls. The surface of a tennis ball is elastic and more flexible than a billiard ball, meaning that after the collision it will take some time before the velocities of the balls are swapped, they will first have to squeeze before expanding and restoring to their original form during the collision. They then continue the paths set out by the other ball. The current path is somewhat shifted compared to their original trajectory, this is analogous to the solitons experiencing a phase shift.

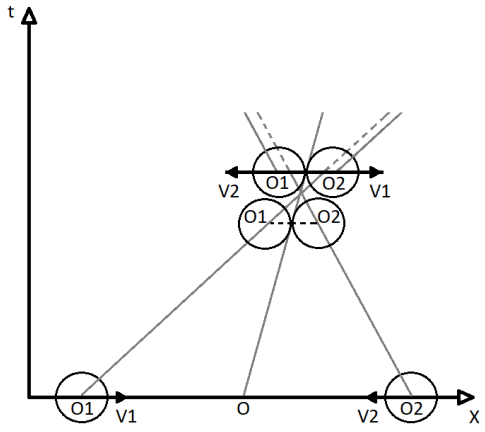


Figure 2.8: *Collision of two tennis balls, used as an analogue for a soliton collisions. The - - - - grey dashed line indicated the original trajectory of the tennis balls. This image is inspired on a figure from Andrei's book [104].*

There are three common type of solitons, the Korteweg-de Vries (KdV) type, topological and envelope soliton types [104]. The equation most associated with KdV soliton is, of course, the KdV equation itself, for topological soliton is it the sine-Gordon (SG) equation

and for the envelope soliton, it is the nonlinear Schrödinger (NLS) equation. The KdV soliton are characterised by their amplitude $u(x - vt)$, that increases with an increase of the velocity v and vice versa. Therefore it is not possible to have any static KdV soliton type solution. In figure 2.9(a) is shown the soliton of the KdV equation, the width of the wave is proportional to the square root of its velocity. The KdV equation is used to describe among others for the behaviour of shallow water. Soliton solutions in the Toda lattice, which we will be discussing in chapter 3 belong to this soliton type. For optical fibre, the main equation is the NLS equation and the soliton solutions are characterised by their amplitudes which depends on the width of the wave and this type of soliton its velocity is independent of its amplitude. The soliton is shown in 2.9(b) and it is the envelope of this wave which is the true soliton. There are two types of envelope solitons and they are named bright and dark solitons, named this way because the bright soliton shown in 2.9(b) corresponds to a pulse in light and the dark soliton corresponds to a dip in the amplitude of the light wave. The SG equation gives rise to solitons that can be static in its movements, because the amplitude and velocity are not depended on each other. A soliton solution of the SG equation is shown in figure 2.9(c), the width of the wave gets narrower the more the velocity increases, causing the slope of the amplitude to become steeper. The SG has been used in a wide range of applications in physics [44], from rigid pendula to a junction made of two superconductors called the Josephson junction. There are different kind of topological solitons, such as the so called *kink* solitons shown in 2.9(c), and *antikink* solitons which moves in the opposite direction to kink solitons. Then there are also the *breathers*, which have a different appearance than the kink and antikink solitons, and are destroyed by any kind of perturbation in the system.

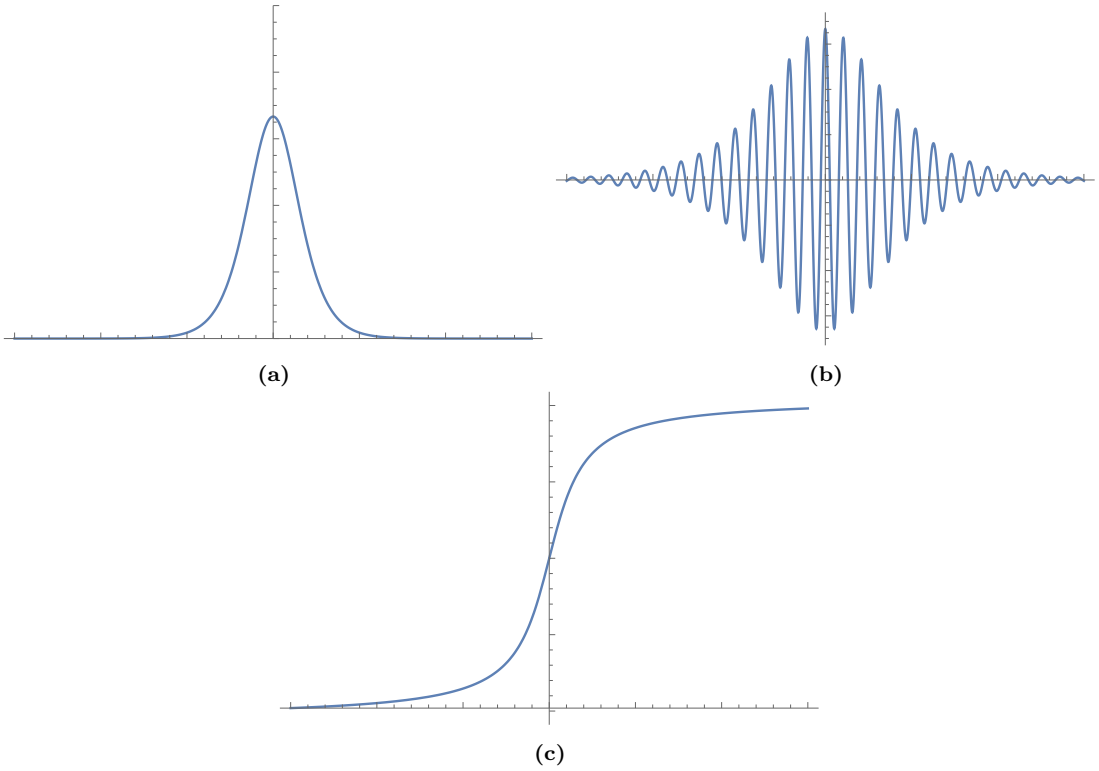


Figure 2.9: *Different kind of solitons. The horizontal axes is the spacial direction and the vertical axes is the amplitude of the solitons. (a) shows a soliton solution from the Kortweg-de Vries equation, (b) from the nonlinear Schrödinger equation and (c) from the sine-Gordon equation.*

Now that we are familiar with a framework that tells us when a physical system is actually computing and with the concept of solitons, we are in a position to review the soliton literature for computing.

2.3 Computing with solitons

In the introduction, we have already tipped on the uses of solitons in fibre communication networks and the goal of realizing an all-optical computer, furthermore that there are benefits for using solitons as information carriers in an optical computer. One of the first applications of solitons being used in communication was the development of switches [38, 51, 52, 119]. They operated on preventing or allowing the flow of solitons, which is the stream of information and this is analogous to a switch in the *on* or *off* state. One of the earliest works on creating functioning Boolean operators using solitons was done by Islam *et al.* [3–5, 15, 28–36, 48–50, 67, 73–80]. He and his colleagues started working on developing and experimenting on Boolean logic gates from 1989 till around 1996. We will shortly be discussing some of their works. Research on soliton logic gates has not stopped with Islam two decades ago, but is still being conducted to this day [16, 25, 54, 65, 95]. Different research groups have used different representations and have used different set-up for realizing primarily Boolean logic gates. We will be going over some recurring approaches for both the representation and logic gate designs. Some researchers in the field were able to go one step further than logic gates, they were able to build a functioning half- [13, 21, 57, 58, 115, 122], full-adders [22] and flip-flops [128].

2.3.1 Representing information with solitons

In section 2.1 we started with examining different representations that were used for defining information in computing. We make the same start in this section. We, therefore, start our review of the soliton computing literature, by looking at how the encoding and decoding of information have been done with solitons. The most common binary value representation used in this field of research is the absence and presence representation. The presence of a soliton is sometimes defined as the amplitude of a soliton that is above a certain threshold. This representation will be used in the design of our logic gates in the Toda lattice in chapter 4. The absence of a soliton within a certain space is often representing the value 0 and the presence the value 1. Logical operations can then be created by allowing or preventing the access of a soliton in a specific area space. Another option is by lowering or increasing the amplitude of a soliton. Using the height of the amplitude as a way to represent the binary values is analogue to electronic logic gates, where the value of the electronic potential difference (voltage) is used to represent binary values.

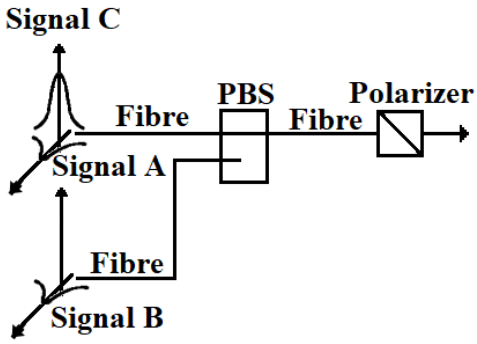


Figure 2.10: *Soliton logic gate design through dragging. The two inputs signals are A and B, and C is the control signal. This control signal is influences by the presence of A and B, which are both orthogonal to the control signal. PBS stands for polarized beam splitter. This image is inspired on a figure from Islam et al. paper [67].*

Let us look at two examples of how the encoding and decoding is being done with solitons. The first example is from Islam *et al.* [67] and their soliton dragging set-up, drawn in figure 2.10. To realize the encoding they have one control signal, which is a soliton and is labelled as signal C in their paper and in the figure. There is one input which uses the same fibre as the control signal, but its signal (signal A) is orthogonal to the control signal. The other input (signal B) can only interact with the control signal when both signals arrive at a polarizing beam splitter around the time. When signal A is present, it will slow down signal C until it reaches the polarizing beam splitter. When signal A is absent, signal B is timed such that it will simultaneously arrive at the polarizing beam splitter with signal C, also influencing the time that C will arrive at the detector. This covers the encoding of information, the presence or absence of the signals A and B representing the values 1 and 0 respectively. The decoding is realized in this particular set-up by looking at the moment the soliton arrives at a certain point in the set-up. There is an internal clock and when the control signal C arrives outside the predetermined moments in time (clock windows), the signal will be associated to the value 1, and when the soliton is within the window, will represent the value 0. With this set-up, the signal C is always enough delayed with the presence of either or both signal A and B, that it will arrive outside the clock window. Thus the authors were able to design a NOR gate. If the representation of the decoding had been the other way around, they would have realized an OR gate instead. This is an example of how the decoding can influence which logic operation we associate with the physical evolution, even though physically nothing has not been changed between the two choices of representation. In this whole set-up, the amplitude of the control signal did not change by the presence of the other signals, for the next example, this will be different.

The second example is more relevant to the main focus of this thesis, namely the realization of logic gates inside the Toda lattice. The possibility was demonstrated by Kubota and Odagaki [91], when they created a NOT, OR and AND gate. Two lattices were used as inputs and they are connected in a Y shape to a third lattice, functioning as the entrance for the logic operation(s) and as the output. By initializing a soliton in one of those lattices, it represented the value 1, when a lattice was used as input without an initialized soliton, this represented the value 0. The amplitude of the solitons in either lattice will always be the same. Most soliton systems do not allow static soliton solutions, therefore most solitons are always moving and the KvD type solitons in the Toda lattice are no difference. This makes it harder to maintain a constant input value over time, in the same way as it is possible to do so with electronics. When a soliton enters the end part of this third lattice, if the amplitude was above a certain threshold, the outcome represented the output value 1. However, when no such soliton entered the end part of the third lattice or there was no soliton at all, the outcome represented the value 0.

2.3.2 Performing operations with solitons

We will postpone a discussion about the logic gates that have been designed for the Toda lattice until section 3.3. In this section, we will be going over other implementations of solitons for the sole goal of computing. One common way of realizing logic gates in physical systems are with collision-based computing, also commonly found in the uncon-

ventional literature with other systems. In collision-based computing there are no wires as we have discussed in section 2.1.2 with the billiard ball model. The signals are free to move in any direction inside the system environment. For a system with soliton solutions, the signal can be solitons. Because there are no wires, the solitons can travel in any direction, this entails that every location in the medium can be used by the soliton. Information that a soliton carries is then manipulated by colliding solitons with another soliton or influenced by inserted material in the homogeneous medium. This can cause the soliton to have a phase shift or a change in the soliton's amplitude. A possible approach for representing logical values is by associating the values with certain locations in the medium where the signals will arrive after it travels through the medium itself. Logic gates can then be constructed by colliding solitons with each other or inserting material that changes the solitons' directions. This approach can be done in systems where solitons repel each other when they come close to each other, thus changing their direction in the system. The output values can then be associated with the position a soliton enters at the end of the medium. This method has been used by Wu [127], Scheur and Orenstein [110], and by Bakaokas and Edwards [14]. Another example can be found in the work of Islam *et al.* [67], which we have already discussed in the previous section with the set-up shown in figure 2.10. In this work, the solitons always arrive at the same location, but their phase shift is changed by the collision of two solitons, causing the control soliton to arrive at a different time slot. If the control soliton does not collide with another soliton during its course, it will arrive at a particular time slot which associated with the value 0. When the control signal is influenced by two soliton collision it will arrive at a different time slot, which is associated with the value 1.

Another set-up for realizing logic gates is the spin ladder. This ladder is made of two lattices, those lattices are made of connected objects with spins, those can be molecules, electrons or magnetic dipoles, and those two lattices are then connected to form the spin ladder. Figure 2.11 shows the spin ladder set-ups. The spin ladder set-up by Veerakumar [125] and Kavitha *et al.* [66] are made from a ferromagnetic material, which is material that is magnetic even in the absence of a magnetic field. Two ways of building a spin ladder is by having the spin of both coupled lattice pointing in the same direction, this is ferromagnetic, or pointing in opposite direction and balancing each other out, this is anti-ferromagnetic. The authors of both works have chosen for a ferromagnetic design. Kavitha *et al.* constructed an isotropic spin ladder, meaning that the ladder is the same in both directions, while Veerakumar constructed an anisotropic spin ladder.

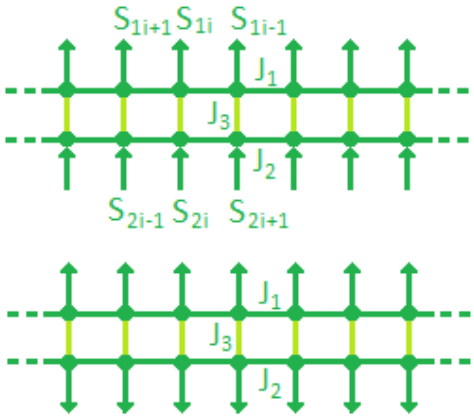


Figure 2.11: *A spin ladder made by coupling two spin lattices together. Two ferromagnetic coupled spin lattices are coupled together. The above image is ferromagnetic and the one below is the antiferromagnetic spin ladder. J_1 is the coupling constant of the first chain, J_2 of the second chain and J_3 is the value of the coupling constant that connects both lattices. This image is inspired by a figure found in the work of Kavitha et al. [66].*

In Veerakumar's work [125], the solitons are initialized in the ladder itself and two solitons are moving towards each other. With certain set of parameter values, both solitons will collide with one another and only their phases are changed by this collision. But by changing the set of parameter values to specific values, the solitons will merge after colliding. The Boolean logic gates can then be constructed by representing 0 if the merged soliton is absent or below a certain amplitude threshold and 1 is represented by the presence of a soliton and have an amplitude above a certain threshold. This approach is different from the work of Kavitha *et al.* [66], where solitons are initialized in one of the spin lattices. Gates are constructed by changing the value of the coupling constant that connects both spin lattice, this constant is J_3 in figure 2.11. By changing J_3, the outcome of the solitons interaction is changed too. The same representation as Veerakumar was used in this work.

There are many other set-ups created with soliton supporting physical systems, such as colliding dark and bright solitons, while those two different soliton solutions represents the Boolean values [101]. Another example being a mixture of photonic crystal material, which is material that allows the photons to be guided in a certain direction, the Boolean values for the in- and output were represented the presence and absence of a soliton [54]. They were able to build an AND gate, because only when both inputs had received a soliton, was there a soliton produced at the output. Those two examples, together with the ones already discussed in this section, are only a few examples of the many researched approaches. In this thesis, we will be focussing on the Toda lattice. We have introduced in this chapter the concept of representing information with physical quantities, states or parameters, those have then been applied in varies designs for realizing logical operations. We have gone over the criteria that needs to be met before a physical system can be categorized as a system that is performing the computation. Within this chapter, we have discussed what a soliton entails in the details sufficient for this thesis and we have seen some of the possible constructions of logic gates when it comes to computing with solitons. In the next chapter, we will be introducing the theory of the Toda lattice and an approach of delaying the soliton inside the lattice. This chapter and the next are introducing the background and theory necessary for the core of this thesis.

Chapter 3

Toda lattice theory and delay

In this chapter we are going over the theory of the Toda lattice, how soliton solutions from the Toda Equation of Motion travel through this lattice and how they are delayed in their movements. In the next chapter, we will implement the theory to create a controllable delay and design and test new logic gates designs in the lattice. The main source for this chapter's information on the Toda lattice theory comes from Toda's book *Theory of Nonlinear Lattices* [124] and we will be using the same notation throughout this chapter as was used in the book.

3.1 Toda Lattice

A Toda lattice is constructed from sites that are connected with their nearest neighbour under the Toda potential. Every site has the mass m. The lattice can have any number of spatial dimensions, but we will start with the simplest case of one spatial dimension. One dimension is enough to create some of the mechanics we will be using in the next chapter and we will not need to go further than two spatial dimensions for constructing logic gates in the lattice. The Toda lattice is an abstract model, but certain behaviour of physical systems can be emulated with the model and the lattice is drawn in figure 3.1. The model can be directly simulated on a specific electronic circuit called the LC ladder network [92, 124] and it is made from capacitors, inductors and resistors. The Korteweg-de Vries equation can be obtained from the Toda lattice equation of motion (e.g.,[61, 85, 124]), by taking the continuum limit of the lattice, that is by changing the distances between sites towards zero and adding sites such that the lattice becomes a continuous line. A complex variable extension of the Toda lattice has been shown to be equivalent to a simplified version of the nonlinear Schrödinger equation [11]. Furthermore, a Toda lattice where the zeroth site is constrained in its movements, these soliton solution set-ups were shown to be equivalent to the soliton solutions of the nonlinear Schrödinger equations [83]. There are therefore a variety of relations between the Toda lattice model and behaviours found in nature.

Figure 3.1: *A Toda lattice made from sites with equal mass m, equal separation L, connected by Toda potential.*

In the lattice every site is placed a distance $\mathbf{L}$ from the neighbouring sites, this is the rest length when the lattice is in equilibrium. The approach is the following, the nth site is placed at position $\mathbf{x}_n$ and the $(n+1)$th site placed at a distance $\mathbf{x}_{n+1}$, this provides us with the following relation in a lattice in equilibrium

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{L}, \tag{3.1}$$

When the sites are not in their initial positions, the difference between the x_n and x_{n+1} sites' current length and the rest length is given by.

$$\mathbf{r}_n = \mathbf{x}_{n+1} - (\mathbf{x}_n + \mathbf{L}), \tag{3.2}$$

where $\mathbf{r}_n$ is called the relative displacement. When the current length is equal to the rest length, the relative displacement is zero, as it should be. Because we are working in one dimension at the moment, the vectors become scalar values. The above expression then becomes

$$r_n = x_{n+1} - (x_n + L). \tag{3.3}$$

The advantage of writing r_n in the way described above, is that we can place the origin at any place we desire. Additionally, when we change the site's coordinates, it will not affect the value of r_n. Looking at the above formula we see that when $r_n < 0$ the lattice is compressed. It becomes easier to see this when we rewrite the above expression

$$x_{n+1} - (x_n + L) = r_n < 0 \Rightarrow \tag{3.4a}$$
$$x_{n+1} < L + x_n. \tag{3.4b}$$

The reverse $r_n > 0$ implies that $x_{n+1} > L + x_n$, thus the lattice is stretched. The reason for r_n being called the relative displacement in the literature can be seen when we look at the displacement of the site positions. We write the current position of a site at n as x_n and the initial position as x'_n. This than gives us an expression for the sites' own displacement δx_n based on the initial position, which is equal to

$$\delta x_n = x_n - x'_n. \tag{3.5}$$

Writing it in another way $x_n = x'_n + \delta x_n$. The value of r_n is then given by

$$r_n = x_{n+1} - (x_n + L) = x'_{n+1} + \delta x_{n+1} - (x'_n + \delta x_n + L) = \delta x_{n+1} - \delta x_n, \tag{3.6}$$

Therefore r_n is the difference between two displacements, in other words the relative displacement between two sites. If the entire lattice is shifted the same amount in the same direction, that would mean that $\delta x_{n+1} = \delta x_n$, thus $r_n = 0$, which has the same effect as shifting your origin. Now that we have a lattice, it is time to introduce the forces that holds the sites together. The Toda potential is given by the following function

$$\phi_{T,n}(r_n) = \frac{a}{b} e^{-br_n} + ar_n + \text{constant } (a, b > 0), \tag{3.7}$$

where a and b are constants. The unit of a is given as the energy per length [J/m], which is equivalent to a newton [N]. While the unit of b is the reciprocal of the length [m^{-1}].

The force felt on the sites exerted by the Toda potential can be found by taking the spacial derivative of the potential

$$f_{T,n}(r_n) = -\frac{d\phi_{T,n}(r_n)}{dr_n} = a(e^{-br_n} - 1). \tag{3.8}$$

Below in figure 3.2 we have plotted the force, from $r_n = -10$ to $r_n = 0$ and $r_n = 0$ to $r_n = 10$. As can be seen in the plot, the force scales very differently in both ranges. Between $r_n = -10$ and $r_n = 0$ we see that the e^{-br_n} term dominates, whereas in the other range the -1 term dominates. Bringing this back to the lattice, we see that there is a much bigger force when the sites are compressed than when they are stretched apart. Situation of analogues to a spring would be of a spring that is very stiff when it is being pressing together, but relatively easy to stretch out. Furthermore the force required to stretch the spring even any further becomes independent of the current extension.

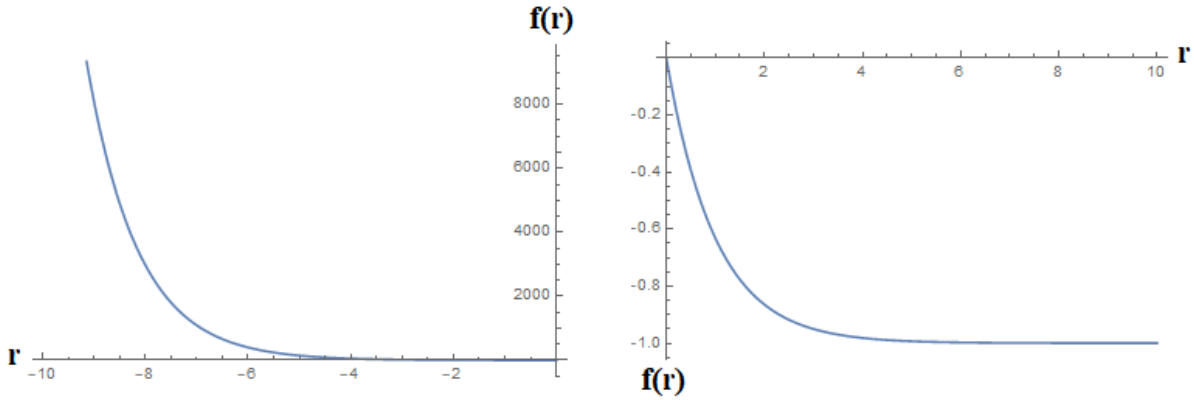


Figure 3.2: *The force as function of r_n for two regions. The parameters have the value $a = b = 1$ in both plots and they show the same function, but are split, allowing the viewer to observe that the scale of both regions are very different. Note the very different y axes scales in both figures.*

In figure 3.2 we have plotted the graph for the case $a, b > 0$. But if $a, b < 0$ we would see that term 1 will dominate in the range $r_n < 0$ and the term $-e^{br_n}$ the range $r_n > 0$. This means that the force between two sites is constant when we shrink the distance. When we increase the distance, a strong force will try to keep the sites close together. Throughout this thesis the values for the constants will be $a, b > 0$, because this thesis uses research that has chosen this values. The results of the controllable delay in the next chapter would be different if we choose $a, b < 0$.

In the next section, we will need the harmonic oscillator potential and the associated force, therefore, we are introducing them in this section. In our daily lives we come in contact with many physical objects that behave as harmonic oscillators, such as springs found beneath the keys of our keyboards and harmonic waves are used in radio stations to generate radio waves. The harmonic oscillator potential is given by Hooke's law

$$\phi_{h,n}(r_n) = \frac{\kappa}{2} r_n^2. \tag{3.9}$$

where κ is a constant, most often with the constant for a spring. The force exerted by

this potential is

$$f_{h,n}(r_n) = -\frac{d\phi_{h,n}(r_n)}{dr_n} = -\kappa r_n. \tag{3.10}$$

As we can see the force of the harmonic oscillator is linear, rather than the nonlinear behaviour generated from the Toda potential we saw earlier. For the Toda lattice, an analogy of the spring constant κ would be the combination of ab. We find this combination when we look at the first order Taylor expansion of the Toda potential, which is

$$\phi_{T,n}(r_n) = \text{const.} + \frac{ab}{2} r_n^2 \tag{3.11}$$

3.2 Installing a soliton in the lattice

Now that we have a functional form for both the Toda and harmonic potentials, we can create an equation for the Toda lattice to initialize a soliton inside it. To start with, we would like to know how much force exerted is required on the individual sites such that the creation of a soliton is possible, which would travel through the Toda lattice. The equation of motion for the lattice is given by

$$m\frac{d^2 r_n}{dt^2} = \phi_T'(r_{n+1}) - 2\phi_T'(r_n) + \phi_T'(r_{n-1}), \tag{3.12}$$

where the notation $\phi'(r_n) = \frac{d\phi(r_n)}{dr_n}$ is used. The simplest soliton solution that satisfies this equation of motion is given by the following expression for the force

$$f_n(r_n) = \frac{m\omega^2}{b}\text{sech}^2(k_0 n \pm \omega t), \tag{3.13}$$

which is the force felt on every site at every moment in time while a soliton travels in the lattice. In the above expression ω is defined as $\omega = \sqrt{\frac{ab}{m}}\sinh k_0$, where m is the mass, and k_0 is a parameter of the soliton called the wavenumber. For a left moving solitary wave, we take the + solution, and the − solution will give us a right moving wave. The speed of a soliton in the lattice is given by

$$v = \frac{\omega}{k_0}. \tag{3.14}$$

In figure 3.3 we have plotted the speed against the wavenumber for $a = b = m = 1$. As we can see in the figure, increasing the wavenumber increases the speed of a soliton moving in the lattice.

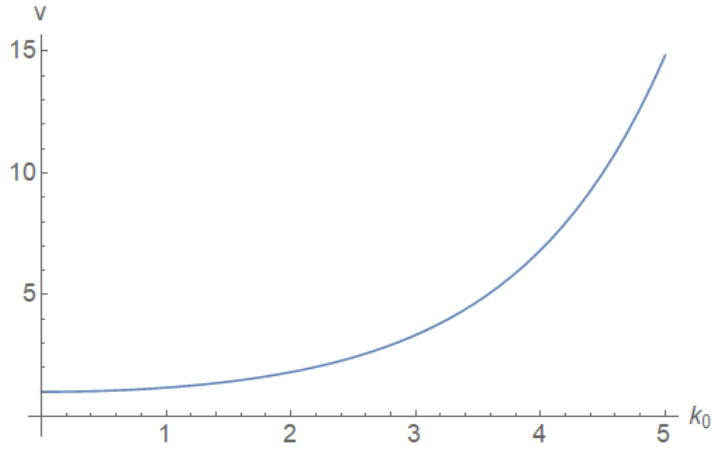


Figure 3.3: *The speed of a soliton plotted against the wavenumber. Equation 3.14 is plotted with the values* $a = b = m = 1$.

The energy of a soliton is given by

$$E_0 = \frac{2a}{b}\big(\sinh\, k_0 \cosh\, k_0 - k_0\big). \tag{3.15}$$

The energy is plotted against the wavenumber in figure 3.4. We see again that the energy increases when the wavenumber increased. In the next section, we will be using this equation together with the expression for the speed.

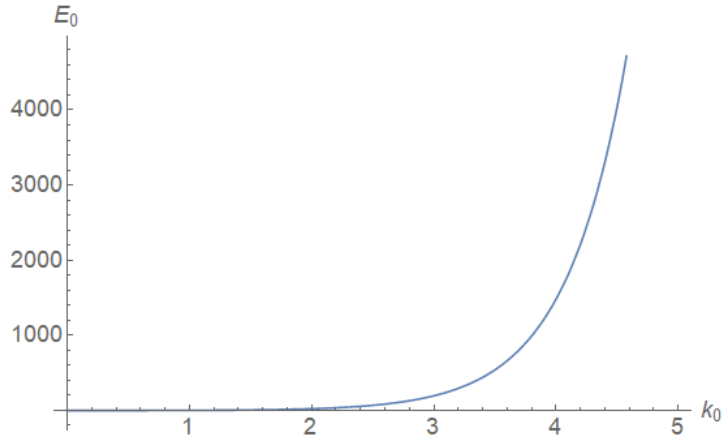


Figure 3.4: *The energy, in Joules, of a soliton plotted against the wavenumber. Equation 3.15 is plotted with the values* $a = b = 1$.

For the simulation we would like to prepare a soliton with a certain wavenumber into our lattice and let the program simulate its dynamics. One way of preparing the lattice is to stretch and compress sites inside the lattice in a certain way, a good parameter for doing this would be r_n. To be able to do this we combine expressions 3.8 and 3.13, to get

$$\frac{m\omega^2}{b}\operatorname{sech}^2(k_0 n - \omega t) = f_n = a(e^{-br_n} - 1) \Rightarrow \tag{3.16a}$$

$$r_n = \frac{-1}{b}\,\ln\Big(\frac{m\omega^2}{ab}\operatorname{sech}^2(k_0 n - \omega t) + 1\Big). \tag{3.16b}$$

To initialize a soliton into the lattice we need to determine the position it will start at. Let the starting position be at $n = n_c$. The next step is to eliminate the time dependency in the above expression and find a way to incorporate the starting location. We can do this by searching for the moment at which the soliton has its peak at position n_c. To do this we need to find the maximum value of a function. This can be found by taking its derivative and placing that equal to zero, then rewrite the obtained expression to find an expression to find the time t_c were the peak of the soliton is at site n_c.

$$0 = \frac{df_n(n = n_c, t = t_c)}{dr_n} = \frac{d}{dr_n}\frac{m\omega^2}{b}\text{sech}^2(k_0 n_c \pm \omega t_c) \tag{3.17a}$$

$$= \mp \frac{2m\omega^3}{b}\text{sech}^3(k_0 n_c \pm \omega t)\sinh(k_0 n_c \pm \omega t_c) \Rightarrow$$

$$0 = \sinh(k_0 n_c \pm \omega t_c) \Rightarrow \tag{3.17b}$$

$$0 = k_0 n_c \pm \omega t_c \Rightarrow \tag{3.17c}$$

$$t_c = \mp \frac{k_0 n_c}{\omega}. \tag{3.17d}$$

Thus the peak of the soliton is at positon n_c at time $t_c = \mp\frac{k_0 n_c}{\omega}$. We were able to divide both sides by the sech^3 term because this function is never equal to zero, which is not the case for the sinh function. The choice of the + and − solution will depend if we want to have a left or right moving soliton. Plugging this into 3.16b and we have

$$r_n = \frac{-1}{b}\ln\Big(\frac{m\omega^2}{ab}\text{sech}^2(k_0(n \pm n_c)) + 1\Big). \tag{3.18}$$

In our simulation, we will not be directly using the relative displacement, but we will be using the current position of the sites x_n, therefore to initialize our soliton we need to rewrite the expression 3.3 into

$$x_{n+1} = r_n + x_n + L, \tag{3.19}$$

Combining this expression with 3.18 and we obtain an expression for the position of every site in our lattice when initializing a soliton

$$x_{n+1} = \frac{-1}{b}\ln\Big(\frac{m\omega^2}{ab}\text{sech}^2\big(k_0(n - n_c)\big) + 1\Big) + x_n + L. \tag{3.20}$$

When we write it in this form, we need to specify the position of the first site, because x_{n+1} is a function of x_n. Let's call this first site n_s and it will be placed at position x_s =constant that is some constant value on the x-axis that does not depend on another site's position. From there all the site positions go up, the next site is placed at x_{s+1}. In figure 3.5 we show the position of the sites when a soliton moves in the lattice that has been initialized by equation 3.20.

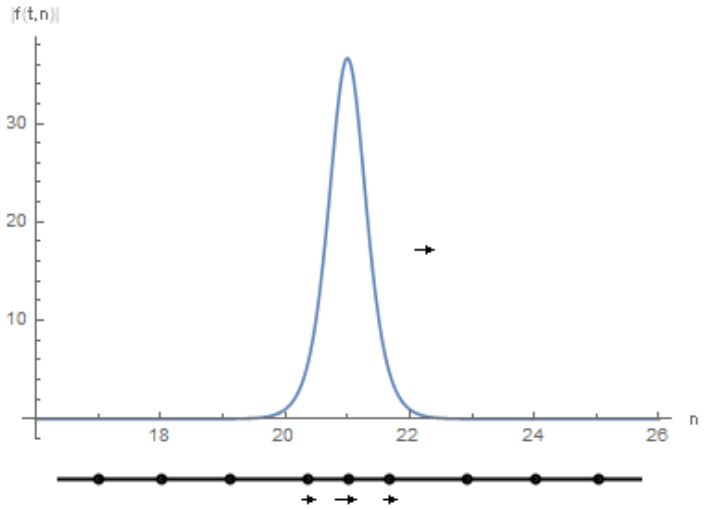


Figure 3.5: *A soliton moving to the right, below the graph is an indication of the position of the sites in the Toda lattice. The images shows the force vs positions for a soliton. The graph shows a continuous line, however sites in the lattice are discrete, thus it will be a discrete line when we create a graph in the simulation. Here a continuous line has been drawn to make the visualization of a soliton easier.*

3.3 Temporary trapping a soliton

Now that we can initialize solitons, we are going over the phenomenon of delaying the solitons in its travels. We will need this phenomenon in the next chapter when we are going to design logic gates in the lattice. In 2005 Kubota and Odagaki [89] investigated what would happen to the transmission of a soliton when it crosses a single harmonic potential in the Toda lattice. They found that depending on the harmonic potential constant, the soliton might travel unhindered or the amplitude decreases. The decline of the amplitude is caused by the harmonic potential being off-resonance with the soliton. The approximated formula they found is the harmonic potential constant as a function of the solitons wavenumber, which is the value that allows the soliton to travel unhindered across two sites connected by the harmonic potential. The formula is given by

$$\kappa_{max} \simeq \frac{\sinh^2 k_0}{\ln(1+\sinh^2 k_0)}. \tag{3.21}$$

In 2006 [90] the same authors found that with two or more sites connected by the harmonic potential, it is possible to trap a soliton in the Toda lattice for a certain period. The time that the soliton was trapped depends on the harmonic potential constant, if the connection between two sites is 'rigid enough', the soliton can pass through the series of harmonic potentials unhindered. We can think of this as being equivalent to two sites being connected by a metal rod. When the constant is lower, the sites connected by the harmonic potential will follow the motion of the Toda potential with some delay, causing the soliton to be trapped for a certain time. The series of harmonic potentials can be seen as an impurity in the Toda lattice, as the impurity only behaves as the first approximation of the Toda potential. While the soliton is trapped in the impurity, it will oscillate and dissipating some of its energy, the amount will depend on the value of the harmonic potential constant and the resonance frequency of the soliton. The value of the harmonic potential constant that gives the soliton the maximum trapping period is given by the following approximate formula

$$\kappa_{trap} = \frac{\theta_{harm}^2}{1.6E_0}, \tag{3.22}$$

where Kubota and Odagaki with the aid of simulations found that θ_{harm} was between $3\pi/2$ and 2π. For comparisons, if we initialize a soliton with wavenumber $k_0 = 3.5$, then use equation 3.21, we find that an undisturbed transmission across the impurity will require the harmonic potential constant to be 48.733 N/m. Using the above equation 3.22 when we want to know the longest delay possible for the same soliton, the constant needs to have a value between 0.026 and 0.046 N/m. The phenomena of trapping a soliton temporarily is drawn abstractly in figure 3.6 and a figure created by the simulation is shown in figure 3.7. In the abstract depiction, the height of the soliton is given by the dimensionless normalized 'energy density' function $h(n,\tau)$ and has not been drawn in figure 3.6. We will shortly see what this function entails. In both figures, we see that after some time the soliton T continues its travel in the same direction as the initial soliton I. There is also a reflection soliton created labelled as R.

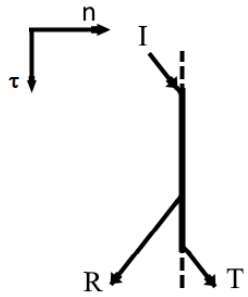


Figure 3.6: *An abstract depiction of a soliton scattering against the impurity. The - - - - dashed line shows the location of the impurity, I is the incident soliton, R the reflection and T the transmission. The horizontal axes shows the sites n in the lattice and vertically is the dimensionless 'time' τ.*

In figure 3.7 some of the energy is lost due to vibrations of the impurity, this can be seen in the figure as small ripples leaving the impurity. There are also waves being reflected, some of those reflected waves are solitons. In the next chapter, we will take a closer look at how to distinguish between the ripples and solitons coming from the impurity. Due to the impurity, the energy of the initialized soliton is transferred into small ripples, transmission and reflection solitons. The energy of transmitted soliton in 3.7 is about 1/3 energy of the original soliton.

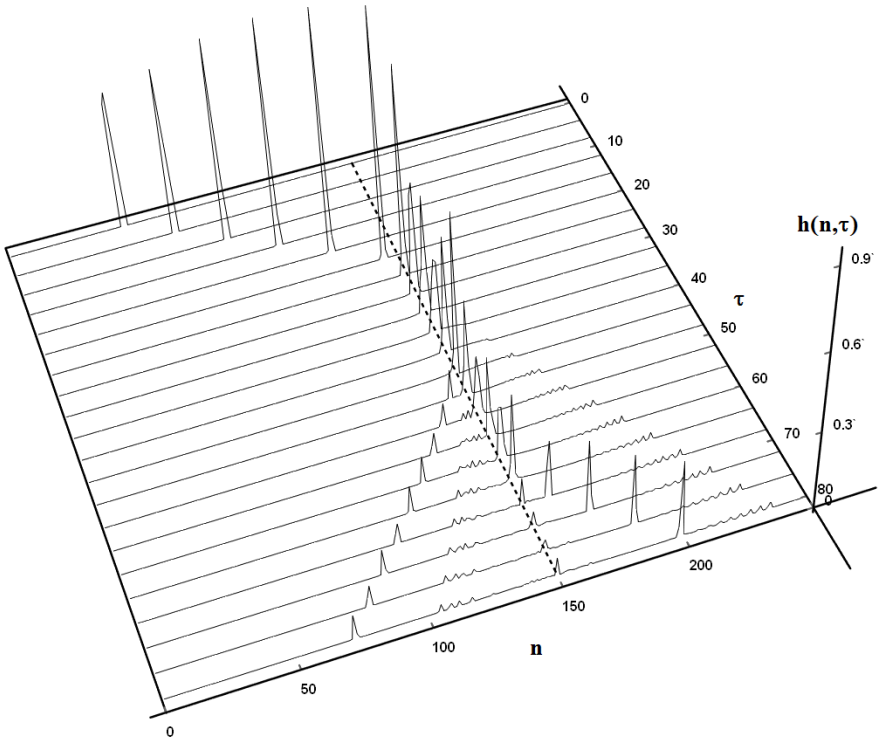


Figure 3.7: *A soliton that is delayed in its travels, the thick - - - - dashed line shows the location of the impurity. The high $h(n, \tau)$ is the dimensionless 'energy' density, n is the label for the individual sites and τ is the dimensionless 'time'. The wavenumber in this simulation was $k_0 = 3.5$ and the constant for the sites connected by two harmonic potential was $\kappa = 0.03$. This figure has been created by the simulation of author himself and this simulation is able to create the same images as has been shown in Kubota and Odagaki's paper [90, figure 5]. A more in depth comparison between Kubota and Odagaki and the authors simulations is included in section A.1.3.*

The dynamic of a system of the physical world can often be encapsulated by an object called the Hamiltonian, and for the Toda lattice, this is not any different. The generic form of the Hamiltonian for this lattice is given by

$$H = \sum_n \left[\frac{p_n^2}{2m} + \phi_n(r_n) \right], \tag{3.23}$$

where p_n is the momentum. We are already familiar with the potential $\phi_n(r_n)$, it is either the harmonic or Toda potential. To show the same results as Kubota and Odagaki, such as shown in figure 3.7, we will connect together the impurity made of multiple sites connected by the harmonic potential, between two Toda lattice. The set-up for the system is drawn in figure 3.8. We will be using this set-up in the next chapter to create an controllable delay lattice and addition logic gate for the Toda lattice. The potential $\phi_n(r_n)$ can be written as

$$\phi_n(r_n) = \begin{cases} \frac{a}{b} e^{-br_n} + ar_n - \frac{a}{b} & \text{for } 0 \leq n < i \text{ and } n \geq j \\ \frac{Kr_n^2}{2} & \text{for } i \leq n < j, \end{cases} \tag{3.24}$$

where the impurity contains $N+1 = j-i+1$ sites in total, each two sites are connected by the harmonic potential, thus the impurity contains N amount of harmonic potentials. In equation 3.7 for the Toda potential we had a constant named *constant*, we are free to assign any kind of constant value for this constant, because the simulation will indeed be indifferent to our chosen values, but to make life easier we chose to set this constant equal to $-\frac{a}{b}$ in the above expression. The reason for this will become apparent after we introduced the dimensionless variables.

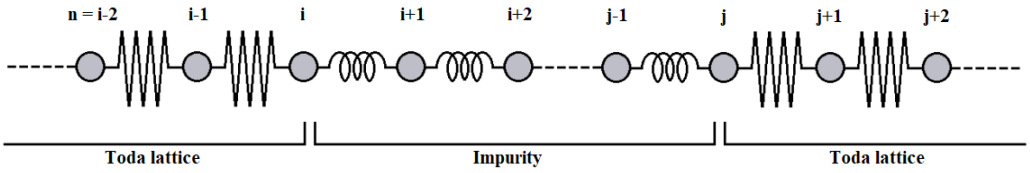


Figure 3.8: *The experimental set-up. Between the two Toda lattices is an impurity made from $N = j-i$ sites, each two sites are connected by the harmonic potential. The lattice can be imaged as a series of sites being connected by springs. The sites in the impurity would then be connected by conventional springs and the sites in the Toda lattice connected by springs following the Toda potential. Those Toda potential springs would behave quite different than the familiar conventional springs as we have discussed in the first section.*

To allow us to focus more on the characteristics of the dynamics and less on keeping track of the different constants of both potentials, we are changing the variables into dimensionless variables. Those variables are the same as the variables that were introduced by Kubota and Odagaki in [89–91]. One advantage of using dimensionless variables is the experiment model is indifferent to the units you are measuring the quantity in. The dimensionless variables are written as

$$P_n = \left(\frac{b}{ma}\right)^{1/2} p_n, \quad R_n = br_n, \quad K = \frac{\kappa}{ab}, \quad \tau = \left(\frac{ab}{m}\right)^{1/2} t, \quad \Phi(r_n) = \frac{a}{b}\phi(r_n). \tag{3.25}$$

The potential in dimensionless units is written as

$$\Phi_n(R_n) = \begin{cases} e^{-R_n} + R_n - 1 & \text{for } 0 \leq n < i \text{ and } n \geq j \\ \frac{KR_n^2}{2} & \text{for } i \leq n < j. \end{cases} \tag{3.26}$$

We already know the energy of a soliton is given by equation 3.15. To obtain the energy density function the Hamiltonian needs to be rewritten into dimensionless variables, we then divide it by the energy of the soliton to obtain an expression that is normalized. Between the different simulations we will be using the normalized energy because it allows us to assign percentages, how much energy of the initial soliton is being transported to the reflected or transmitted solitons.

$$h_n(\tau) = a\Big(P_n^2(\tau) + \Phi_n[R_n(\tau)] + \Phi_{n-1}[R_{n-1}(\tau)]\Big)/2bE_0, \tag{3.27a}$$

$$\text{where } \sum_n h_n(\tau) = 1. \tag{3.27b}$$

In the above function $\sum_n \phi_n(r_n)$ has been replace by $\Big(\Phi_n[R_n(\tau)] + \Phi_{n-1}[R_{n-1}(\tau)]\Big)/2$, because we want to know the energy value of each individual site for the function h_n. Knowing the energy of the individual sites rather than the sum of the entire system allows us to draw the plots such as those in figure 3.7. Every site has two neighbouring sites, except the sites at the boundary, and the sites interact with their neighbours through the potentials connecting them. The term $\Phi_n[R_n(\tau)]$ describes the potential energy between sites n and $n+1$, and $\Phi_{n-1}[R_{n-1}(\tau)]$ describes the energy between sites $n-1$ and n. The potential term needs to be multiplied by $\frac{1}{2}$ so that we do not count their contribution double in expression 3.27b.

Chapter 4

Controllable delay and logic gates

In the previous chapter, the theory of the Toda lattice was introduced, in this chapter, we are going over the method that was used to simulate the time evolution of the lattice, how the logic gates were designed and the mechanic that makes the design possible, the controllable delay in the soliton travels. We measure how the delay is influenced by the wavenumber of the soliton, the number of sites in the impurity and by the incident time. This chapter will be finished by demonstrating the operations of the designed logic gates, those being the XOR and OR gate.

4.1 Simulation

4.1.1 Integration method

We can initialize a soliton in the Toda lattice, it is now time to determine the solitons new position a moment in time later and how this can be done inside the simulation. Equation 3.16b describes only the dynamics of the solitons, but not how the sites in the lattice needs to change and equation 3.20 does not have any time dependencies. Verlet is an integration method which uses the current site position x_i and the position of a moment earlier in time x_{i-1}, to determine the future position x_{i+1}. This integration method is one of the simplest integration methods available and another simple method is called Euler integration. During some of the initial tests studying both integration methods, the Verlet integration method was closer to the predicted values for the Toda lattice than those created with the Euler integration, therefore we chose the Verlet integration. In a later section, we will be discussing the fluctuation in the total amount of energy seen in the simulation. The total amount of energy is conserved, so when we sum over all the energy in the system, the number that we get should be constant. But small deviations from this number has been seen and this small fluctuation is caused by using this particular integration method, other integration techniques will display a different amount of energy fluctuation. If we would divide the moments in time that we simulate to infinitely small, then we would not observe any fluctuations, but the fluctuations were small enough, as will be discussed in a later section, that we can demonstrate the logic gate designs with confidence. The new position of a site given by the Verlet method is

the following

$$x_{i+1} = 2x_i - x_{i-1} - \alpha(\delta t)^2, \tag{4.1}$$

where δt is the step in time we are making, the smaller this step is, the more accurate the simulation will be and α is the acceleration. We have chosen to use α as the acceleration symbol to differentiate this quantity from the Toda potential constant a. The acceleration can be found from Newton's law by rewriting $f = m\alpha$. For the Verlet method, we need to know the previous site positions when we initialize solitons in the simulation for the first time. This knowledge is important because it will also determine the solitons direction over time, if it is a right or left moving soliton. We can find an expression by subtracting the step in time δt from the left-hand side of 3.17d and then rewriting the expression again for t_c

$$t_c = \mp\Big(\frac{kn_c}{\omega} \pm \delta t\Big). \tag{4.2}$$

Combining the above expression with 3.16b and 3.19, we obtain the following expression,

$$x_{n+1,i-1} = \frac{-1}{b}\,\ln\Big(\frac{m\omega^2}{ab}\mathrm{sech}^2\big(k(n-n_c) + \omega\delta t\big) + 1\Big) + x_{n,i-1} + L, \tag{4.3}$$

were the index i is used for the steps in time. The results created by applying this integration method have been compared with the results of Kubota and Odagaki [90] by reconstructing figure 5 of their work. Our simulation found the same values as Kubota and Odagaki did for the delay period in the soltions travels. The length of the lattice was sufficiently long enough in the simulation that the boundary condition did not influence the final results. If we opted for a fixed boundary condition, the length of the lattice would be shorter, but this condition will cause the solitons to bounce back from the end of the lattice. This is analogous to a ball hitting a solid wall. We are almost done with the preparations needed for the Verlet integration, we now only need to define the momentum for equation 3.27b, the normalized energy $h_n(\tau)$. The dimensionless momentum is given by

$$P_n = m\frac{R_n(\tau) - R_n(\tau - \delta\tau)}{\delta\tau}. \tag{4.4}$$

Because the momentum will be squared in $h(\tau)$, the sign of the momentum is not important to be defined here.

4.1.2 Building a controllable delay

We have now prepared everything necessary for the simulation, we are in a position to measure the controllable delay. Up to this point, everything was based on the works of Toda or Kubota and Odagaki, but there has not been an investigation concerning what happens when one soliton is trapped in the impurity and another soliton arrives at the impurity. This investigation will lead us to a delay with a certain degree of controllability, which provides us with the mechanic to create additional logic gates in the lattice. The controllable delay is created by initializing two solitons with the same wavenumber, which

are initialized Δn_I sites apart from each other. We initialize the right moving solitons left of the impurity and because both solitons have the same wavenumber, they will arrive at impurity at two different times. This difference we will be calling the *incident time* $\Delta\tau_I$. The incident time is short enough that one soliton is still trapped in the impurity so that a second soliton is able to scatter against the impurity. This influences the energy stored in the trap in such a way that it prolongs the trapping period of the transmission soliton significantly. Figure 4.1 shows the set-up for the controllable delay.

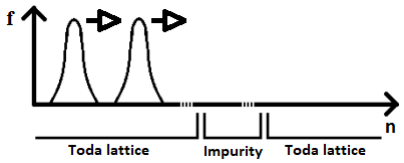


Figure 4.1: *The controllable delay set-up. We initialize two solitons in one Toda lattice in such a way that they will move towards the impurity. The image indicates the force magnitude exerted on every site in the lattice, showing the locations of both solitons.*

We will initialize one soliton always at position n_0 and the second soliton will be initialized at position $n_0 - \Delta n_I$. In figure 4.2 we can see the difference in outcome when we only change the incident time. An abstract depiction of the solitons travels is drawn in figure 4.3. Figure 4.3(a) depicts the situation of figure 4.2(a) and 4.3(b) depicts 4.2(b). The function $h_n(\tau)$, the height of the graph has not been drawn in this depiction.

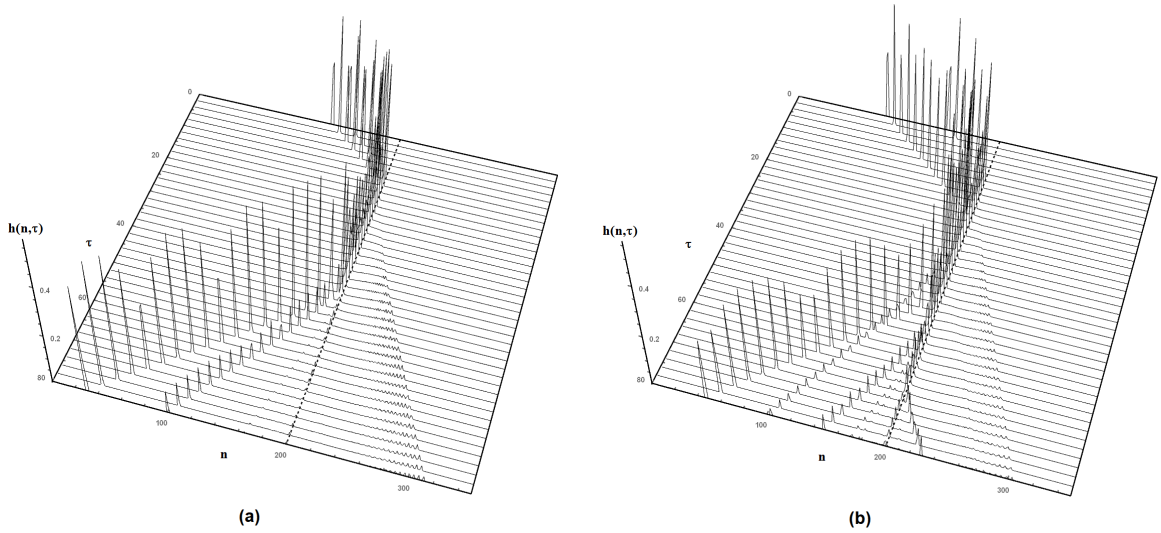
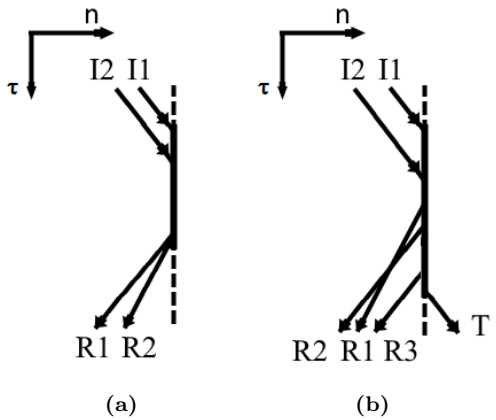


Figure 4.2: *Two solitons are scattered by the presence of the impurity. The incident soltions has a wavenumber $k_0 = 3.5$ and there where $N = 2$ sites connected by the harmonic potential. The thick - - - - dashed line indicates the position of the impurity. (a) The incident time is $\Delta\tau_I = 3.39$, which cause both solitons to be reflected after a delay. (b) When we increase the incident time to $\Delta\tau_I = 12.69$ we still have two solitons that are being reflected, but there is also a transmission. A third reflection can also be seen.*



(a) (b)

Figure 4.3: *An abstract depiction of two solitons scattering against the impurity. The - - - - dashed line shows the location of the impurity, I1 is the first incident soliton, I2 the second. The reflections are R1, R2 and R3 and T the transmission. The horizontal axes shows the sites n in the lattice and vertically axes is the dimensionless 'time' τ. (a) this image depicts the dynamics of figure 4.2(a) and (b) depicts figure 4.2(b).*

The incident time is given by

$$\Delta\tau_I = \Big(\frac{\Delta n_I}{v}\Big)\sqrt{\frac{ab}{m}} = \Delta n_I \frac{k_0}{\sinh\, k_0}. \tag{4.5}$$

$\frac{\Delta n_I}{v}$ gives a quantity in units of seconds, to make the expression dimensionless we need to multiply this quantity by $\sqrt{\frac{ab}{m}}$, because this quantity has the units of frequency. From now on we will be calling the initialized solitons *incident solitons*, to distinguish those solitons from the transmitted and reflected solitons. Throughout the simulations for measuring the delay caused by the controllable delay set-up, the mass will be $m = 1$, constants of the Toda potential will be $a = b = 1$ and for the harmonic potential the constant will be $K = 0.03$. We have chosen to use the same values a, b and m as those used by Kubota and Odagaki [89–91], this makes comparing the results from this chapter with the other authors easier. The value of κ has been chosen as such specific value, because Kubota and Odagaki have found that with a wavenumber of k_0, this harmonic potential constant gives the biggest possible delay [90]. Also because this value represents a very weak connection between two sites which are connected by the harmonic potential, compared to the connection created from the Toda potential. Such a weak connection is needed when trapping a soliton, as determined by Kubota and Odagaki [90]. The code for the controllable delay is described in appendix A, together with the complete program for the delay written in Wolfram Mathematica programming language.

4.2 Measuring the controllable delay

4.2.1 Delay by scattering

We find in our simulation that when the incident time was small, there were only reflections, but when we increased the incident time between the two solitons, transmission(s) were observed. Some incident time values have only one or more reflections, those are for

the wavenumbers $k_0 = 2.5$, 3 and 3.5). When the incident time is big enough, such as $k_0 = 3.5$ and 4, a third reflected soliton can be observed and for $k_0 = 4$ a second transmission. In figure 4.4 we show the delay time in dimensionless units for different incident times $\Delta\tau_I$. The time of delay for the reflections and transmissions are measured as the difference between the moment when the first incident soliton arrives at the impurity and the moment when a soliton leaves the impurity, either as a reflected or transmitted soliton. By changing the incident time, we change the time the solitons leaves the impurity. The delay of the transmissions in this set-up was larger than was measured by Kubota and Odagaki [90], who found the delay to be $\tau_{trap} \simeq 32.5$ for $k_0 = 3.5$, $K = 0.03$, $N = 2$ and one soliton.

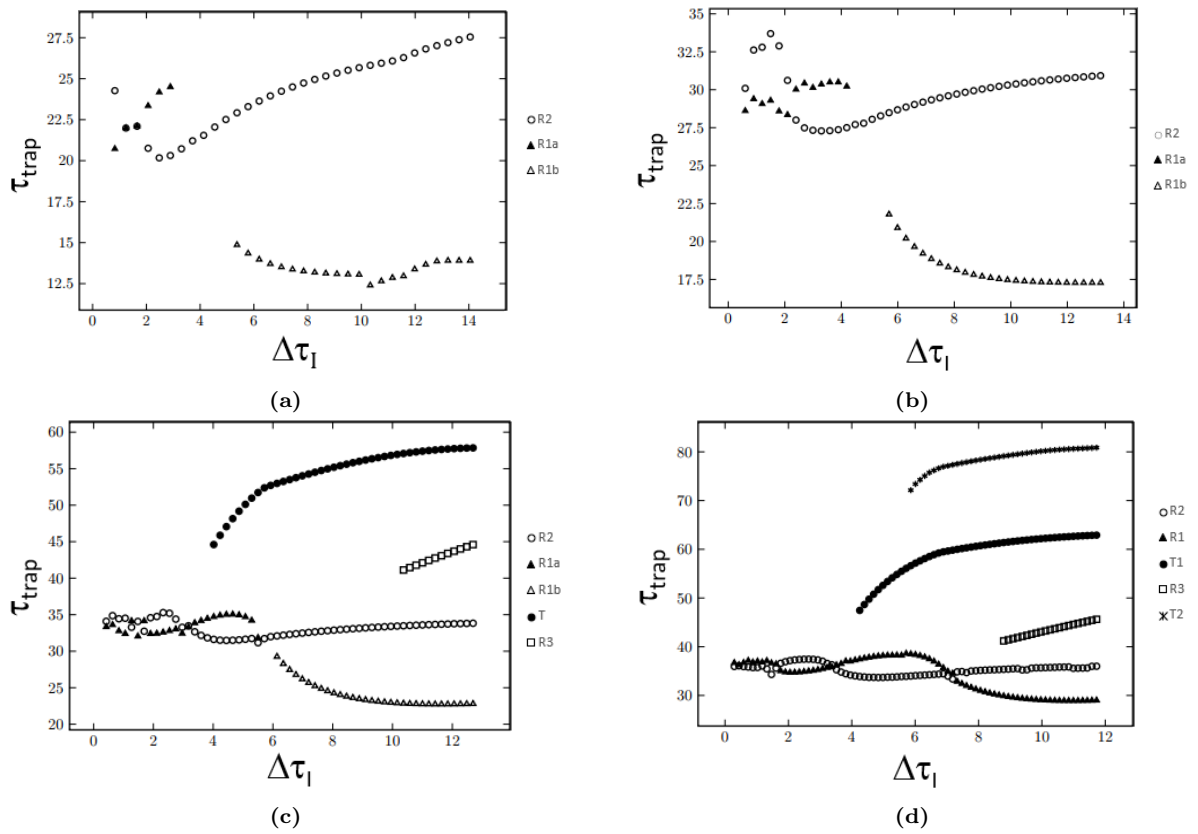


Figure 4.4: *The time delay when solitons are reflected and transmitted. The time difference between the first incident soliton arriving at the impurity and a soliton being reflected or transmitted from the impurity, is plotted against the incident time. This is shown for four different wavenumbers, namely (a)* $k_0 = 2.5$*, (b)* 3*, (c)* 3.5 *and (d)* 4*. The reflections are indicated by the letter R in the legend and the transmissions by the letter T. For this graph the number of sites connected by the harmonic potential was* $N = 2$ *with a harmonic potential constant of* $K = 0.03$ *for all four wavenumbers.*

While gathering data for figure 4.4, it was observed for $k_0 = 2.5$, 3 and 3.5, that the reflected soliton $R1a$, becomes so small in energy and had moved to the same position as the ripples, that it can not be distinguished from the ripples any longer. When we increased $\Delta\tau_I$ we observed a second reflected soliton, $R1b$, emerging from the impurity. The distinction between the reflections R_1a and R_2 can be ambiguous at times, because there are values of $\Delta\tau_I$ for which they leave the impurity at almost the same time. To determine which of the two reflections was R_1a and R_2, we followed the general trend of

the time they had left the impurity, how much energy the reflections had and which of the reflections increased or decreased in energy over time.

4.2.2 Measuring delay

The relative displacement R_n is shown for the sites $n = i$, $i + 1$ and $n = j = i + 2$ in figure 4.5, together with the moment in time when the incident soliton is trapped in the impurity and the moment the soliton leaves. We are looking at three sites connected by the harmonic potential, thus we are looking at $N = 2$ sites. The most accurate way to measure the delay time of both the reflections and transitions is to measure when the relative displacement of R_i, R_{i+1} and $R_{j=i+2}$ restore themselves to their equilibrium positions, after they have moved a significant amount. This comes from the characteristics of the Toda lattice, the potential that connects sites behaves as if it were reluctant to be compressed. The same method was also applied in [90]. We have compared the time delay by another method, in which we measured the change in force three sites away (on both sides) from the impurity. If the force changes significantly (in all cases above 5 N) and it goes back down again shortly after, we then know that a soliton has passed that point. The described method gave close results with the method of following the relative displacement, but it gave a longer delay because it was measured three sites away from the impurity. Our criteria that the emerging signal cannot be ripples, but has to be a single soliton, implied that we did not observe delays in the transmission for $k_0 = 2.5$ and 3, even when the relative displacement R_3 changed significantly. In this we deviate from the results of Kubota and Odagaki where they did record delays in transmission for those wavenumbers.

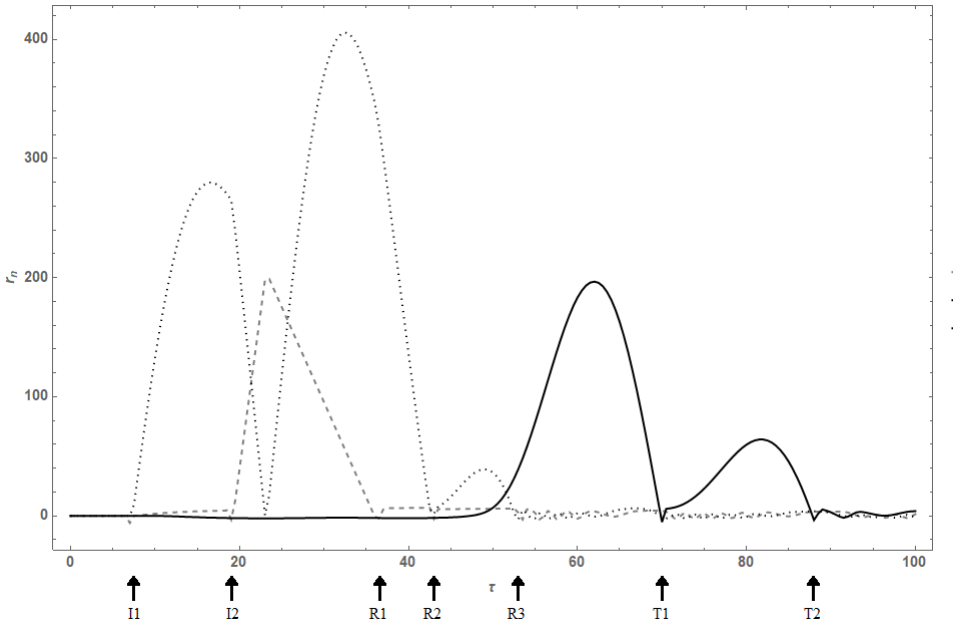


Figure 4.5: *The sites displacement against time. In the graph the moment that the incident solitons $I1$ and $I2$ hits the impurity are indicated, the time the reflections leave $R1$, $R2$ and $R3$, and the transitions $T1$ and $T2$ escaping the impurity. Relative displacement R_i is shown by the - - - - grey dashed line, R_{i+1} by the ····· grey dotted line and $R_{j=i+2}$ by the —— black solid line. This images was made with the values $k_0 = 4$, $\Delta\tau_I = 11.73$, $K = 0.03$ and $N = 2$.*

4.2.3 Energy from scattering

In our results, we only count the outgoing waves as reflections or transmissions when the energy of the wave was above 0.1% of the normalized energy density, and only when the majority of the energy of those waves are located at one or two sites. The last requirement allows us to distinguish signals usable for computation from the ripples. The goal of this controllable delay is to eventually build logic gates, that will eventually lead to create circuits from those gates. Therefore this requires that signals coming from one gate as output can be used as input for the next gate, that gates can be cascaded with each other. Thus the energy of the transmitted soliton is very important and this is dependent on the wavenumber, as shown by equation 3.15 and the incident time. From the work of Kubota and Odagaki [90] we know that it is also dependent on the harmonic potential constant. The undesired ripples can be seen in figure 4.2(a), they are moving to the right after the incident soliton hit the impurity. In 4.2(b) can be seen that the different reflected solitons have a different energy value from each other and they, therefore, move at different speeds through the lattice. Comparing 4.2(a) with 4.2(b), we see that the reflected solitons escapes the impurity at different moment. In figure 4.6 we show the normalized energies of the emerging solitons.

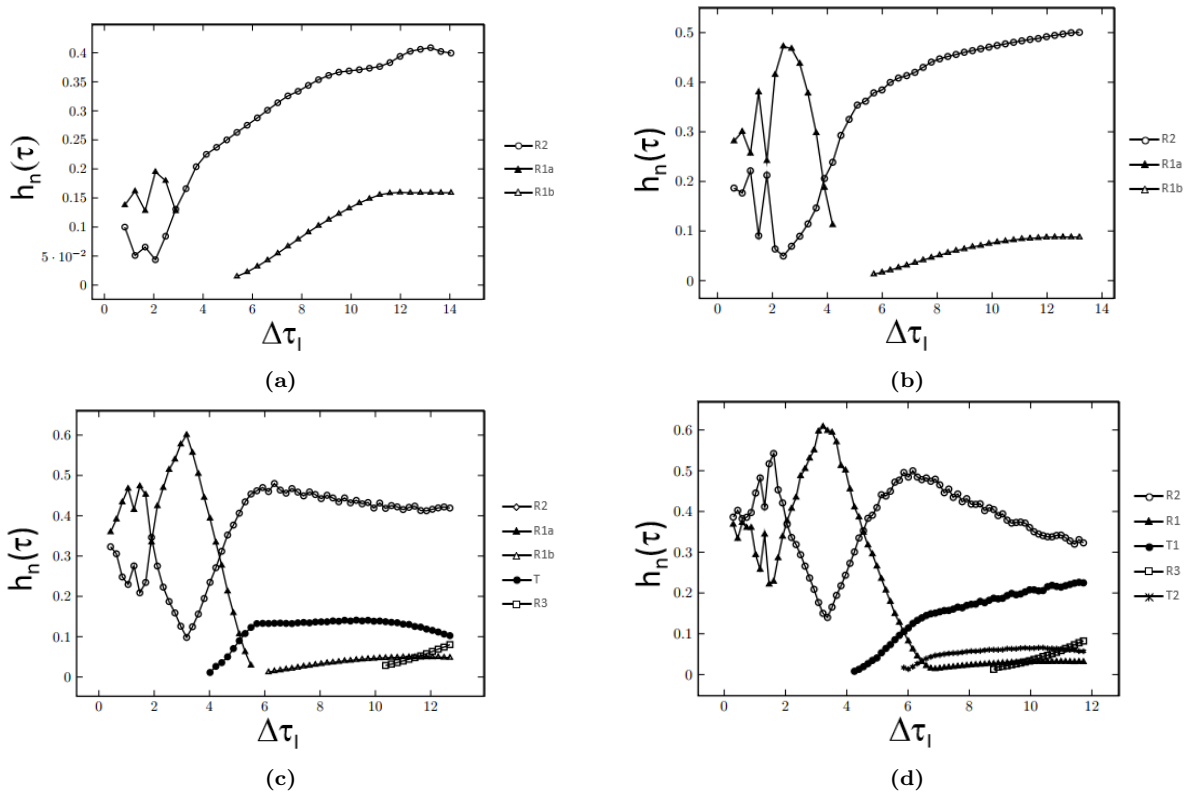


Figure 4.6: *The energy of the reflected and transmitted solitons plotted against the incident time. This has been done for the four different wavenumbers (a) $k_0 = 2.5$, (b) 3, (c) 3.5 and (d) 4. The reflections are indicated by the letter R in the legend and the transmissions by the letter T. The number of sites connected by the harmonic potential was $N = 2$ with a potential constant of $K = 0.03$ for all four wavenumbers.*

The steps in time δt was 5 ms. Because of the size of the time step, the error in the total amount of the normalized energy was smaller than 2% when the solitons were initialized, by making the steps in time even smaller will decrease this error. Shortly after initializing the incident solitons, this error became smaller than 0.1%, the energy can practically be seen as conserved for the rest of the simulation, about a second after the initialization.

4.2.4 Increasing size of the impurity

When we increased the number of sites in the impurity in the Toda lattice, we observe that the delay of the first soliton being transmitted is increased. This is shown in figure 4.7. For figure 4.7(b) we chose one particular incident time which gave only one reflection, for (c) an incident time which for the wavenumbers 3.5 and 4 gave three reflections. We changed the size of the impurity for two different incident times to investigate how this difference in incident time affects the transmitted soliton energy. On the time scale, this difference in incident time does not affect delay time by much, the biggest contributor is the number of sites in the impurity, as can be seen in 4.7(a). What we can take away from this figure is that the energy of the transmission is dependent on the size of the impurity, the wavenumber, harmonic potential constant and to some extent on the incident time. An important observation is that there is a transmission for $k_0 = 3$ when there are $N = 4$ sites connected by the harmonic potential, this cannot be observed for $N = 2$ or $N = 3$. Another observation was the incident time within the regime where there was only one reflection, $4.19 \leq \Delta\tau_I < 5.39$, when we increased the N to ≥ 8 for $k_0 = 3.5$, we observe a second reflection. If we increase the number of harmonic potentials in the impurity to $N = 10$ for both the incident times for $k_0 = 3.5$, a second transmission can also be observed. The wavenumber $k_0 = 2.5$ does not show any transmissions when the sites in the impurity was increased from two to ten. For the wavenumbers $3, 3.5, 4$ the energy of the transmitted soliton changes for different sizes of the impurity. The biggest transmitted soliton energy with a wavenumber of 3.0 and with $\Delta\tau_I = 11.98$ was with $N = 4$ sites in the impurity, while the same wavenumber but with an incident time of $\Delta\tau_I = 4.79$ has its the biggest transmission when there are $N = 3$ sites.

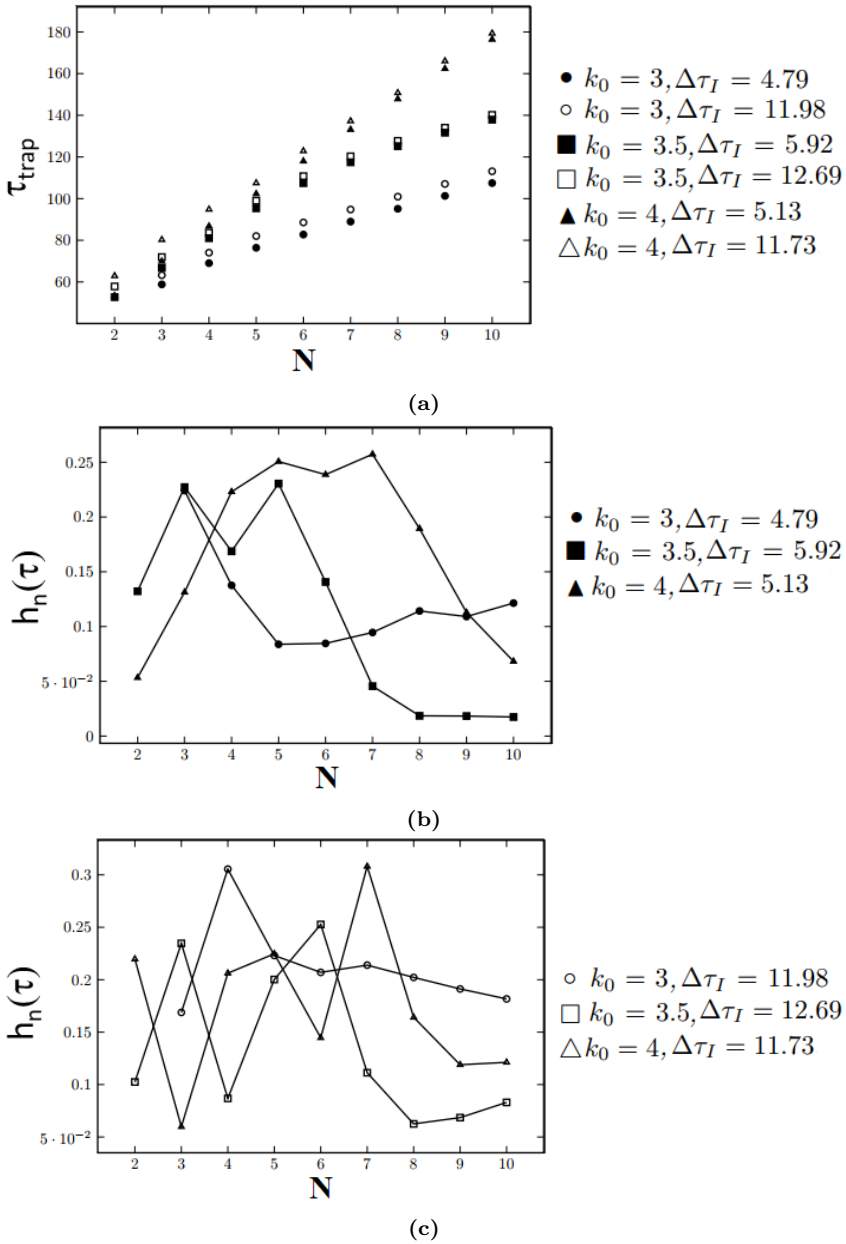


Figure 4.7: *Delay and energy for the first soliton being transmitted. (a) shows the delay of the transmitted soliton against different number of sites N in the impurity, for different wavenumbers and incident delay. (b) shows the energy of the transmitted soliton against different N values within the regime were $k_0 = 2.5, 3, 3.5$ only had one reflected soliton. (c) shows the energy of the transmitted soliton for an incident time much longer than was shown in (b). The wavenumber $k_0 = 2.5$ does not show any transmissions when the number of harmonic potential are increased.*

4.3 Demonstrating the logic gates operation

4.3.1 Constructing soliton logic gates

In section 4.2.1 we have seen that for only the wavenumbers $k_0 = 3.5$ and 4 there was a transmission observed. To construct a logic gate in the lattice using the controllable delay will require the incident solitons to have a wavenumber of $k_0 \geq 3.5$. To demonstrate the

functionality of the logic gate design presented in this section, we will be using incident solitons with a wavenumber of $k_0 = 3.5$, because for this value there was one transmission observed, rather than the two transmissions for $k_0 = 4$. For a soliton with a wavenumber of $k_0 = 3.5$, Kubota and Odagaki [90] found that an impurity with a harmonic potential constant of 0.03 N/m, gave the longest delay possible. Therefore this is the constant value we will be using in our logic gate demonstration.

In 2013 Kubota and Odagaki [91] were able to design three different logic gates in the Toda lattice, they showed a design for NOT, OR and AND gates. With the controllable delay mechanic shown in the previous section, it is possible to design an XOR and a differently designed OR gates. The logic value 0 is represented by the absence of a soliton and the presence represents the value 1, both for in- and output, which is the same representation as was used by Kubota and Odagaki. For the logic gates design, we have two incoming lattices that will function as our inputs and they are connected to one lattice, which functions as our output. The set-up is shown in figure 4.8. One input is longer than the other, shown in the figure in the upper line as a "valley". This allows us to initialize two solitons at the same site n in both lattices. Because one lattice is longer than the other, this causes both solitons to arrive at the impurity at two different times, we therefore are in the same situation as we were when measuring the controllable delay. The length of the valley can be found using equation 4.5, together with the results from the previous section. The input and output lattices all have the same values for the potential constants a, b and K.

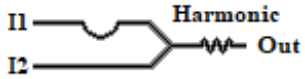


Figure 4.8: *Logic gate design in the Toda lattice. This gate uses the absence of a soliton to represent the value* 0 *and the presence as the value* 1*. The upper input lattice I1 is longer than the bottom, the extra sites are shown as a "valley" in the figure.*

The program that simulates the logic gate is also described in appendix A, together with the complete program for the delay written in Wolfram Mathematica programming language.

4.3.2 Performing XOR and OR operations

As shown in section 4.2.1, for certain incident time values there are only reflections. To build the XOR gate we need to be within that incident time. For $k_0 = 3.5$ that would be $\Delta\tau_I$ <3.81, thus using equation 4.5 the extra length for input $I1$ needs to be less than 19 sites. With the wavenumber $k_0 = 4$ the incident time needs to be $\Delta\tau_I$ <4.25, which is a length smaller than 29 sites. In figure 4.9a we show the working of this XOR for the input state 01, which gives an output of 1. As we can see in this figure, there is only one soliton initialized, it does not matter in which input this is initialized. An abstract depiction of the gates' operation with the input state 01 is drawn in figure 4.10(a). When we initialize two solitons in both lattice, we get an output of 0, as shown in figure 4.9b. As can be seen in the figure it takes one soliton longer to arrive at the impurity than the other. This input state has been drawn abstractly in figure 4.10(b). In this set-up,

the incoming solitons are reflected back to both input lattices. In figure 4.9b when the soliton is reflected from the impurity, the reflected soliton arrives at the point where the three lattices are connected, a portion is then reflected back towards the impurity again. Because the potential constant on every Toda lattice is the same, the reflected soliton will experience 'a lattice' (the two input lattices together) that has a constant of $2a$, this will make it more difficult for the solitons to crossover to the input lattices, causing some of the energy to be reflected back to the impurity. We have observed the same kind of reflections when we had a one dimensional lattice and the second half of this lattice had a constant value of $2a$. This is in line with results from Nakamura's study what the effects it has on of a solitons travels through the lattice if some sites have different mass or constant values of the Toda potential [106]. In the simulation, the new reflected soliton heading for the impurity, but were too small to become a transmission soliton, later on, the energy of this soliton was therefore dissipated by the impurity in the form of the ripples.

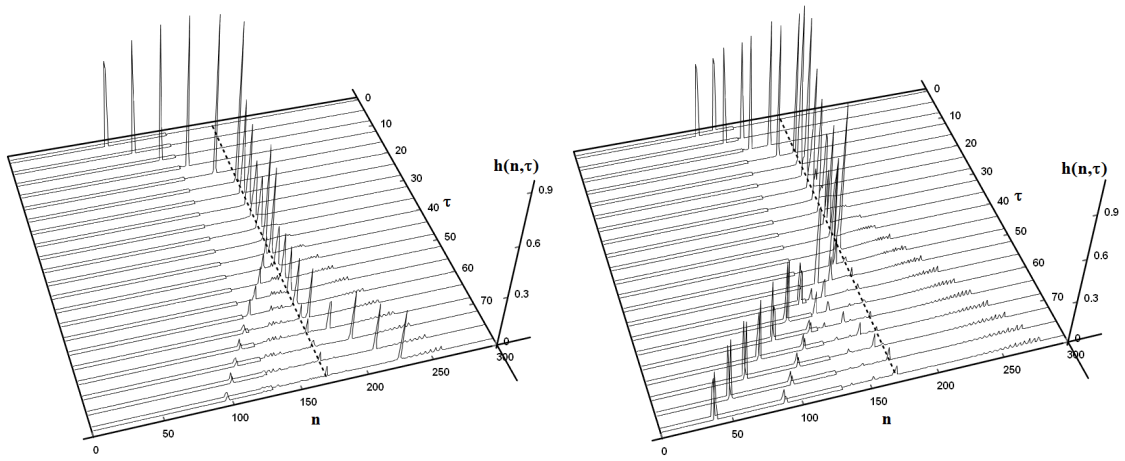


Figure 4.9: *XOR gate operation being performed in the Toda lattice. In (a) input state is* 01 *and output is* 1*, (b) input state is* 11 *and output is* 0*. The thick - - - - dashed line indicates the location of the impurity. The number of sites in the impurity were* $N = 2$*, wavenumber* $k_0 = 3.5$ *and one input was 15 sites longer than the other.*

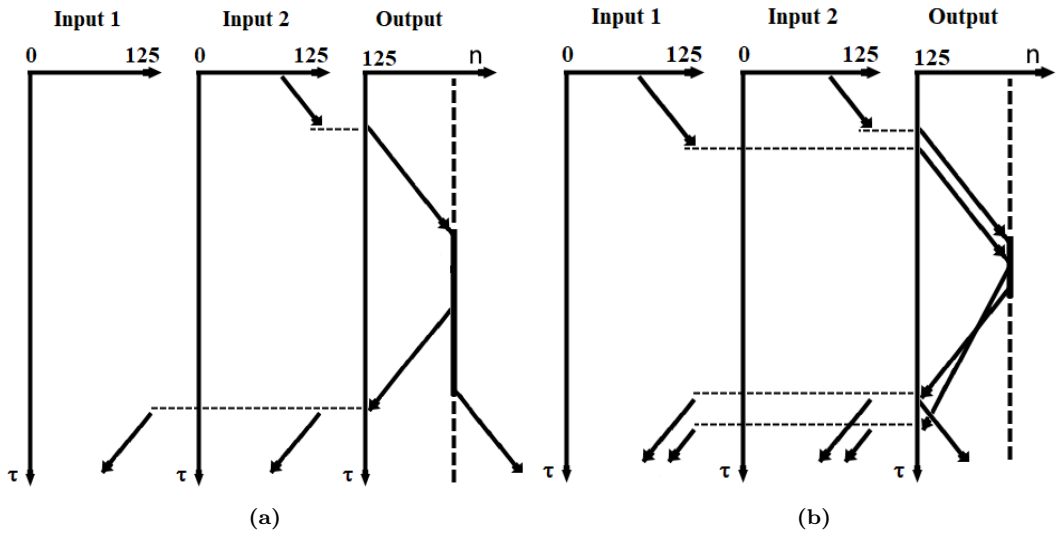


Figure 4.10: *An abstract depiction of the XOR gate operations in the Toda lattice. In (a) is depicted that the solitons travelling from figure 4.9(a) and (b) depicts figure 4.9(b). When a soliton leaves on of the inputs (the lattice functioning as the logic gate inputs), it enters the output lattice. The reflection solitons leaving the impurity will enter both input lattices. Their energy is divided to both lattices, this means that the speed of the soliton is lower, but in this depiction this has not been incorporated to keep the depiction more straightforward.*

If we increase the length of the input lattice even more, we can have transmissions for the input state 11, as was seen from our controllable delay investigation. The energy of transmitted soliton will depend on the increased length in the upper lattice, as length corresponds to the incident time. We still have transmissions for the 01 and 10 input state, just as before. To construct an OR gate we give the input $I1$ 19 or more sites than $I2$ for $k_0 = 3.5$, we would need more than 29 sites for $k = 4$. The gate operation for the 11 input state is shown in figure 4.11. This design is an alternative to the OR gate design from [91]. Again in this design, a reflection can be seen when the reflected soliton tries to enter both input lattices.

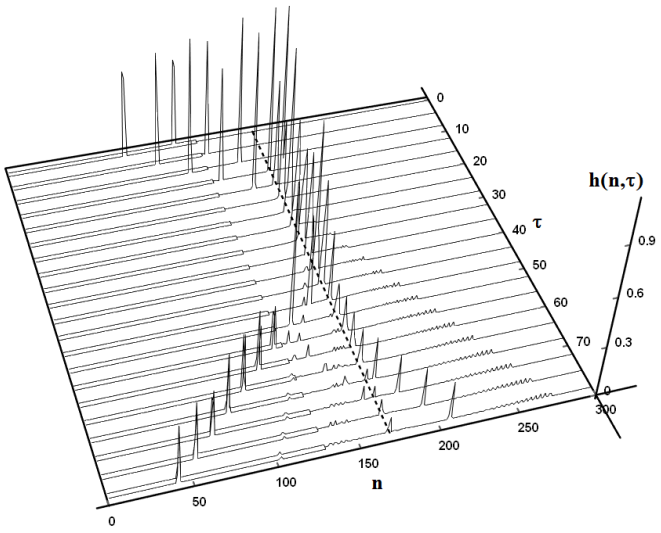


Figure 4.11: *OR gate operation being performed in the Toda lattice. Two solitons are initialized corresponding to the input state* 11 *and which gives an output of* 1*. The thick - - - - dashed line indicates the location of the impurity. The number of sites in the impurity were* $N = 2$*, wavenumber* $k_0 = 3.5$ *and one input was 35 sites longer than the other.*

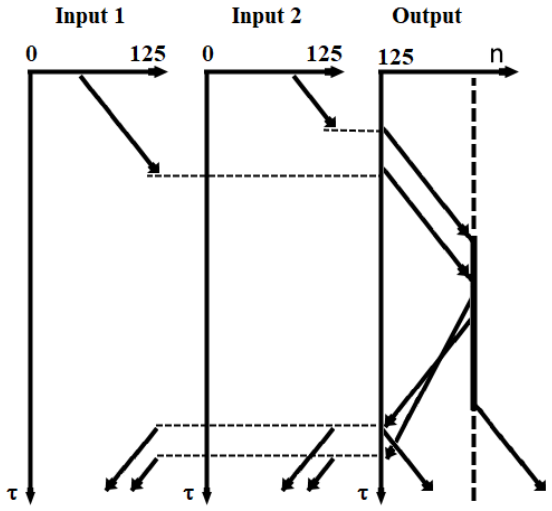


Figure 4.12: *An abstract depiction of the OR gate operations in the Toda lattice. It depicts the state 11 of the OR gate as shown in figure 4.11.*

In section 2.1 we have gone over some of the points discussion in Horsman *et al.* paper [69]. It is now time to go over those points and reflect if the above described logic gates could qualify for a physical system capable of performing computations. The information has been encoded in representing the value 1 and 0 as the presence and absence of a soliton in the lattice respectively. The same representation is also used for decoding the information back, with the extra requirement that the presence of a soliton is only recognized if the energy of this soliton is $> 0.1\%$ of the incident soliton. The abstract model m_p of the logic gate designs has been established in chapter 3 and in figure 4.8. Our Mathematica program simulates a physical system in the initial state p and the program then evolves this system to the state p'. What we have done with the simulation is evolving the state m_p of the model into the state m'_p of the physical system. The simulation is based on the theories of the Toda lattice, therefore the simulation plays the role of $\mathcal{C}_\mathcal{T}$ in the framework. There exist relations $\widetilde{\mathcal{R}}_\mathcal{T}$ between the abstract Toda lattice model and a few physical systems, as was discussed in 3. But direct relations $\widetilde{\mathcal{R}}_\mathcal{T}$ and $\mathcal{R}_\mathcal{T}$ between the logic gate design in the lattice and any physical systems has not been established as of this moment. We can not therefore complete the diagram of the framework. One implementation of the Toda lattice is the LC ladder network [92, 124]. To build our logic gate design in this implementation is in theory possible, we would be able to measure both input incident solitons and the output soliton by the voltage across specific capacitors in the network. The shape of our incident soliton would be visible on an oscilloscope. The precise values of the coils, capacitors and resistors, and the shape of the network required to realize any kind of Toda lattice logic gate designs is an open question. Knowing those values would be knowing the relations $\widetilde{\mathcal{R}}_\mathcal{T}$ and $\mathcal{R}_\mathcal{T}$. But with such a physical realization, we would be completing the framework diagram from figure 2.4 and be allowed to categorize this physical implementation and the gate designs from thesis together as a physical system that is performing computations.

Chapter 5

Conclusion and outlook

This chapter summarizes the outcome of the thesis, the implications and providing an outlook on some continuations possible for the presented work.

5.1 Summary and conclusion

This thesis aimed to explore the computing capabilities of the Toda lattice. The soliton solutions of the Toda lattice or its variation have been shown to be equivalent to the nonlinear Schrödinger equations or a variation thereof shown by the work of Arnold [11] and Kaup [83]. Theoretically, it is possible to translate the results presented in this thesis to soliton solutions of a system governed by the nonlinear Schrödinger equations. In this thesis, the number of possible logic gates in the Toda lattice was extended to include an XOR gate design and an alternative design for the OR gate. The gates presented in this work are capable of being cascaded with the OR and AND gate presented by Kubota and Odagaki [91]. The mechanism for the additional logic gates was the controllable delay and this delay temporarily traps the soliton who are travelling through the Toda lattice. The trap, which was a lattice made from sites that were connected by the harmonic oscillator potential, which acted as an impurity inside the Toda lattice. A degree of controllability in the amount of delay was created by changing the incident time. This time parameter was the time difference between two solitons arriving at the impurity. To construct any logic gate, a representation of information needs to be defined. In this thesis, the binary value 0 was represented by the absence of a soliton and the value 1 by the presence of a soliton carrying an energy that was $\geq 0.1\%$ of the total initialized solitons. The difference between the design of the XOR and OR gate was the extra length in one of the lattices, which causes the initialized solitons to have a different incident time, creating different behaviour in the way solitons are transmitted from the impurity.

The contributions of this thesis has been the first investigation of soliton scattering against an impurity while another soliton was trapped inside this impurity, as discussed above. The results of the investigation has been incorporated in the next contribution, a new design for a XOR and an OR logic gate inside the Toda lattice. In appendix A a code programmed in Mathematica has been added, which is the third contribution which has been made to the field of soliton computing. The code creates a Toda lattice which has

been validated by comparing the results of other authors [90, 91]. After the validation stage, the simulation has been used in the investigation of soliton scattering against a trapped soliton and for testing the functionality of the new logic gate designs.

While the Toda lattice is used in different fields, ranging from solid matter physics [17] to modelling DNA [105], the lattice has not been researched much in its computing capabilities, such as those presented in this work. While cascading can be done in the lattice, the logic gate design presented in this work and in those from Kubota and Odagaki, still requires an amplifier when building more complex circuits. After every logical operation around $10 - 30\%$ of energy remains of the original input left for the output soliton to be used. Therefore energy losses after every logic gate operation will require amplifiers to build a physical circuit of cascaded gates, when we are using the mentioned designs in this thesis. Using a different representation for the soliton boolean values or a different set-up, might mean that amplifier is not required when building complex circuitry, such as the representation used below in section 5.2.1. The Toda lattice is among the mediums that have soliton solutions and are capable of performing logical operations. Which soliton system would be the best suited for carrying information in a fully functional optical computer, that question at present can not be answered with certainty. In section 5.2.2 we go over one suggestion that could help with answering this question and how the Toda lattice may play a part in answering. This thesis has focused on using the Toda lattice in the context of optical computing, but because the lattice can also be used to model switches in DNA [64, 86] using solitons as information carriers, it is not a stretch to incorporate the results into a set-up with DNA to construct logic gates analogue to the design presented here. Which paradigm will be used in future technology is never certain and therefore which path researchers need to take to get us from one paradigm to another is also uncertain, we may only hope that our contribution will be fruitful in the future.

5.2 Suggestions for future research

In this section, I will describe some possibilities for continuing the results found in this thesis. As has been pointed out in the introduction in chapter 3, there are some variations of the Toda lattice equivalent to the nonlinear Schrödinger equations (NSE). Thus the first suggestion for further research is to translate the results from this thesis to the NSE, enabling one to construct logic gates that were done in this work in the NSE. There are different approaches demonstrated in the NSE to be functional logic gates [7, 8, 13, 14, 16], which of those approaches are the most optimal for a given situation is still an open question.

5.2.1 Non Boolean soliton logic gates

The literature on computing with solitons is mainly focussed on binary logic. We may wonder if not other forms of logic are possible. In appendix B we give a small introduction into multi-valued and fuzzy logic. With the success of Boolean electronics in the past century, the tendency to apply the familiar Boolean logic for other physical systems is

understandable. Perhaps much can be gained using a different type of logic when applying solitons for computing. We may wonder if the reason for the popularity of Boolean logic in the soliton computing field is due to the properties of soliton or merely by the success of Boolean logic in electronics. If it is the latter, then we may ask the following question *what is the most optimal logic for a given medium with soliton solutions for constructing logic operations*. Optimal could mean in a given context the most energy efficient, the fastest, the smallest in a certain space, the most secure or perhaps robust in carrying information. Some information representations used in computing with solitons do not lend itself for representations in other logic systems, for example, the presence and absence of a soliton is limited to Boolean logic. We will now go through one example in the Toda lattice which could allow for the construction of multi-valued logic gates.

The multi-valued logic gate option uses the presence of a soliton within a certain space and time to represent the values for the output. Let us call this representation *clock window*, because it depends on the time of an external clock which value is associated with the output value. In the treatment of this representation, we will be using the Toda lattice as an example of how such a representation can be used to construct logic gates. We start the example of with Boolean logic for simplicity, but it is possible to extend it to another valued logic system and we show this with an extension to ternary. Suppose we have a lattice where a soliton travels from left to right and we have a detector that detects when a soliton is located within a certain region of the lattice. Figure 5.1 shows how such a set-up in the Toda lattice would look like.

Figure 5.1: *The detection of a soliton within a region of the lattice.*

One option is for the detector to start in the state 'unknown' at the beginning of the experiment and when it detects a soliton, it will change to the state A or B, depending when it is measured:

$$mT \leq t < (m+1)T - \epsilon T \begin{cases} \text{state will be state A if } m \text{ is even} \\ \text{state will be state B if } m \text{ is odd} \end{cases} \tag{5.1}$$

where the current time of the detection is t, m is an integer and the duration of the detecting certain states lasts for $T(1-\epsilon)$ seconds. T is the clock window and ϵ the extra buffer time required to still be able to distinguish between the different states. The detector will stay in this new state until it detects another soliton. We have named these states neutrally as A and B because they can represent boolean states either as (0,1) or as (1,0). The duration time of the detector needs to be chosen such that a slow soliton is not detected twice, neither that a fast soliton is never detected, therefore T has to depend on the speed of the soliton. A part of the soliton might be within the detection region and the other part is outside when the detector changes state, this would cause the detector to change states twice. To prevent this from happening we have introduced a small value ϵ and this value will depend on the size of the detection region and the width of the soliton. If one would use the same method of influencing the soliton as was done in

this thesis, then the ripples created from the interaction of the harmonic oscillator with a soliton would need to be taken into account when measuring to ensure that the detector does not change its state due to the ripples. Therefore we need to have a cutoff where waves below a certain amount of energy are not detected. The outlined representation allows us to expand and create more states. For example for the detection of three states A, B and C one possible division would be

$$mT/3 \leq t < (m+1)T/3 - \epsilon T/3 \begin{cases} \text{state will be A if } m \text{ is } 1, 4, 7, 10, \ldots \\ \text{state will be B if } m \text{ is } 2, 5, 8, 11, \ldots \\ \text{state will be C if } m \text{ is } 3, 6, 9, 12, \ldots \end{cases} \tag{5.2}$$

where the duration is now $T/3$. This way of defining the representation of logical values, could perhaps be the only possibility in the Toda lattice to define more states than binary. We have seen that it is possible to delay a soliton in the Toda lattice and this can be used to create a NOT gate with this clock window set-up, as shown in figure 5.2.

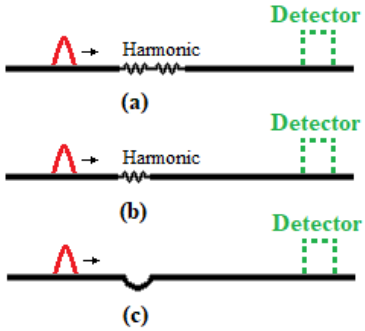


Figure 5.2: *Three possible ways to design a NOT gate with the clock window representation. (a) The series of harmonic oscillators delays the soliton in its travelling, (b) the harmonic oscillator is slightly off-resonance with the soliton, causing it to lower the energy of the soliton, which in turn causes the soliton becomes slower and (c) the soliton needs to travel a longer path, therefore delay the time when its arrives at the detector.*

In 5.2(a) the energy of the transmission soliton after being trapped in the impurity is lower than the initial soliton, this will also mean that the speed is reduced. Another way of realizing a NOT gate is shown in (b), one harmonic oscillator is placed in the Toda lattice and the harmonic potential constant is chosen in such a way that it is off-resonance just enough with the soliton's frequency, causing the soliton speed to reduce because it has lost some of its energy. A third design for a NOT gate shown in (c), this is to create a longer path for the soliton to travel on. In all three cases the soliton arriving at a later point of time than it normally would arrive at, therefore it will be seen as a different state. To construct an OR or NOR gate for this kind of representation in the Toda lattice we need the lattices to act as our inputs and output. The mechanism of the controllable delay can again be used in this set-up. The set-up is shown in figure 5.3 and the truth table for this gate is shown in table 5.1. When we do not have an impurity in the Toda lattice, we want the initialized solitons for example in the A state, to later combine the solitons at the branching site (the site were the three lattice come together) and arrive at the detector in the A state interval. The same holds true for solitons initialized in the B state, where solitons arrive at the detectors in the B state interval. We construct our

series of harmonic oscillators in such a way that solitons initialized in the A or B still arrive at the detectors A or B state. However, when we initialize one soliton in the A state and the other in the B state, we want the soliton transmitted from the impurity to arrive at the B state of the detector.

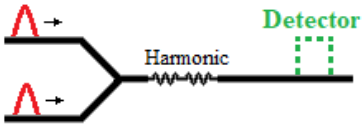


Figure 5.3: *OR or NOR gate set-up for the clock windows representation.*

I2	I1	Out
A	A	A
A	B	B
B	A	B
B	B	B

Table 5.1: *Truth table of an OR or NOR gate.*

When using this design for constructing the OR or NOR gate, we should keep in mind that for wavenumber $k_0 = 4$ a second soliton transmission was observed, therefore we need to keep the wavenumber of the initial soliton lower than $k_0 = 4$. So far only the construction of a NOT and, OR or NOR gate has been shown. In the thesis we have only experimented with two solitons of the same wavenumber, it is still unknown what difference it will make in the outcome when we have two solitons scattering against the impurity with different wavenumbers. An investigation into *the effect on the transmission delay when both initial solitons do not share the same wavenumber* could give us a hint in the components needed to constructed multi-valued logic gates using the discussed representation of this section. The option here presented only shows one possible approach of implementing fuzzy or multi-valued logic, as stated before, a non-Boolean logic option has not been extensively (if at all) been researched. Some of the benefits for non-Boolean logic are discussed in appendix B.

5.2.2 General soliton computing framework

In this work we have extended the number of logic gate possibilities in the Toda lattice and in chapter 2 we have seen different physical systems with soliton solutions who were capable of performing Boolean operations. But a question one might ask is the following, *which material additions does a system with soliton solutions need to contain if it is going to be used for (unconventional) computation.* With material additions I mean the extra material that needs to be implemented in the system to influence that system such that construction logic gates are possible. Phrasing the question differently, given an arbitrary physical system with soliton solutions, what kind of physical possible material properties other than those contained in the system itself, are required to incorporate before we can perform any kind of computation with this system. Those additions in the Toda lattice were the harmonic oscillator, without those harmonic oscillators there were only solitons

moving in the lattice and colliding with one another, never changing their initialized amplitude or velocity. There are systems and representations which makes it possible to perform operations without any additional material, such as the set-up shown by Wu, Chen and Chu [37], where they change the angle of the incoming solitons to change the operator being performed on the soliton. In the Toda lattice, we have used the absence of a soliton to represent the logic value 0 and the presence as the value 1. Only having solitons moving in the lattice would not be enough to perform logical operations in the Toda lattice with this logic value representation. Ideally for this question a framework would be developed that would apply to every physical system with soliton solutions and indicates if that system is capable of performing computations, which representation would be optimal, how operations are done and how information can be stored in this system for some time. The framework will ideally focus on the properties that all soliton systems have in common and rather than the details for every specific system. The framework would be an asset to researchers working in the soliton computing field and possibly increasing the amount of useful soliton systems for computing we know so far. Optical solitons have the most potential of being used for creating marketable optical computers, but other soliton systems can be used in other future technology for their robustness as information carriers. There are already switches made in DNA with solitons [64, 86] to name only one example with potential for implementation of solitons.

The framework does not necessarily have to specify what the "best" implementation would be, how to increase the speed, size, or even lower the amount of complexity of a soliton computing set-up. It may very well be that this framework would not be the most practical implementation, just as Boolean AND and OR gates are universal, but often it is not practical to make entire circuits from those gates alone. Ideally, the framework not only answers questions being asked by the researches, but also gives an indication of how to minimize the amount of extra materials required for performing computations for a given physical system with solitons. The framework could be broken down into three parts, encoding and decoding information, operating on that information and, storing and retrieving information, those parts are requirements for any kind of physical computer. The framework would then be able to focus on minimizing the amount of extra material properties required for each part. The quest for minimization has had a lot of interest and much has been written about it. In cellular automata (CA) for example, there is the search for the smallest Turing machine in a CA [42], smallest self-reproducing in a CA [40] and the smallest real-time prime number generator in a CA [1]. In mathematics, there was the search for the smallest number of necessary Boolean operators, in physics the minimization (unification) of theories, and for engineering, there is the search for minimizing (shrinking in size) electronic computers.

Appendix A

Simulation code

This appendix contains the program code for simulating the Toda lattice, and the results created by this program can be found in chapter 4. The code is written for Wolfram Mathematica version 12 and can be found here https://git.cs.york.ac.uk/mc1723/soliton-computing-in-the-toda-lattice-controllable-delay-and-logic-gates.git. The program has been made completely from scratch by the author, thus it does not contain any copyrighted materials, other than from the author himself. The first section gives a general description of the program written in pseudo code. This section finishes with a comparison between the results from the authors' program and the results from Kubota and Odagaki's paper on delay of a soliton in the Toda lattice [90]. The second section will be the code used for the controllable delay measurements, and the code used for simulating the logic gates. The amount of commentary in this second section will be kept to a minimum. The code can be copied into a Mathematica file and be used to obtain the results from chapter 4.

A.1 Pseudo code

We start with the code for creating the controllable delay, followed by the code for simulating the new logic gates designs.

A.1.1 Controllable delay description

The very first step is to have a piece of code that creates a static Toda lattice. After that we initialize a soliton into this static lattice.

```
ListCurrentPosition[];
ListPreviousPosition[];
E0 = TotalEnergy[k0];
```

We create two lists of positions, one being the current positions of all the sites, using equation 3.20 to determine the positions of the one dimensional lattice together with the initialized soliton(s). The other list contains the previous positions of the sites in the lattice and the solitons where they would have been at time $t - \delta t$ using 4.3. This list is used for calculating the momentum P_n and for the Verlet integration to determine the

new position of the sites after a step δt in time. The set-up of the lattice that the code simulates is drawn in figure A.1.

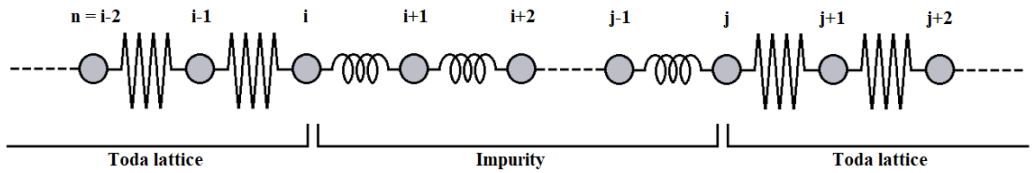


Figure A.1: *Redrawn figure of the experimental set-up, this image is also shown in figure 3.8. n indicates the label assigned to the sites. Between the two Toda lattices is an impurity made from $N = j - i$ sites connected by the harmonic potential.*

After the two lists of positions of the sites in lattices are determined, the total amount of energy `E0` stored in the lattice is calculated and this value is used later to normalizing the energy density function. The parameter `k0` is the wavenumber.

```
Main{
  TotalForceList[n=i] = Force(n=i-1) + Force(n=i+1);
  CheckDelay[SolitonDelayList];
  Verlet[];
  If Mod(t,SavingTime) = 0,
    Add CurrentTotelEnergy to StoreEnergyForEverySiteList[];
  }
  t = t + TimeStep;
}

Plot3D(CalculateEnergyOverTime[]);
```

When the preparations are done, we enter the main program. Here we simulate the dynamics of the soliton in the lattice. We first calculate the total force on each site, the program does not store the data permanently, but at every step in the time, the forces on every site is recalculated, because the magnitude and direction of the force are only important for the function `verlet[]`. For most sites the force would be the sum of the force of their neighbouring site to their right and left. The only exception are the sites on both ends of the lattice, because they only have one neighbouring site. In the function `Force(n=i)` we also check if the force is caused by the harmonic or Toda potential. `CheckDelay` measures if a soliton has escaped the impurity, the function will then save the duration the soliton was trapped in a list named `SolitonDelayList`, an explanation how this is done can be found in section 4.2.2. It is in the `SolitonDelayList` list that we used to create the results that were discussed in chapter 4. This is followed by Verlet integration, which is a method to simulate dynamics of the lattice and from this method the new position of the sites due to the force exerted of them is calculated and changed in `ListCurrentPosition[]`. Before the current site position is changed, its current position is copied into the previous position list `ListPreviousPosition[]`. Every `SavingTime` amount of seconds we store the current energy of every site in a list, the energy is determined by equation 3.27b with the momentum defined by 4.4. If we collected the energy of every site at every δt step, it would diminish the readability of

the graphs, we therefore collect only at a certain number of steps that have past. An example of the readability of the graph can be seen in figure 4.2. After the simulation has run for a predetermined period of time, we plot the energy of every site, against the sites and at the moments in time the data of the simulation has been stored.

A.1.2 Logic gate simulation description

The majority of the program for the simulation logic gates is the same as the previous program for measuring the controllable delay. But in the previous program where we were working in one dimension, we are now working in two dimensions with three connected Toda lattices together and one of the lattices containing the impurity. The set-up is redrawn in figure A.2.

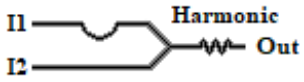


Figure A.2: *Redrawn set-up for the logic gate design in the Toda lattice. Two lattices are used as the inputs named* `Up` *for I1 and* `Down` *for I2, they are together connected to a third one* `Middle` *and used as the output of the logic gate. In the* `Middle` *lattice is also an impurity incorporated, made from sites connected by the harmonic potential.*

```
ListCurrentMiddlePosition[];
ListCurrentUpPosition[];
ListCurrentDownPosition[];
ListPreviousMiddlePosition[];
ListPreviousUPosition[];
ListPreviousDownPosition[];
E0 = TotalEnergy[k0, Up, Down, Middle];
```

Rather than having only one lattice to prepare, initialize solitons and calculate its previous position, we have three. The two lattices named *Up* and *Down* will function as our inputs and in the lattice named *Middle* the logic operations will be performed and this lattice will function as our output. It is this middle lattice that the impurity has been incorporated. In the previous code we only needed to calculate the energy of one lattice, now we need to know the total amount of energy of the whole set-up together to properly normalize the energy of every site, to allowing us to compare the different simulations.

```
Main{
  ListForceUp[n=i] = Force(n=i-1,Up) + Force(n=i+1,Up);
  ListForceDown[n=i] = Force(n=i-1,Down) + ForceDown(n=i+1,Down);
  ListForceMiddle[n=i] = Force(n=i-1,Middle) + Force(n=i+1,Middle);
  ListForceBranch[n=i] =
     Force(n=i-1,Up)+Force(n=i-1,Down) + Force(n=i+1,Middle);
  Verlet[Up,Down,Middle];
  If Mod(t,SavingTime) = 0
        Add CurrentTotelEnergy to
       StoreEnergyListUp[];
       StoreEnergyListDown[];
```

```
        StoreEnergyListMiddle[];
    }
    t = t + TimeStep;
}

Plot3D(CalculateEnergyOverTime[Up,Down,Middle]);
```

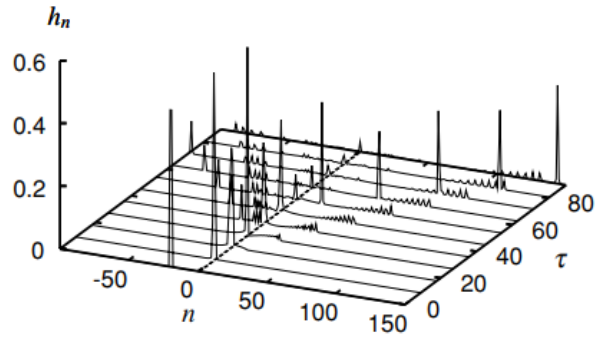
The next step is to calculate the total amount of force, just as before, but now for all three lattices separately. There is one site that connects all three lattices, this site has been named the `Branching site`. This required a separate code to calculate its total force. The branching site does form a part of the `Middle` lattice. The next step as was done before, is to perform the Verlet integration and change the positions of the lattice one δt step in time. We then have to save the total amount of energy of every lattice, we do this because eventually, we are plotting the energy of every site against the moment in time. It is therefore important for demonstrating the logic gate functions to show the input lattices separately, rather than having them atop of each other in the graph.

A.1.3 Comparing results

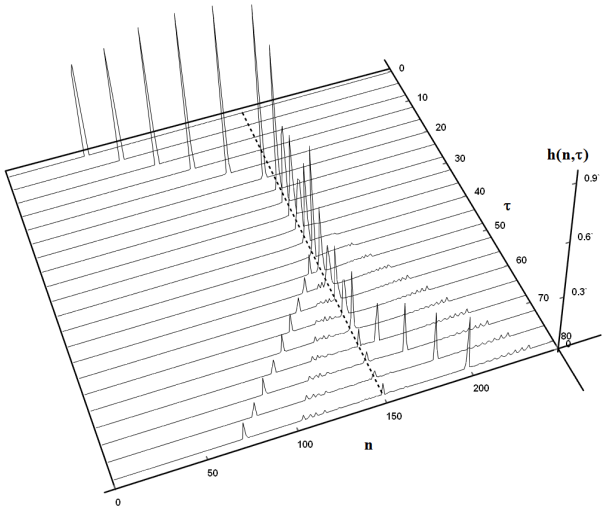
The program that was used to produce the results in chapter 4 has been validated by comparing the duration of the delay and the graphs from both the simulation of the author and from the work done by Kubota and Odagaki [90]. Only after the program of the author agreed the theory explained in chapter 3 and with the results of Kubota and Odagaki, did the author perform the measurements on the controllable delay and tested the functionality of the OR and XOR gate designs.

The duration of the delay was measured in the controllable delay program described in section A.1.1 and compared with the results shown in Kubota and Odagaki's paper [90, figure 5]. In their work they showed a graph of the harmonic potential constant against the trapping time τ_{trap}. The program of the author was able to produce the same trapping 'time' values as those seen in Kubota and Odagaki's work. A graph from both simulations is shown in figure A.3, were we see a soliton travelling in the Toda lattice and interacting with an impurity in the form of a series of harmonic potentials.

Figure A.3 shows that the behaviour of both simulation are the same. In both figures after the collision ripples escape the impurity, followed by a reflection soliton and ripples in the opposite direction. In figure A.3(a) the soliton is trapped in the impurity between the dimensionless 'time' $\tau = 0 - 8$ and escapes between $\tau = 44 - 52$. Therefore the soliton has been trapped for a duration of about 44 moments in 'time'. This correspondence to figure A.3(b), were the soliton enters the impurity between $\tau = 20 - 24$ and escapes between $\tau = 64 - 68$. The speed of the soliton moving through the lattice, before and after the collision, appears to be the same in both figures, as the transmitted soliton has moved about 50 sites further in the lattice after 20 moments in 'time'.



(a)



(b)

Figure A.3: *Comparing the simulation result of two different programs. Both programs show a soliton scattering against an impurity. This impurity is a series of harmonic potentials in the Toda lattice. After a delay, a soliton continues the journey and another soliton escapes the impurity as a reflection. The wavenumber in both simulations were $k_0 = 3.5$, a constant value of $\kappa = 0.03$ and the number of harmonic potentials was $N = 2$. (a) is a figure out of Kubota and Odagaki's paper [90, figure 4], (b) shows the result from the authors' program in the same situation. The lines on the plane in figure (a) are drawn every 8 dimensionless moments in 'time', while in (b) they are drawn every 4 moments in 'time'. The axes of both images are drawn from a different angle. Furthermore in figure (a) the sites have been numbered from -100 to 150, while the sites in (b) have been from 0 to 250. While the angle, site numbering and the time a soliton enters the impurity is different, however, both figures do show the same behaviour.*

A.2 Mathematica code

A.2.1 Controllable delay code

The start of the program is to set the values for the constants we are using. SSP is the Soliton Position and SpringPosition is the position of a site is attached to a harmonic potential. The wavenumber k needs to be >3 for transmission to appear, otherwise there are only reflections, explained in chapter 4.

```
a = 1; b = 1; m = 1; k = 4;
ka = 0.03;
timestep = 0.005; ParticleLength = 10; ParticleAmount = 550;
SSP = 270; SpringPosition = 400; BranchEndPosition = -60; distance = 350;
w = Sqrt[a b] Sinh[k];
```

`LastHarmonicSpring` the number of harmonic springs between both Toda lattices.

```
SpringList = {SpringPosition,SpringPosition + 1};
LastHarmonicSpring = Length[SpringList];
```

We are using Verlet integration, where 'a' in this expression is the acceleration found by f=ma

```
Verlet[x_, xOld_, a_]:= 2 x - xOld + a timestep^2
```

To find the time at which the soliton is at position SSP, we calculate 't'. A table is then made with the values for the x_n's, to create this soliton at SSP. OldSoliton is needed for the Verlet integration. Changing the OldSoliton list (t-timestep) into (t+timestep) will cause the soliton to move to the left rather than the right.

```
t = k SSP/(w);
Soliton =  Table[1/b (Log[
   1 + w^2 /a Sech[w *-t + k *n]^2]), {n, SSP - 5, SSP + 5, 1}];
OldSoliton = Table[1/b (Log[
   1 + w^2 /a Sech[w*-(t - timestep) + k *n]^2]), {n, SSP - 5, SSP + 5, 1}];

Solitonlist = {Soliton[[1]], Sum[Soliton[[i]], {i, 2}],
   Sum[Soliton[[i]], {i, 3}], Sum[Soliton[[i]], {i, 4}],
   Sum[Soliton[[i]], {i, 5}], Sum[Soliton[[i]], {i, 6}],
   Sum[Soliton[[i]], {i, 7}], Sum[Soliton[[i]], {i, 8}],
   Sum[Soliton[[i]], {i, 9}], Sum[Soliton[[i]], {i, 10}],
   Sum[Soliton[[i]], {i, 11}]};
OldSolitonlist = {OldSoliton[[1]], Sum[OldSoliton[[i]], {i, 2}],
   Sum[OldSoliton[[i]], {i, 3}], Sum[OldSoliton[[i]], {i, 4}],
   Sum[OldSoliton[[i]], {i, 5}], Sum[OldSoliton[[i]], {i, 6}],
   Sum[OldSoliton[[i]], {i, 7}], Sum[OldSoliton[[i]], {i, 8}],
   Sum[OldSoliton[[i]], {i, 9}], Sum[OldSoliton[[i]], {i, 10}],
   Sum[OldSoliton[[i]], {i, 11}]};
```

A list of position are created with one initialized soliton at SSP. Also a list is created with the positions at time-timestep. This list is used to keep trace of the positions of the particles in the lattice along the simulation.

```
ParticlePositionList = Table[ParticleLength*i, {i, 1, ParticleAmount}];
OldPositionList = ParticlePositionList;
len = Length[Soliton];
For[i = 1, i < ParticleAmount - SSP + 6 + 1, i++,
  If[i < len + 1,
   ParticlePositionList[[SSP - 6 + i]] =
    Soliton[[i]] + ParticlePositionList[[SSP - 6 + i - 1]] + ParticleLength;
   OldPositionList[[SSP - 6 + i]] =
    OldSoliton[[i]] + OldPositionList[[SSP - 6 + i - 1]] + ParticleLength;,
     ParticlePositionList[[SSP - 6 + i]] = ParticlePositionList[[SSP - 6 + i - 1]]
     + ParticleLength;
     OldPositionList[[SSP - 6 + i]] = ParticlePositionList[[SSP - 6 + i]];
  ]
];

(*This code can be used to place a second soliton at the right of
first soliton.'distance' is the number of particles away the second
soliton is from the first.*)
For[i = 1, i < ParticleAmount - distance + 6 + 1, i++,
  If[i < len + 1,
   ParticlePositionList[[distance - 6 + i]] =
    Solitonlist[[i]] + ParticlePositionList[[distance - 6 + i]];
   OldPositionList[[distance - 6 + i]] =
    OldSolitonlist[[i]] + OldPositionList[[distance - 6 + i]],
   ParticlePositionList[[distance - 6 + i]] =
    ParticlePositionList[[distance - 6 + i]] + Solitonlist[[11]];
   OldPositionList[[distance - 6 + i]] =
    ParticlePositionList[[distance - 6 + i]];
   ]
  ];
```

Momentum and potential energy formulas. `Potential2` is for the harmonic spring and `Potential1` is for the Toda potential. E0 is needed to normalize the energy value. In the code we are using dimensionless variables.

```
Vn[xnew_, xold_] := (xnew - xold)/(timestep);
rn[xn_, xnmin_] := ParticleLength + xnmin - xn;
Potential1[rn_, const_, a_] := a/b E^(-b rn) + a rn + const;
Potential2[rn_] := 1/2 rn^2 ka;

energyMemory = {}; For[i = 1, i <= ParticleAmount, i++,
 ra = If[i == 1, 0, rn[ParticlePositionList[[i]], ParticlePositionList[[i - 1]]]];
    (*Calculating the difference from the equilibrium distance of particles*)
 rb = If[i != ParticleAmount, rn[ParticlePositionList[[i + 1]],
   ParticlePositionList[[i]]], 0];
 Vm = Vn[ParticlePositionList[[i]], OldPositionList[[i]]];
   (*Calculating the momentum*)
 energy = (Vm^2 +
   If[MemberQ[SpringList, i], Potential2[b ra], Potential1[b ra, -a/b, a]] +
   If[MemberQ[SpringList - 1, i], Potential2[b rb], Potential1[b rb, -a/b, a]])/2;
 energyMemory = AppendTo[energyMemory, {0, i, energy}];];
 E0 = Sum[energyMemory[[i, 3]], {i, ParticleAmount}]
```

The code below lets soliton(s) move for `Endtime` seconds. At the particle position `SpringPosition`, the particle is connected with harmonic springs to particles `SpringPosition-1` and `SpringPosition+1`. The other springs follows the Toda potential. At the end of the simulation we have list of all the new position of the particle `(memoryList2)` and the positions at certain points in time `(memoryList3)`. The last list `(memoryList)` contains the force of every particle.

```
time = 0;
memorylist = {}; memorylist2 = {}; memorylist3 = {}; Endtime = 100/timestep;
DelayTime = {}; DelayTimeBool = False;
DelayTimeLeft = {}; DelayTimeBoolLeft = False; DelayTimeRight = {};
DelayTimeBoolRight = False; BooleanList = {False, False, False, False};
TauDelay = {, {}, {}};
For[j = 0, j < Endtime + timestep, j++,
  FList = Table[0, {n, 1, ParticleAmount,1}];
  (*Temperary list to trace the force for every step in time*)

  For[m = 1, m < ParticleAmount, m++,

   If[m != 1 && m != ParticleAmount,r1 = ParticleLength +
    ParticlePositionList[[m - 1]] - ParticlePositionList[[m]];
   (*Calculating the difference from the equilibrium distance of particles*)
    r2 = ParticleLength + ParticlePositionList[[m]] - ParticlePositionList[[m + 1]];
    forse = (If[MemberQ[SpringList, m], ka*r1, -Force[r1]] +
     If[MemberQ[SpringList - 1, m], -ka*r2, Force[r2]]);
     (*Calculate the total force the particle will feel*)
     ];

   If[m == 1 || m == ParticleAmount, FList[[m]] = 0;,
   (*To account for the boundaries of the lattice and avoiding an overflow
   in the program. The solitons do now "bounce back"*)
    FList[[m]] = forse;
    ];
   ];
```

```
(*The below two blocks is used to find the delay time*)
If[Abs[ParticlePositionList[[SpringPosition - 1]] -
    ParticlePositionList[[SpringPosition - 2]] -
    ParticleLength] > 0.01 && BooleanList[[1]] == False,
    BooleanList[[1]] = True;
    TauDelay[[1]] = time;]; (*Green line, incoming soliton collision moment, reflection*)
If[Abs[ParticlePositionList[[SpringPosition - 1]] -
     ParticlePositionList[[SpringPosition - 2]] - ParticleLength] >
     5 && BooleanList[[2]] == False,
     BooleanList[[2]] = True;];(*Green line, reflection*)
If[Abs[ParticlePositionList[[SpringPosition - 2]] -
     ParticlePositionList[[SpringPosition - 3]] - ParticleLength] >
     5 && BooleanList[[3]] == False,
     BooleanList[[3]] = True];(*Gray-Dashed line, reflection*)
If[Abs[ParticlePositionList[[
     SpringList[[LastHarmonicSpring]] + 1]] -
     ParticlePositionList[[SpringList[[LastHarmonicSpring]]]] -
     ParticleLength] > 10 && BooleanList[[4]] == False,
     BooleanList[[4]] = True];(*Black line, transmission*)

If[Abs[ParticlePositionList[[SpringPosition - 1]] -
     ParticlePositionList[[SpringPosition - 2]] - ParticleLength] <
     0.1 && BooleanList[[2]] == True,
     TauDelay[[2]] = AppendTo[TauDelay[[2]], time];
     BooleanList[[2]] = False;];(*Green line, reflection*)
If[Abs[ParticlePositionList[[SpringPosition - 2]] -
     ParticlePositionList[[SpringPosition - 3]] - ParticleLength] <
     0.1 && BooleanList[[3]] == True,
     TauDelay[[2]] = AppendTo[TauDelay[[2]], time];
     BooleanList[[3]] = False];(*Gray-Dashed line, reflection*)
If[Abs[ParticlePositionList[[
     SpringList[[LastHarmonicSpring]] + 1]] -
     ParticlePositionList[[SpringList[[LastHarmonicSpring]]]] -
     ParticleLength] < 0.1 && BooleanList[[4]] == True,
     TauDelay[[3]] = AppendTo[TauDelay[[3]], time];
     BooleanList[[4]] = False];(*Black line, transmission*)


For[m = 1, m < ParticleAmount, m++,
 (*There the new position is being calculated with Verlet*)
 tempPosition = ParticlePositionList[[m]];
 ParticlePositionList[[m]] = Verlet[ParticlePositionList[[m]], OldPositionList[[m]],
   FList[[m]]];
 OldPositionList[[m]] = tempPosition;
 ];

(*Data is being stored. The Mod[j,xx] is used to lower the number of data point.
This is needed to draw legible graphs*)
If[Mod[j*timestep, 0.5] == 0,
  memorylist = AppendTo[memorylist, {time, FList}];
  memorylist2 = AppendTo[memorylist2, {time, ParticlePositionList}];
  memorylist3 = AppendTo[memorylist3, {time, OldPositionList}];];
  time = time + timestep;
];
```

This block of code is used to plot the energy of the lattice.

```
energyMemory = {}; energyMemory2 = {}; len = Length[memorylist];
forceList2 = {}; leen = Length[memorylist[[1, 2]]];
For[j = 1, j < len + 1, j++, energyMemory = {};
 For[i = 1, i < leen + 1, i++,
  ra = If[i == 1, 0, rn[memorylist3[[j, 2, i]],
   memorylist3[[j, 2, i - 1]]]];
   (*Calculating the difference from the equilibrium distance of particles*)
  rb = If[i != leen, rn[memorylist3[[j, 2, i + 1]], memorylist3[[j, 2, i]]], 0];
  Vm = Vn[memorylist2[[j, 2, i]], memorylist3[[j, 2, i]]]; (*Calculating the momentum*)
  If[i < BranchEndPosition, energy = (Vm^2 +
    If[MemberQ[SpringList, i], Potential2[b ra], Potential1[b ra,-a(2 b), a /2]] +
    If[MemberQ[SpringList - 1, i], Potential2[b rb],
      Potential1[b rb, -a/(2 b), a/2]])/(2 E0 a/(2 b));,
       energy = (Vm^2 +
         If[MemberQ[SpringList, i], Potential2[b ra], Potential1[b ra, -a/b, a]] +
         If[MemberQ[SpringList - 1, i], Potential2[b rb], Potential1[b rb, -a/b, a]]
       )/(2 E0 a/b);
   ];
   (*Normalized energy density*)
   energyMemory = AppendTo[energyMemory, {memorylist2[[j, 1]], i, energy}];
  ];
 energyMemory2 = AppendTo[energyMemory2, Line[energyMemory]];
 (*Saving data. Line[] command is need to plot the data in the Graphs3D[] command*)
 ];
 Graphics3D[{Black, energyMemory2, Thick,
  Line[{{0, ParticleAmount, 0}, {80, ParticleAmount, 0}}],
  Line[{{0, 0, 0}, {0, ParticleAmount, 0}}], Dashed,
  Line[{{0, SpringPosition, 0}, {80, SpringPosition, 0}}]},
  Axes -> True, AxesLabel -> {"\[Tau]", "n", "h(n,\[Tau])"},
  AxesOrigin -> {80, 0, 0}, BoxRatios -> {1, 1, 0.5}, Boxed -> False,
  AxesStyle -> Thick, BaseStyle -> {FontWeight -> "Bold", FontSize -> 12}]
```

With the code below a graph can be created to check if the energy is conserved throughout the simulation.

```
memLen = Length[energyMemory];
For[z = 1, z < memLen, z++,
 If[energyMemory[[z, 3]] > 0.0005, Print[{z, energyMemory[[z, 3]]}]]
 ]

SumEnergy = 0; SumTotal = {}; tempMemory = {};
For[q = 1, q < len + 1, q++,
 tempMemory = energyMemory2[[q, 1]];
 SumEnergy = 0;
 For[h = 1, h < ParticleAmount + 1, h++,
  SumEnergy = SumEnergy + tempMemory[[h, 3]];
 ];
 SumTotal =
  AppendTo[SumTotal, {energyMemory2[[q, 1, 1, 1]], SumEnergy}];
 ];
 ListLinePlot[SumTotal, PlotRange -> All]
```

This code is used to see the path the particles takes (around the harmonic springs) over time.

```
len = Length[memorylist2]; temp50 = {}; temp49 = {};
temp51 = {}; temp48 = {}; temp52 = {}; temp47 = {};
temp46 = {}; temp30 = {}; temp40 = {};
For[j = 1, j < len + 1, j++,
 temp46 =
   AppendTo[temp46, {memorylist2[[j, 1]],
   -(memorylist2[[j, 2,SpringPosition - 4]] - ParticleLength)}];
 temp47 =
   AppendTo[temp47, {memorylist2[[j, 1]],
   -(memorylist2[[j, 2,SpringPosition - 3]] -
   memorylist2[[j, 2, SpringPosition - 4]] - ParticleLength)}];
 temp48 =
   AppendTo[temp48, {memorylist2[[j,1]],
   -(memorylist2[[j, 2, SpringPosition - 2]] -
   memorylist2[[j, 2, SpringPosition - 3]] - ParticleLength)}];
 temp49 =
   AppendTo[temp49, {memorylist2[[j, 1]],
   -(memorylist2[[j, 2, SpringPosition - 1]] -
   memorylist2[[j, 2, SpringPosition - 2]] - ParticleLength)}];
 temp50 =
   AppendTo[temp50, {memorylist2[[j,1]],
   -(memorylist2[[j, 2, SpringPosition]] -
   memorylist2[[j, 2, SpringPosition - 1]] - ParticleLength)}];
 temp51 =
   AppendTo[temp51, {memorylist2[[j, 1]],
   -(memorylist2[[j, 2, SpringPosition + 1]] -
   memorylist2[[j, 2, SpringPosition]] - ParticleLength)}];
 temp52 =
   AppendTo[temp52, {memorylist2[[j, 1]],
   -(memorylist2[[j, 2,SpringList[[LastHarmonicSpring]] + 1]] -
   memorylist2[[j, 2, SpringList[[LastHarmonicSpring]]]] -
   ParticleLength)}];
 temp30 =
   AppendTo[temp30, {memorylist2[[j, 1]],
   -(memorylist2[[j, 2, SpringList[[LastHarmonicSpring]] + 2]] -
   memorylist2[[j, 2, SpringList[[LastHarmonicSpring]] + 1]] -
   ParticleLength)}];
 ]

ListPlot[{temp48, temp49, temp52}, Joined -> True, PlotRange -> Full,
 PlotStyle -> {{Gray, Dashed}, {Black, Dotted}, Black},
 BaseStyle -> {FontWeight -> "Bold", FontSize -> 12}, Frame -> True,
 FrameLabel -> {"\[Tau]", "\!\(\*SubscriptBox[\(r\), \(n\)]\)"},
 PlotLegends -> {"Toda-2", "Toda-1", "Toda+1"}]
```

A.2.2 Logic gate simulation code

The repetition of commentary from the previous section has been left out, it will be clear to the reader which previous commentary correspondents to which blocks of code in this section.

```
a = 1; b = 1; m = 1; ParticleLength = 10; ParticleAmount = 300;
SSP = 80; k = 3.5; ka = 0.03; SpringPosition = 170;
distance = 115;(*position of the second soliton*)
BranchEndPosition = 130;(*Length of the branch chain*)
angle =0 Degree; (*Angle between the middle lattice and the upper and lower lattice*)
aBranch = a;(*factor difference between the main chain and the branch if required*)
w = Sqrt[aBranch b] Sinh[k];
timestep = 0.005;
```

```
SpringList = {SpringPosition, SpringPosition + 1};
LastHarmonicSpring = Length[SpringList];

t = k SSP/(\[Omega]);
Soliton =
  Table[1/b (Log[
      1 + w^2 /(aBranch b) Sech[w *-t + k *n]^2]),
      {n, SSP - 5, SSP + 5, 1}];
OldSoliton =
  Table[1/b (Log[
      1 + w /(aBranch b) Sech[w *-(t - timestep) + k *n]^2]),
      {n, SSP - 5, SSP + 5, 1}];

(*Use this only when one soliton needs to exits in either input of the chains*)
(*extraSpace = Sum[Soliton[[i]], {i, 1, Length[Soliton]}]*)

BranchPositionListUp =
  Table[{ParticleLength*i, 2*Sin[angle]}, {i, 1, BranchEndPosition}];
BranchPositionListDown =
  Table[{ParticleLength*i + extraSpace, 0}, {i, 1,BranchEndPosition}];
BranchOldListUp = BranchPositionListUp;
BranchOldListDown = BranchPositionListDown;
len = Length[Soliton];
For[i = 1, i < BranchEndPosition - SSP + 6 + 1, i++,
  If[i < len + 1,
   BranchPositionListUp[[SSP - 6 + i, 1]] =
    N[Soliton[[i]]] + BranchPositionListUp[[SSP - 6 + i - 1, 1]] +
     ParticleLength;
   BranchOldListUp[[SSP - 6 + i, 1]] =
    N[OldSoliton[[i]]] + BranchOldListUp[[SSP - 6 + i - 1, 1]] +
     ParticleLength;,
   BranchPositionListUp[[SSP - 6 + i, 1]] =
    BranchPositionListUp[[SSP - 6 + i - 1, 1]] + ParticleLength;
   BranchOldListUp[[SSP - 6 + i, 1]] =
    BranchPositionListUp[[SSP - 6 + i, 1]];
   ]
  ];

(Comment this for loop when you want to use only one soliton*)
For[i=1,i<BranchEndPosition-distance+6+1,i++,
If[i<len+1,
BranchPositionListDown[[distance-6+i,1]]=N[Soliton[[i]]]+\
BranchPositionListDown[[distance-6+i-1,1]]+ParticleLength;\
BranchOldListDown[[distance-6+i,1]]=N[OldSoliton[[i]]]+\
BranchOldListDown[[distance-6+i-1,1]]+ParticleLength;,
BranchPositionListDown[[distance-6+i,1]]=BranchPositionListDown[[\
distance-6+i-1,1]]+ParticleLength;
BranchOldListDown[[distance-6+i,1]]=BranchPositionListDown[[distance-\
6+i,1]]];
]
];

PositionRound =
  Round[BranchPositionListUp[[BranchEndPosition, 1]], 10^-10];
BranchPositionListMiddle =
  Table[{ParticleLength*i + PositionRound, Sin[angle]},
    {i, 1, ParticleAmount - BranchEndPosition}];
BranchOldListMiddle = BranchPositionListMiddle;

Vn[xnew_, ynew_, xold_, yold_] :=
 Sqrt[(xnew - xold)^2 + (ynew - yold)^2]/(timestep);
rn[xn_, yn_, xnmin_, ynmin_] :=
 Sign[ParticleLength + xnmin - xn]*
 Sqrt[(ParticleLength + xnmin - xn)^2 + (yn - ynmin)^2];
Potential1[rn_, const_, a_] := aBranch (a/b E^(-b rn) + a rn + const);
 Potential2[rn_] := 1/2 rn^2 ka;
 E0 =1*N[2 aBranch/b (Sinh[k] Cosh[k] - k)]

ExtraTime = 0.5;
```

```
EnergyModule[I_, ti_] :=
 Module[{i = I, tim = ti},
  If[i <= BranchEndPosition,

   ra = If[i == 1,
     0,
     rn[BranchPositionListUp[[i, 1]], BranchPositionListUp[[i, 2]],
      BranchPositionListUp[[i - 1, 1]],
      BranchPositionListUp[[i - 1, 2]]]
     ]; (*Calculating the difference from the equilibrium distance of particles*)
   rb = If[i != BranchEndPosition,
     rn[BranchPositionListUp[[i + 1, 1]],
      BranchPositionListUp[[i + 1, 2]], BranchPositionListUp[[i, 1]],
      BranchPositionListUp[[i, 2]]],
     rn[BranchPositionListMiddle[[1, 1]],
      BranchPositionListMiddle[[1, 2]] + Sin[angle],
      BranchPositionListUp[[i, 1]], BranchPositionListUp[[i, 2]]]
     ];
   Vm = Vn[BranchPositionListUp[[i, 1]], BranchPositionListUp[[i, 2]],
      BranchOldListUp[[i, 1]], BranchOldListUp[[i, 2]]];
   energyUp = (Vm^2 + Potential1[b ra, -aBranch/b, aBranch] +
       Potential1[b rb, -aBranch/b, aBranch])/2;
   energyMemoryUp = AppendTo[energyMemoryUp, {tim + ExtraTime, i, energyUp/E0}];

   ra = If[i == 1,
     0,
     rn[BranchPositionListDown[[i, 1]],
      BranchPositionListDown[[i, 2]],
      BranchPositionListDown[[i - 1, 1]],
      BranchPositionListDown[[i - 1, 2]]]
     ]; (*Calculating the difference from the equilibrium distance of particles*)
   rb = If[i != BranchEndPosition,
     rn[BranchPositionListDown[[i + 1, 1]],
      BranchPositionListDown[[i + 1, 2]],
      BranchPositionListDown[[i, 1]], BranchPositionListDown[[i, 2]]],
     rn[BranchPositionListMiddle[[1, 1]],
      BranchPositionListMiddle[[1, 2]] - Sin[angle],
      BranchPositionListDown[[i, 1]], BranchPositionListDown[[i, 2]]]
     ];
   Vm = Vn[BranchPositionListDown[[i, 1]],
     BranchPositionListDown[[i, 2]], BranchOldListDown[[i, 1]],
     BranchOldListDown[[i, 2]]];
   energyDown = (Vm^2 + Potential1[b ra, -aBranch/b, aBranch] +
     Potential1[b rb, -aBranch/b, aBranch])/2;
   energyMemoryDown =
    AppendTo[energyMemoryDown, {tim - ExtraTime, i, energyDown/E0}];
   ,

   ra = If[i == BranchEndPosition + 1,
     rn[BranchPositionListMiddle[[1, 1]],
      BranchPositionListMiddle[[1, 2]] + Sin[angle],
      BranchPositionListUp[[BranchEndPosition, 1]],
      BranchPositionListUp[[BranchEndPosition, 2]]],
     rn[BranchPositionListMiddle[[i - BranchEndPosition, 1]],
      BranchPositionListMiddle[[i - BranchEndPosition, 2]],
      BranchPositionListMiddle[[i - 1 - BranchEndPosition, 1]],
      BranchPositionListMiddle[[i - 1 - BranchEndPosition, 2]]]
     ];
   (*Only importent when the two branches come together*)
   rc = If[i == BranchEndPosition + 1,
     rn[BranchPositionListMiddle[[1, 1]],
      BranchPositionListMiddle[[1, 2]] - Sin[angle],
      BranchPositionListDown[[BranchEndPosition, 1]],
      BranchPositionListDown[[BranchEndPosition, 2]]],
     0
     ];
   rb = If[i != ParticleAmount,
     rn[BranchPositionListMiddle[[i + 1 - BranchEndPosition, 1]],
      BranchPositionListMiddle[[i + 1 - BranchEndPosition, 2]],
      BranchPositionListMiddle[[i - BranchEndPosition, 1]],
```

```
      BranchPositionListMiddle[[i - BranchEndPosition, 2]]],
     0
     ];
   Vm = Vn[BranchPositionListMiddle[[i - BranchEndPosition, 1]],
     BranchPositionListMiddle[[i - BranchEndPosition, 2]],
     BranchOldListMiddle[[i - BranchEndPosition, 1]],
     BranchOldListMiddle[[i - BranchEndPosition, 2]]];
   energyMiddle = (Vm^2 +
       If[MemberQ[SpringList, i], Potential2[b ra],
        Potential1[b ra,
         If[i == BranchEndPosition + 1, -aBranch/b, -a/b],
         If[i == BranchEndPosition + 1, aBranch, a]]] +
       If[MemberQ[SpringList - 1, i], Potential2[b rb],
        Potential1[b rb, -a/b, a]] +
       If[MemberQ[SpringList, i], Potential2[b rc],
        Potential1[b rc, -aBranch/b, aBranch]])/2;
   energyMemoryMiddle =
    AppendTo[energyMemoryMiddle, {tim, i, energyMiddle/E0}];

   ]
  ]

Force[r_, a_] := a (E^(-b  r) - 1)

Verlet[x_, xOld_, a_] := 2 x - xOld + a timestep^2

(*Program can only handle if the spring is away enough from the point
that the branches come together*)

time = 0;
energyMemoryUp2 = {};
energyMemoryDown2 = {};
energyMemoryMiddle2 = {};
memorylistUp = {};
memorylistDown = {}; memorylistMiddle = {};
Endtime = (80 + timestep)/timestep;
DelayTime = {}; DelayTimeBool = False; DelayTimeLeft = {};
DelayTimeBoolLeft = False; DelayTimeRight = {}; DelayTimeBoolRight = False;
BooleanList = {False, False, False, False}; TauDelay = {, {}, {}};
For[j = 0, j < Endtime, j++,
 FListUp =
  Table[0, {n, 1, BranchEndPosition, 1}]; (*temperary list to trace the force for every step in time*)
 FListDown = Table[0, {n, 1, BranchEndPosition, 1}];
 FListMiddle = Table[0, {n, 1, ParticleAmount - BranchEndPosition,1}];
 For[i = 1, i < ParticleAmount, i++,

 (*To accounts for the boundaries of the lattice and avoiding an
 overflow in the program. the solitons do now "bounce back"*)
  If[i == 1 , FListUp[[1]] = 0; FListDown[[1]] = 0;];
  If[i == ParticleAmount, FListMiddle[[ParticleAmount]] = 0];

  (*----Begin Block Force---*)
  If[i <= BranchEndPosition && i != 1,

   ra = rn[BranchPositionListUp[[i, 1]], BranchPositionListUp[[i, 2]],
     BranchPositionListUp[[i - 1, 1]],
     BranchPositionListUp[[i - 1, 2]]] ;

   rb = If[i != BranchEndPosition,
     rn[BranchPositionListUp[[i + 1, 1]],
       BranchPositionListUp[[i + 1, 2]], BranchPositionListUp[[i, 1]],
       BranchPositionListUp[[i, 2]]],
     rn[BranchPositionListMiddle[[1, 1]],
       BranchPositionListMiddle[[1, 2]] + Sin[angle],
       BranchPositionListUp[[i, 1]], BranchPositionListUp[[i, 2]]]
     ];
   FListUp[[i]] = (If[MemberQ[SpringList, i], ka*ra, -Force[ra, aBranch]] +
      If[MemberQ[SpringList - 1, i], -ka*rb, Force[rb, aBranch]]);
       (*calculate the total force the particle feels*)
```

```
 ra = rn[BranchPositionListDown[[i, 1]],
   BranchPositionListDown[[i, 2]],
   BranchPositionListDown[[i - 1, 1]],
   BranchPositionListDown[[i - 1, 2]]];

 rb = If[i != BranchEndPosition,
   rn[BranchPositionListDown[[i + 1, 1]],
     BranchPositionListDown[[i + 1, 2]],
     BranchPositionListDown[[i, 1]], BranchPositionListDown[[i, 2]]],
   rn[BranchPositionListMiddle[[1, 1]],
     BranchPositionListMiddle[[1, 2]] - Sin[angle],
     BranchPositionListDown[[i, 1]], BranchPositionListDown[[i, 2]]]
   ];
 FListDown[[i]] = (If[MemberQ[SpringList, i], ka*ra, -Force[ra, aBranch]] +
    If[MemberQ[SpringList - 1, i], -ka*rb, Force[rb, aBranch]]);
 ,

 ra = If[i == BranchEndPosition + 1,
   rn[BranchPositionListMiddle[[1, 1]],
     BranchPositionListMiddle[[1, 2]] + Sin[angle],
     BranchPositionListUp[[BranchEndPosition, 1]],
     BranchPositionListUp[[BranchEndPosition, 2]]],
   rn[BranchPositionListMiddle[[i - BranchEndPosition, 1]],
     BranchPositionListMiddle[[i - BranchEndPosition, 2]],
     BranchPositionListMiddle[[i - 1 - BranchEndPosition, 1]],
     BranchPositionListMiddle[[i - 1 - BranchEndPosition, 2]]]
   ];
 (*Only importent when the two branches come together*)
 rc = If[i == BranchEndPosition + 1,
   rn[BranchPositionListMiddle[[1, 1]],
     BranchPositionListMiddle[[1, 2]] - Sin[angle],
     BranchPositionListDown[[BranchEndPosition, 1]],
     BranchPositionListDown[[BranchEndPosition, 2]]],
   0
   ];
 rb = If[i != ParticleAmount,
   rn[BranchPositionListMiddle[[i + 1 - BranchEndPosition, 1]],
     BranchPositionListMiddle[[i + 1 - BranchEndPosition, 2]],
     BranchPositionListMiddle[[i - BranchEndPosition, 1]],
     BranchPositionListMiddle[[i - BranchEndPosition, 2]]],
   0
   ];
 FListMiddle[[i - BranchEndPosition]] =
  (If[MemberQ[SpringList, i], ka*ra, -Force[ra,
     If[i == BranchEndPosition + 1, aBranch, a]]] +
   If[MemberQ[SpringList - 1, i], -ka*rb, Force[rb, a]] +
     If[MemberQ[SpringList, i], -ka*rc, -Force[rc, aBranch]]);
 ]
(*----End Block Force---*)
];

(*Calculating the energy*)
If[Mod[j*timestep, 4] == 0,
  energyMemoryUp = {}; energyMemoryDown = {};
  energyMemoryMiddle = {};
  For[i = 1, i <= ParticleAmount, i++,
   EnergyModule[i, time]
   ];
  energyMemoryUp2 = AppendTo[energyMemoryUp2, Line[energyMemoryUp]];
  energyMemoryDown2 = AppendTo[energyMemoryDown2, Line[energyMemoryDown]];
  energyMemoryMiddle2 = AppendTo[energyMemoryMiddle2, Line[energyMemoryMiddle]];
  ];
```

```
  (*There the new position is being calculated with Verlet*)
  For[m = 1, m <= BranchEndPosition, m++,
    tempPositionUp = BranchPositionListUp[[m, 1]];
    BranchPositionListUp[[m, 1]] =
     Verlet[BranchPositionListUp[[m, 1]], BranchOldListUp[[m, 1]],
      FListUp[[m]]];
    BranchOldListUp[[m, 1]] = tempPositionUp;

    tempPositionDown = BranchPositionListDown[[m, 1]];
    BranchPositionListDown[[m, 1]] =
     Verlet[BranchPositionListDown[[m, 1]], BranchOldListDown[[m, 1]],
       FListDown[[m]]];
    BranchOldListDown[[m, 1]] = tempPositionDown;
    ];

  For[m = BranchEndPosition + 1, m < ParticleAmount, m++,
    tempPositionMiddle = BranchPositionListMiddle[[m - BranchEndPosition, 1]];
    BranchPositionListMiddle[[m - BranchEndPosition, 1]] =
     Verlet[BranchPositionListMiddle[[m - BranchEndPosition, 1]],
      BranchOldListMiddle[[m - BranchEndPosition, 1]],
      FListMiddle[[m - BranchEndPosition]]];
    BranchOldListMiddle[[m - BranchEndPosition, 1]] = tempPositionMiddle;
    ];

  (*Data is being stored. The Mod[j, xx] is used to lower the number of data point.
  This is needed to draw legible graphs*)
  If[Mod[j*timestep, 2] == 0,
    memorylistUp=AppendTo[memorylistUp,{time,BranchPositionListUp}];
    memorylistDown=AppendTo[memorylistDown,{time, BranchPositionListDown}];
    memorylistMiddle = AppendTo[memorylistMiddle, {time, BranchPositionListMiddle}];
    ];
  time = time + timestep;
  ];

Lengte = Length[energyMemoryUp2];
energyMemoryConnect = {};
energyMemoryConnect2 = {};
LineList = {};
For[g = 1, g <= Lengte, g++, tempTime = energyMemoryMiddle2[[g, 1, 1, 1]];
 energyMemoryConnect = AppendTo[energyMemoryConnect,
   Line[{{tempTime, BranchEndPosition,
      energyMemoryDown2[[g, 1, BranchEndPosition, 3]]}, {tempTime,
      BranchEndPosition + 1, energyMemoryMiddle2[[g, 1, 1, 3]]}}]];
 energyMemoryConnect2 = AppendTo[energyMemoryConnect2,
   Line[{{tempTime, BranchEndPosition,
      energyMemoryUp2[[g, 1, BranchEndPosition, 3]]}, {tempTime,
      BranchEndPosition + 1, energyMemoryMiddle2[[g, 1, 1, 3]]}}]];
 (*For creating a line between the two lattices*)
 LineList = AppendTo[LineList,
   Line[{{tempTime - ExtraTime, BranchEndPosition, 0},
   {tempTime + ExtraTime, BranchEndPosition, 0}}]];
 ]

Graphics3D[{energyMemoryUp2, energyMemoryConnect2, Black,
  energyMemoryMiddle2, energyMemoryConnect, energyMemoryDown2,
  LineList, Thick, Line[{{-4*ExtraTime, 0, 0}, {time + 2, 0, 0}}],
  Line[{{-4*ExtraTime, 0, 0}, {-2*ExtraTime, ParticleAmount, 0}}],
  Dashed, Line[{{0, SpringPosition, 0}, {time, SpringPosition, 0}}],
  Text["n", {time + 10, ParticleAmount/2, 0}],
  Text["\[Tau]", {time/2, ParticleAmount + 20, 0}]}, Axes -> True,
 AxesOrigin -> {time + 2, ParticleAmount, 0},
 AxesLabel -> {" ", " ", "h(n,\[Tau])"},
 LabelStyle -> {"Input", Black},
 Ticks -> {{0, 10, 20, 30, 40, 50, 60, 70}, {0, 50, 100, 150, 200,
    250, 300, 350}, {0, 0.30, 0.60, 0.90}}, BoxRatios -> {1, 1, 0.5},
 Boxed -> False, AxesStyle -> Thick,
 BaseStyle -> {FontWeight -> "Bold", FontSize -> 12}]
```

Appendix B

Alternative for Boolean logic

In this appendix, we go over alternative logic systems available that could be used for computing in physical systems as an alternative to Boolean logic. The two main alternatives are multi-valued and fuzzy logic. The most known logic used for computation is Boolean logic which is the set of $\{0, 1\}$ with modulo 2. With two inputs there are $2^2 = 4$ possible combinations and therefore there are $2^4 = 16$ operators possible. All the possible operators labelled f_i are shown in table B.1. Some of those operators can be used to construct all other operators, they are called *universal operators* or universal gate. Those are the NOR, NAND, f_2, f_3, f_{12} and f_{13}. The operators f_2, f_3, f_{12} and f_{13} require an extra condition to be universal.

BA	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
00	0	1	0	0	0	1	1	1	0	0	0	0	1	1	1	1
01	0	0	1	0	0	1	0	0	1	1	0	1	0	1	1	1
10	0	0	0	1	0	0	1	0	1	0	1	1	1	0	1	1
11	0	0	0	0	1	0	0	1	0	1	1	1	1	1	0	1
	0	NOR			AND	$\overline{B}$	$\overline{A}$	XNOR	XOR	A	B	OR			NAND	**1**

Table B.1: *All the possible outputs for two inputs. Some of the outputs are associated to certain gates. Two operators are input independent and four of them only rely on a single input.*

B.1 Introducing multi-valued and fuzzy logic

While Boolean logic has seen the most application in the computing field, there are however also other logic available. The first logic we are going to introduce is fuzzy or continuous logic. In fuzzy logic, the values are between 0 and 1 with including those numbers, written in mathematical notation the element $x \in [0, 1]$. Boolean logic is very useful when the question can be answered with either true or false, but when we ask another person if they agree with us when we ask if "the weather is warm today", they may reply with "it is a little bit warm". Fuzzy logic was made to accommodate those types of statements, where the answer or conclusion may lay between true and false. In fuzzy

logic, we can no longer write for every possible operator for every possible combination of two inputs, but there are certain defined operators [109]. The AND operator becomes $\min(x, y)$, which gives a value x or y depending on which of the two is a lower number. For the OR operator, we have $\max(x, y)$, which does the opposite of the previous operator. The last operator NOT or inverter is given by $1 - x$ for an input value x, the operators are summarized in table B.2.

x AND y	$\min(x, y)$
x OR y	$\max(x, y)$
NOT x	$1 - x$

Table B.2: *The operators in fuzzy logic. On the left is the operation and on the right how this is performed in fuzzy logic.*

Fuzzy logic has been used to create fuzzy controllers, which uses an analogue input to determine which actions to take. Those controllers have been employed for household appliances [129]. Furthermore, fuzzy logic is also used in human mind models that have to operate in an imprecise environment containing uncertainties [82]. An extension to fuzzy logic into the complex number is named complex logic [2] and this has been applied in the realizing of logic gates in nuclear magnetic resonance.

Another logic we are introducing is many-valued, also called multivalued logic. Technically fuzzy logic also belongs to this category, but in this work we will use the term many-valued logic to refer to logic that uses a bigger than 2 set of integers. From all possible multi-valued logic, ternary is perhaps the most studied and we will use it as the main example of multi-valued logic. Ternary logic uses a set of three number and is most often represented as $\{0, 1, 2\}$ or the balanced version $\{-1, 0, 1\}$. With two inputs there are $3^2 = 9$ possible combinations of input values, which means there are $3^9 = 19,683$ numbers of possible operators for two inputs. Ternary is called by some authors the 'Goldilock choice among numbering systems' [68], based on the calculations of W. Alexander in 1964 [6]. In those calculations, he argues that ternary is the most economic numbering system to represent other numbers in, more on this in the next section. Comparing binary to ternary, the latter will use less memory to store the same amount of information and more information can be processed at once. But the same can be said of any bigger than two number systems. Focussing on physical applications, the implementation of bigger number systems brings more complexity with it and most experimental settings do not allow for storing and processing four states (quaternary), eight states (octal) or other multivalued states. There are however some systems that will naturally concede bigger number systems, but seems to be fewer in numbers than the systems with binary. This is perhaps the reason nowadays most technology is based on binary, which is the smallest useful number system possible. Relays were used in early computing and they are binary in nature, its successor the vacuum tube is also binary in nature and its successor the transistor again binary. While binary counting has its golden age, it does not mean that binary is the most optimal counting system for future advances in technology. In the past people have tried to create a ternary computer, in 1958 [116] the researchers from the Moscow State University build the ternary computer SETUN. The research was ongoing for twenty years and was done under the guise of being an educational project

for engineers. From the first SETUN computer, there were 50 computers made and they were used across the USSR. An old overview of area were multi-valued logic has been used can be found in Hurst's work *Multiple-valued logic - its status and its future* [71]. Ternary and other multi-valued logic has not seen many applications in general, but even recently it has seen some uses [23, 39, 46, 70, 120].

B.2 Most economic base number

As was mentioned in the previous section, Alexander [6] has argued that the most economical integer to represent numbers is three. We will now be following his arguments. Let the base number, also called radix, be R, the number of digits by n and N is the maximum number we can represent in this base and with the number of digits[1]. N is given by

$$N = R^n - 1. \tag{B.1}$$

[2] This can be rewritten to an expression for the number of digits as a function of this maximal number

$$n = \mathrm{Log}_R(N+1) = \frac{\ln(N+1)}{\ln R}. \tag{B.2}$$

The next quantity that we are introducing is the radix economy E. This quantity was created to give some measurement of the effectiveness of the representation of a number in a given base, which in turn indicates the relative material cost for storing or displaying a given number. It is given by

$$E = nR = \frac{R\,\ln(N+1)}{\ln R}. \tag{B.3}$$

So far we have not required R to be an integer and for the next step, we will continue to do so, thus treating R as a continuous function. To find the minimum radix needed to express the number N, we can take the derivative with respect to the base of the above expression and putting that equal to zero

$$0 = \frac{dE}{dR} = \frac{\ln(N+1)}{\ln R} - \frac{\ln(N+1)}{(\ln R)^2} \Rightarrow \ \ln R = 1 \Rightarrow R = e. \tag{B.4}$$

So the most efficient or economical bases would be $e \approx 2.718$, the closest integer is three. Thus ternary can be seen as being more economic than binary when it comes to storing and representing numbers. A practical downside of ternary is that the distinction between states tend to be smaller than with binary. The authors in [68] mentions three examples where three has been found to be a better number system than two.

[1] In other parts of this thesis the letter R, n and N had a different definition, but in this section we will not be discussing the Toda lattice or harmonic springs, so this will not cause any confusion.

[2] Alexender did not include this extra -1 in his expression, for example in base 10 you can store the number 99 with two digits, but his expression with base 10 and two digits will give you the maximum number of $N = R^n = 10^2 = 100$ rather than 99. All authors after him do include this extra -1. It has however no bearing on his conclusion.

The examples that are mentioned are telephone menus, pan balance scale and folder order systems. For systems that do allow both binary and ternary, it will depend on which of the two logic is more practical, cheaper or faster. In 1970 Cassée and Strutt [26] gave an estimation of how much ternary would reduce the number of gates compared to binary needed when calculating with numbers $\ll 1$. They found that for every ternary gate you would need 1.59 binary gates. However, the same arguments for ternary can also be used for comparing quaternary with ternary and we would then find that quaternary would reduce the number of gates even further. But the biggest gain in reducing the number of gates going from one base into one higher, would be going from binary to ternary. From the reasoning given above, we can conclude that there are some arguments to be made in favour of ternary compared to the much applied binary logic.

References

[1] Y. Abe, K. Miyamoto and H. Umeo, "Construction of smallest real-time prime generators on cellular automata," in *Proceeding Second International Conference on Computer Technology and Development (ICCTD 2010)*, Cairo, Egypt (S. Mahmoud and Z. Lian, eds.), Institute of Electrical and Electronics Engineers, pp. 338-342.

[2] P.M. Aguiar, R. Hornby, C. Mcgarry, S. O'Keefe and A. Sebald, "Discrete and Continues system of logic in nuclear magnetic resonance," *Int. J. Unconv. Comput.,* vol. 12, issue 2-3, pp. 109-132, 2016.

[3] K.H. Ahn, B.C. Barnett, M.N. Islam and G.R. Williams, "Low-latency Soliton Logic Gate In EDFA's." *Int. Opt., Proc. IEE/LEOS Summer top. meeting,* 1995.

[4] K.H. Ahn, B.C. Barnett, K.O. Hill, M.N. Islam, B. Malo, M. Vaziri and G.R. Williams, "Soliton logic gate using low-birefringence fiber in a nonlinear loop mirror," *Opt. Lett.,* vol. 15, issue 20, pp. 1671-1673, 1995.

[5] K.H. Ahn, B.C. Barnett, X.D. Cao, D.Q. Chowdhury, K.O. Hill, M.N. Islam, B. Malo, M. Vaziri and G.R. Williams, "Experimental demonstration of a low-latency fiber soliton logic gate," *J. Light. Tech.,* vol. 14, issue 8, pp. 1768-1775, 1996.

[6] W. Alexander, "The ternary computer," *Electronics and Power,* vol. 10, issue 2, pp. 36-39, 1964.

[7] J. Ali, M.S. Aziz, S.M. Roslan, S. Saktioto, "Photonic AND OR gate by using fiber coupler," Laser Tech. Opt. vol. 7, issue 8, pp. 9-13, 2015.

[8] U. Al-Khawaja, S.M. Al-Marzoug and H. Bahllouli, "All-optical switches, unidirectional flow, and logic gates with discrete solitons in waveguide arrays," *Opt. Exp.,* vol. 24, issue 10, Art. no. 260807, 2016.

[9] P. Ambs, "Optical computing: A 60-year adventure," *Advances in optical technologies,* 2010, Art. no. 372652, 2010.

[10] I.S. Amiri, M.M. Ariannejad, H. Ahmad and M. Ghasemi, "Transmission performances of solitons in optical wired links," *App. Comp. Inf.,* vol. 13, issue 1, pp. 92-99, 2017. Arnold

[11] J.M. Arnold, "Complex Toda lattice and its application to the theory of interacting optical solitons," *J. Opt. Soc. Am. A,* vol. 15, issue 5, pp. 1450-1458, 1998.

[12] T.F. Assunção, M.L. Lyra, E.M. Nascimento and A.S.B. Sombra, "Phase-shift-controlled logic gates in Y-shaped nonlinearly coupled chains," *Phy. Rev. E,* vol. 93, issue 2, Art. no. 022218, 2016.

[13] A.G. Bakaokas, "An all-optical soliton FFT computational arrangement in the 3NLSE-domain," *UCNC 2016 Pro. 15th Int. Conf. Uncon. Comp. Nat. Comp.,* 2016, ISBN: 978-3-319-41311-2.

[14] A.G. Bakaokas and J. Edwards, "Computation in the 3NLS Domain Using First and Second Order Solitons," *Int. J. Uncon. Comp.,* vol. 5, issue 6, pp. 523-545, 2016.

[15] B.C. Barnett, C.S. Chiang and M.N. Islam, "Low-latency soliton logic gates in erbium-doped fiber amplifiers," *J. Opt. Soc. America B: Opt. Phy.,* vol. 11, issue 12, pp. 2394-2401, 1994.

[16] A. Barthakur and S. Nandy, "Pairwise three soliton interactions, soliton logic gates in coupled nonlinear Schrodinger equation with variable coefficients," *Commun. Nonlinear Sci. Numer. Simulat.,* vol. 69, pp. 370-385. 2019.

[17] V. Bazhanov, P. Dorey, K. Kajiwara and K. Takashi (Eds), "Special issue on fifty years of the Toda lattice," Special section, *J. Phys. A-Math. Theor.,* vol. 51, issue 6, 2018.

[18] M. Bechmann, M. Roselló-Merino, A. Sebald and S. Stepney, "Classical computing in nuclear magnetic resonance," *Int. J. Unconv. Comput.* vol. 6, issue 3-4, pp. 163-195, 2012.

[19] A. Bednyakova and S.K. Turitsyn, "Adiabatic soliton laser," *Phy. Rev. Lett.,* vol. 114, issue 11, Art. no. 113901 2015.

[20] E.R. Berlekamp, J.H. Conway and R.L. Guy, "Winning ways for your mathematical plays," *Academic Press,* vol. 4, pp. 940-957, 2004.

[21] K. Bhambri, N. Gupta and A. Raj, "Realization of all optical half adder by utilizing DM soliton pulses," *Int. J. Comp. Appl.,* vol. 92, issue 12, pp. 24-26, 2014.

[22] K. Bhambri, N. Gupta and A. Raj, "Realization of all optical full adder by utilizing DM soliton pulses," *Int. J. Comp. Appl.,* vol. 96, issue 19, pp. 13-16, 2014.

[23] P. Bhowmik, T. Chattopadhyay and J. N. Roy, "Design of all optical ternary logic based half adder circuit and it's applications," *J. Opt.,* vol. 44, issue 4, pp. 322-329, 2015.

[24] M. Bohr, "A 30 year retrospective on Dennard's MOSFET scaling paper," *IEEE Solid-State Circuits Society Newsletters,* vol. 12, issue 1, pp. 11-13, 2007.

[25] A. Butrym, B.A. Kochetov, V.R. Tuz and I. Vasylieva, "Logic gates on stationary dissipative soltions," *Phy. Rev. E,* vol. 99, Art. no. 052214, 2019.

[26] P.R. Cassée and M.J. Strutt, "Is there any advantage of ternary logic as compared with binary?" *IEEE T. Comput.,* correspondence June 1970, pp. 559.

[27] G.C. Chase, "History of mechanical computing machinery," *Ann. Hist. Comput.,* **2**(3), pp. 198-226, 1980.

[28] M.W. Chbat, M.N. Islam, P. Prucnal and C.E. Soccolich, "Energy Contrast from Cascaded Soliton Logic Gates," *Int. Pho. Res. WA1.,* 1992.

[29] M.W. Chbat, M.N. Islam, P.R. Prucnal and C.E. Soccolich, "Cascade of ultrafast soliton-dragging and trapping logic gates," *IEEE Pho. Tech. Lett.,* vol. 4, issue 9, pp. 1043-1046, 1992.

[30] M.W. Chbat, B.J. Hong, M.N. Islam and P.R. Prucnal, "Ultrafast soliton-trapping AND gate," *J. Light. Tech.,* vol. 10, issue 12, pp. 1215-1235, 1992.

[31] M.W. Chbat, M.N. Islam, J.R. Sauer and C.E. Soccolich, "Ultrafast logic gates in a soliton ring network," *Conference paper: Broadband Analogue and Digital Optoelectronics, Optical Multiple Access Networks, Integrated Optoelectronics, Smart Pixels,* LEOS 1992 Summer Topical, WA1.

[32] M.W. Chbat, C.R. Menyuk and S. Saxene, P.K.A. Wai, "Analysis of a soliton-based logic module for a ring network," *J. Light. Tech,* vol. 14, issue 8, pp. 1776-1787, 1996.

[33] C.-J. Chen, M.N. Islam, C.R. Menyuk and C.E. Soccolich, "Chirp mechanisms in soliton-dragging logic gates," *Opt. Lett.,* vol. 16, issue 4, pp. 214-216, 1990.

[34] C.-J. Chen and M.N. Islam, C.E. Soccolich, "All-optical time-domain chirp switches," *Opt. Lett.,* vol. 16, issue 7, pp. 484-486, 1990.

[35] C.-J. Chen, D.J. Digiovanni, M.N. Islam, C.M. Schoeder, J. R. Simpson and C.E. Soccolich "Prechirper to relax the timing restrictions for soliton-dragging logic gates," *Opt. Lett.,* vol. 16, issue 8, pp. 593-595, 1990.

[36] C.-J. Chen, C.R. Menyuk, S. Saxene and P.K. Wai, "Modeling of soliton-dragging logic gates with gain," *Opt. Lett.,* vol. 19, issue 17, pp. 1370-1372, 1994.

[37] M.-H. Chen, C.-H. Chu and Y.-D. Wu, "All-optical logic device using bent nonlinear tapered y-junction waveguide structure," *Fiber Integrated Opt.,* vol. 20, issue 5, pp. 517-524, 2001.

[38] S. Chi and T.-T. Shi, "Nonlinear photonic switching by using the spatial soliton collision," *Opt. Lett.,* vol. 15, issue 20, pp. 1123-1125, 1990.

[39] Y. Ching, H.K. Choi, C. Hong, N. Kim, S.-Y. Lee, D. Mahalu, M. Seo, V. Umansky, "Ballistic transport in quantum point contacts," *Sci. Rep.,* vol. 4, Art. no. 3806, 2014.

[40] H.-H. Chou, J.D. Lohn and J.A. Reggia, "Cellular Automata Models of Self-replicating Systems," *Advances in Computers,* M. Zelkowitz (ed), Academic Press, New York, vol. 47, pp.141-183, 1998.

[41] M. Conrad and K.-P. Zauner, "Conformation-based computing: a rationale and a recipe," in *Molecular computing,* MIT Press, Cambridge, 2003, chapter 1, pp. 1-31. ISBN: 9780262194877.

[42] M. Cook, "Universality in elementary cellular automata," *Complex systems,* vol. 15, issue 1, pp. 1-40, 2004. Copeland

[43] B.J. Copeland, "What is computing?" *Synthese,* vol. 108, issue 3, pp. 335-359, 1996.

[44] J. Cuevas-Maraver, P.G. Kevrekidis and F. Williams, *The sine-Gordon Model and its Applications*. Springer International Publishing, 2014, ISBN: 978-3-319-06721-6.

[45] L. Debnath, "A brief historical introduction to solitons and the inverse scattering transform-a vision of Scott Russell," *Int. J. Math. Educ. Sci. Tech.,* vol. 38, issue 8, pp. 1003-1028, 2007.

[46] V. Deibuk, V. Shults and I. Turchenko, "Optimized design of the universal ternary gates for quantum/reversible computing," *IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS),* 2015, Warsaw, Poland.

[47] R.H. Dennard, "Design of ion-implanted MOSFET's with very small physical dimensions," *IEEE J. Solid-state Circuits,* vol. 9, issue 5, pp. 256-268, 1974.

[48] D.J. DiGiovanni, B.J. Hong, M.N. Islam, J.R. Simpson and C.E. Soccolich, "Soliton-Dragging Logic Gate with an Erbium-Doped Fiber Amplifier," *Int. Pho. Res. TuF3.,* 1992.

[49] D.J. DiGiovanni, M.N. Islam, W. Pleibel, J.R. Simpson, C.E. Soccolich, E.A. de Souza and R.H. Stolen, "Soliton dragging in discrete and distributed amplifiers," *Quan. Opt.,* vol. 5, pp. 257-273, 1993.

[50] S. P. Dijaili, M. N. Islam and J. R. Sauer, "A Soliton ring network," *J. Light. Tech.,* vol. 11, issue 12, pp. 2182-2190, 1993.

[51] N.J. Doran and D. Wood, "Soliton processing element for all-optical switching and logic" *J. Opt. Soc. Am. B,* vol. 4, issue 11, pp. 1843-1846, 1987.

[52] N. J. Doran D.S. Forrester and B. K. Nayar , "Experimental investigation of all-optical switching in fibre loop mirror device," *Electron. Lett.,* vol. 25, issue 4, pp. 267-269, 1989.

[53] D.B. Duncan, Heriot-Watt University, Department of Mathematics. (2001). "Soliton on the Scott Russell Aqueduct on the Union Canal near Heriot-Watt University." [online]. Available: http://www.ma.hw.ac.uk/solitons/soliton1.html [Accessed: 22 November 2019].

[54] D. Erni, V. Jandieri, R. Khomeriki, T. Onoprishvili, J. Pistora, "Digital signal processing in coupled photonic crystal waveguides and its application to an all-optical AND logic gate," *Opt. Quan. Elec.,* vol. 51, Art. no. 121, 2019.

[55] M.V. Facao, M.F. Ferreira, S.V. Latas and M.H. Sousa, "Optical solitons in fibers for communication systems," *Fiber integrated Opt.,* vol. 24, issue 3-4, pp. 287-313, 2005.

[56] G. Farmelo, "The strangest man: the hidden life of Paul Dirac, quantum genius," Faber and Faber, 2010, ISBN: 978-0571222865.

[57] A.C. Ferreira, A.F.G.F. Filho, W.B. (de) Fraga, G.F. Guimaraes, J.W.M. Menezes, C.S. Sobrinho and A.S.B. Sombra, "All-optical half-adder using all-optical XOR and AND Gates for optical generation of 'Sum' and 'Carry' " *Fiber Int. Opt.,* vol. 29, issue 4, pp. 254-271, 2010.

[58] A.C. Ferreira, W.B. (de) Fraga, G.F. Guimaraes, F.T. Lima, M.L. Lyra, J.W.M. Menezes, and A.S.B. Sombra, "Study of the performance of an all-optical half-adder based on three-core non-linear directional fiber coupler under delayed and instantaneous non-linear kerr responses," *Fiber Int. Opt.,* vol. 30, issue 3, pp. 201-230, 2011.

[59] R.P. Feynman, "Simulating physics with computers," *Int. J. Theor. Phys.,* vol. 21, issue 467, pp. 467-488, 1982.

[60] R.P. Feynman, *Feynman lectures on computation* Addison-Wesley Publishing Company, Inc. 1996, ISBN: 0-201-48991-0.

[61] H. Flaschka and D.W. McLaughlin, *Canonically conjugate variables for the Korteweg-de Vries Equation and the Toda Lattice with periodic boundary conditions. Porg. Theor. Phys.,* vol. 55, pp. 438-456, 1976.

[62] F. Fredkin and T. Toffoli, "Conservative logic" *Int. J. Theor. Phys.,* 21, pp. 219-253, 1982.

[63] I. Gabitov, D.D. Holm, B.P. Luce and A. Mattheus, "Recovery of solitons with nonlinear amplifying loop mirrors," *Opt. Lett.* vol. 20, issue 24, pp. 2490-2492, 1995.

[64] G. Gaeta, "Results and limitations of soliton theory of DNA transcription," *J. Biol. Phys.,* vol. 24, pp. 81-96, 1999.

[65] A. Ghadi, "Phase sensitive, all-optical and self-integrated multi-logic AND, OR, XOR, and NOT gates" *Phys. Lett. A,* Art. no. 126432, 2020.

[66] D. Gopi, L. Kavitha and P. Sathishkumar, "Soliton-based logic gates using spin ladder," *Commun. nonlinear sci. numer. simulat.,* vol. 15, pp. 3900-3912, 2010.

[67] J.P. Gordon, M.N. Islam and C.E. Soccolich, "Ultrafast digital soliton logic gates," *Opt. Quan. Elec.,* vol. 24, issue 11, pp. 1215-1235, 1992.

[68] B. Hayes, "Third base," *American Scientist,* vol. 89, issue 6, pp. 490-494, 2001.

[69] C. Horsman, V. Kendon, S. Stepney and R.C. Wagner, "When does a physical system compute?" *Proc. Royal Soc. A,* vol. 470, Art. no. 20140182, 2014.

[70] H. Huacan, L. Yangtian and J. Yi, "Ternary optical computer architecture," *Phys. Scr.,* vol. 118, pp. 98-101, 2006.

[71] S.L. Hurst, "Multiple-valued logic - its status and its future," *IEEE T. Comput.,* vol. 33, issue 12, pp. 1160-1179, 1984.

[72] W. Isaacson, "Einstein: his life and universe," Simon and Schuster, 2007, ISBN: 9780743264730. Islam

[73] M.N. Islam, "Ultrafast all-optical logic gates based on soliton trapping in fibers," *Opt. Lett.,* vol. 14, issue 22, pp. 1257-1259, 1989.

[74] M.N. Islam, "All-optical cascadable NOR gate with gain," *Opt. Lett.,* vol. 15, issue 8, pp. 417-419, 1990.

[75] M.N. Islam, D.A.B. Miller and C. E. Soccolich, "Ultrafast All-Optical Fiber Soliton Logic Gates," *Ultr. Phen. VII,* pp 174-178, 1990.

[76] M.N. Islam, D.A.B. Miller and C.E. Soccolich, "Low-energy ultrafast fiber soliton logic gates," *Opt. Lett.,* vol. 16, issue 16, pp. 909-911, 1990.

[77] M.N. Islam and C.E. Soccolich, "Billiard-ball soliton interaction gates," *Opt. Lett.,* vol. 16, issue 19, pp. 1490-1492, 1991.

[78] M.N. Islam, *Ultrafast fiber switching devices and systems,* 1992, ISBN: 0-521-43191-3.

[79] M.N. Islam, L. Rahman and J.R. Simson, "Special erbium fiber amplifiers for short-pulse switching, lasers, and propagation," *J. Light. Tech.,* vol. 12, issue 11, pp. 1952-1962, 1994.

[80] M.N. Islam, "Advanced logic gates for ultrafast network interchanges," *AIP Conf. Proc.,* vol. 342, issue 605, pp. 605-617, 1995.

[81] K. Iwatsuki, S. Kawai and S. Nishi, "Demonstration of error free optical soliton transmission over 30000 km at 10 Gbit/s with signal frequency sliding technique," *Electron. Lett.,* vol. 31, issue 17, pp. 1463-1464, 1995.

[82] I.R. Jecintha and A.M.O. Teena, "Fuzzy logic, soft computing and applications," *Int. J. Comp. Alg.,* vol. 3, pp. 889-893, 2014.

[83] D.J. Kaup, "Nonlinear Schrödinger solitons in the forced Toda lattice," *Physica D,* vol. 25, issue 1-3, pp. 361-368, 1985.

[84] V. Kendon and S. Stepney, "The role of the Representational Entity in Physical Computing," *International Conference on unconventional computing and natural computing (UCNC),* 2019, Tokyo, Japan. Klein

[85] C. Klein and K. Roidot, "Numerical study of the long wavelength limit of the Toda lattice," *Nonlinearity,* vol. 28, pp. 2993-3025, 2015.

[86] T.C. Kofané, A. Mohamadou and C.B. Tabi, "Soliton excitation in the DNA double helix," *Phys. Scr.,* vol. 77, Art. no. 045002, 2008.

[87] M.D. Kruskal and N.J. Zabusky, "Interaction of 'solitons' in a collisionless plasma and the recurrence of initial states," *Phy. Rev. Lett.,* vol. 15, issue 6, pp. 240-243, 1965.

[88] H. Kubota and M. Nakazawa, "A dispersion-allocated soliton and its impact on soliton communication," *Physics and Applications of Optical Solitons in Fibres: Proceedings of the Symposium,* 1995, Kyoto, Japan.

[89] Y. Kubota and T. Odagaki, "Resonant transmission of a soliton across an interface between two Toda lattices," *Phys. Rev. E.,* vol. 71, Art. no. 016605, 2005.

[90] Y. Kubota and T. Odagaki, "Delay in a soliton transmission across an interface between two Toda lattices," *J. Phys. A: Math. Gen.,* vol. 39, pp. 12343-12353, 2006.

[91] Y. Kubota and T. Odagaki, "Logic Gates Based on Soliton Transmission in the Toda Lattice," *Adv. in App. Phy.,* vol. 1, pp. 29-38, 2013.

[92] T. Kuusela, "Soliton experiments in transmission lines," *Chaos Soliton Fract.,* vol. 5, pp. 2419-2462, 1995.

[93] R. Landauer, "Irreversibility and heat generation in the computing process," *IBM Journal of Research and Development* vol. 5, pp. 183-191, 1961.

[94] M. LaPedus, (24 June 2019), "5nm vs. 3nm," *Semiconductor Engineering* [online], available: http://semiengineering.com/5nm-vs-3nm/ [Accessed 28 June 2019].

[95] N. Li, Q. Liu and C. Tan, "All-optical logic gate based on manipulation of surface polaritons solitons via external gradient magnetic fields," *Phys. Rev. A.,* vol. 101, Art. no. 023818, 2020.

[96] S. Lloyd, "Ultimate physical limits to computation," *Nature,* vol. 406, pp. 1047-1054, 2000.

[97] P.S. Lomdahl, "What is a soliton?" *Los Alamos Science,* related topics, pp. 27-31, (1984).

[98] C.A. Mack, "Fifty years of Moore's law," *IEEE Trans. Semicond. Manuf.,* vol. 24, issue 2, pp. 202-207, 2011.

[99] B. MacLennan, "Natural computation and non-Turing models of computation," *Theor. Comput. Sci.,* vol. 317, pp. 115-146, 2004.

[100] W. Margulis, K. Rottwitt and J.R. Taylor, "Soliton recovery using a nonlinear amplifying loop mirror," *Electron. Lett.,* vol. 31, issue 5, pp. 395-396, 1995.

[101] S. Mitatha, N. Moongfangklanga, P. Phongsanam, A. Polar and P.P Yupapin, "Simultaneous all-optical logic AND and OR gates using photonic circuits," *Elec. Eng., Comp., Tele. and Inf. Tech. 2011 8th Int.*

[102] G.E. Moore, "Cramming more components onto integrated circuits." *Electronics*, vol. 38, issue 8, pp. 114-117, 1965.

[103] G.E. Moore, "Progress in digital integrated electronics," *International Electron Device meeting*, Washington, DC, United States, pp. 11-13, 1975.

[104] A. Mourachkine, *High-temperature superconductivity in cuprates: The nonlinear mechanism and tunneling measurements*. Kluwer Academic, Dordrecht, chapter 5, pp. 101-142, 2002, ISBN: 978-0-306-48063-8.

[105] V. Muto, A.C. Scott, P.L. Christiansen, "A Toda lattice model for DNA: Thermally generated solitons" *Physica D,* vol. 44, issue 1-2, pp. 75-91, 1990.

[106] A. Nakamura, "Interaction of Toda lattice soliton with an impurity atom," *Prog. Theor. Phys.,* vol. 59, pp. 1447-1460, 1978.

[107] Nature News, "Soliton wave receives crowd of admirers," *Nature*, vol. 376, issue 6539, pp. 373, 1995.

[108] T. Nilson and C. Schiebold, *Solution formulas for the two-dimensional Toda lattice and particle-like solutions with unexpected asymptotic behaviour. J. Nonlinear Math. Phy.,* vol. 27, issue 1, pp. 57-94, 2020.

[109] G. Nirmala and G. Suvitha, "Fuzzy logic gates in electronic circuits," *Int J Sci Res.,* vol. 3, issue 1, ISSN 2250-3153, 2013.

[110] M. Orenstein and J. Scheur, "All-optical gates facilitated by soliton interactions in a multilayered Kerr medium," *J. Opt. Soc. Am. B,* vol. 22, issue 6, pp. 1260-1267, 2005.

[111] F. Peper, "The end of Moore's law: Opportunities for natural computing?" *New Gener. Comput.,* vol. 35, pp. 253-269, 2017.

[112] A.M. Perelomov, "Remarks on the mass spectrum of two-dimensional Toda lattice of E_8 type," *J. Nonlinear Math. Phy.,* vol. 27, issue 1, pp. 12-16, 2020.

[113] A. Popescu and R.P. Singh, "An alternative solution to the electro-optic and service bottleneck problem in integrated multi-Gbit/s LANs: the SUPERLAN architecture," *Comput. Netw. ISDN Syst,* vol. 25, pp. 1089-1105, 1993.

[114] S. Rahimi-Keshari, T.C. Ralph and C.M. Caves, "Sufficient conditions for efficient classical simulation of quantum optics," *Phys. Rev. X,* vol. 6, 021039, pp.1-,13, 2016.

[115] R.V.J. Raja and T. Uthayakumar, "Logic gates based all-optical binary half adder using triple core photonic crystal fiber," *J. Opt.,* vol. 20, issue 6, Art. no. 065503, 2018.

[116] [https://mason.gmu.edu/∼drine/], D. Rine, "SETUN's reflections, how the SETUN computer was perceived in the 'Western' scientific community", published year unknown, [online], available: https://mason.gmu.edu/∼drine/TernaryComputers_SETUN_mirrored_July2005.html , [Accessed: 17 July 2018].

[117] R. Saul, "The origins of modern computing," *Department of Computer Science technical report*, Department of Computer Science, Purdue University, United States, 1990.

[118] E. Sperling, (23 May 2018), "Quantum effects at 7/5nm and beyond," *Semiconductor Engineering* [online], available: http://semiengineering.com/quantum-effects-at-7-5nm/ [Accessed 28 June 2019].

[119] G. I. Stegeman, S. Trillo, S. Wabnitz and E. M. Wright, "Soliton switching in fiber nonlinear directional couplers," *Opt. Lett.,* vol. 13, issue 8, pp. 672-674, 1988.

[120] S. Stockinger and O. Trapp, "A continuous and multi valued system as molecular answer for data processing and data storage," *Chem. Sci.,* vol. 5, issue 7, pp. 2677-2682, 2014.

[121] Texas Instruments, "SNx400, SNx4LS00, and SNx4S00 quadruple 2-input positive-NAND gates," Data sheet, *Texas Instruments,* 2017.

[122] S. Thongmee and P.P. Yupapin, "All-optical half adder/substractor using dark-bright soliton conversion control," *Procedia Eng.,* vol. 8, pp. 217-222, 2011.

[123] M. Toda, "Vibration of a chain with nonlinear interaction," *J. Phys. Soc. Japan,* vol. 22, pp. 431-436, 1967.

[124] M. Toda, *Theory of nonlinear lattices.* Springer-Verslag, 2nd ed. 1989, ISBN: 178-3-540-18327-3.

[125] V. Veerakumar, "Eletromagnetic soliton in magnetic and dielectric media," (PhD thesis, chapter 7, Bharathidasan University, Tiruchirappalli, India, supervisor M. Daniel), 22 October 2002. Williams

[126] M.R. Williams, "Early Calculations," in *Computing before computers,* Iowa State University Press, Ames, Iowa 50010, editor W. Aspray, 2000, chapter 1, pp. 42-49. ISBN: 0-8138-0047-1. Wu

[127] Y.-D. Wu, "Nonlinear all-optical switching device by using the spatial soliton collision," *Fiber Int. Opt.,* vol. 24, issue 5, pp. 387-404, 2004.

[128] P. Yupapin, "Novel All-optical Flip-Flop using Dark-Bright Soliton Conversion Control," *Inf. Tech. J.,* vol. 11, issue 10, pp. 1470-1476, 2012.

[129] L.A. Zadeh, "Fuzzy logic, neutral networks, and soft computing," *Commun. ACM,* vol. 37, issue 3, pp. 77-84, 1994.

[130] L. Zhang, "Computing naturally in the billiard ball model," *UC 09: Proceedings of the 8th International Conference on Unconventional Computation,* pp. 277-286, 2009, ISBN: 978-3-642-03744-3.