Quantum Neural Computation

International Series on
INTELLIGENT SYSTEMS, CONTROL, AND AUTOMATION:
SCIENCE AND ENGINEERING

VOLUME 40

For other titles published in this series, go to
www.springer.com/series/6259

Vladimir G. Ivancevic · Tijana T. Ivancevic

Quantum Neural Computation

Springer

Dr. Vladimir G. Ivancevic
Department of Defence
Defence Science &
Technology Organisation
(DSTO)
75 Labs.
Edinburgh SA 5111
Australia
vladimir.ivancevic@dsto.defence.gov.au

Dr. Tijana T. Ivancevic
School of Electrical &
Information Engineering
University of South Australia
Mawson Lakes Campus
Adelaide SA 5095
Australia
tijana.ivancevic@unisa.edu.au

ISBN 978-90-481-3349-9 e-ISBN 978-90-481-3350-5
DOI 10.1007/978-90-481-3350-5
Springer Dordrecht Heidelberg London New York

Library of Congress Control Number: 2009939613

Dedicated to Nick-Nitya, Atma and Kali

Preface

Quantum Neural Computation is a graduate-level monographic textbook. It presents a comprehensive introduction, both non-technical and technical, into modern quantum neural computation. Classical computing systems perform classical computations (i.e., Boolean operations, such as AND, OR, NOT gates) using devices that can be described classically (e.g., MOSFETs). On the other hand, quantum computing systems perform classical computations using quantum devices (quantum dots), that is devices that can be described only using quantum mechanics. Any information transfer between such computing systems involves a state measurement. This book describes this information transfer at the edge of classical and quantum chaos and turbulence, where mysterious quantum-mechanical linearity meets even more mysterious brain's nonlinear complexity, in order to perform a super-high-speed and error-free computations. This monograph describes a crossroad between quantum field theory, brain science and computational intelligence.

Quantum Neural Computation has six Chapters and Appendix. The Introduction gives a glimpse of what is to come later in the book, mostly various forms of quantum computation, quantum neural networks and quantum brain, together with modern adaptive path-integral methods. Chapter 2 gives a modern review of classical neurodynamics, including brain physiology, biological and artificial neural networks, synchronization, spike neural nets and wavelet resonance, motor control and learning. Chapter 3 presents a modern review of quantum physics, including quantum mechanics and quantum field theory (mostly Feynman pat-integral-based), as well as both Abelian and non-Abelian gauge theories (with their pat-integral quantizations). Chapter 4 presents several fields from nonlinear dynamics that are related to quantum neural computation, including classical and quantum chaos, turbulence and solitons, with the special treatment to *nonlinear Schrödinger equation* (NLS, the core model of quantum neural networks, in which quantum superposition meets neural nonlinearity). Chapter 5 gives a personalized review (mostly based on the authors' own papers) of the current research in quantum-brain and quantum-mind. Chapter 6 presents a review of quantum information,

quantum game theory, quantum computation and classical electronic for quantum computing. The Appendix gives a brief review of some mathematical and physiological concepts (necessary for comprehensive reading of the book), followed by classical computational tools used in quantum neural computation.

The objective of *Quantum Neural Computation* is to provide a serious reader with a serious scientific tool that will enable them to actually *perform* a competitive research in the rapidly-growing field of quantum neural computation. The monograph includes a very comprehensive bibliography on the subject and a detailed index.

Target readership for *Quantum Neural Computation* includes all researchers and students of complex, classical + quantum nonlinear systems (in computer science, physics, mathematics, engineering, medicine, chemistry, biology, psychology, sociology, economics, etc.), working in industry, clinics and academia.

Adelaide V. Ivancevic
May 2009 T. Ivancevic

Acknowledgments

The authors wish to thank Land Operations Division, Defence Science & Technology Organisation, Australia, for the support in developing the *Human Biodynamics Engine* (HBE) and crowd modelling simulator. In particular, we thank Dr. Darryn Reid, LOD–LRR Task Leader, for his support.

We also express our gratitude to *Springer* book series *Intelligent Systems, Control and Automation: Science and Engineering* and especially to the Editor, Professor Spyros Tzafestas.

Contents

1

Introduction

In this Introductory Chapter we give a glimpse of what is to come in later the book, mostly various forms of quantum computation, quantum neural networks[1] and quantum brain, together with modern adaptive path-integral methods.

[1] Roughly speaking, *artificial neural network* (ANN) is a system loosely modeled on the human brain (see, e.g. [II07b]). The field goes by many names, such as *connectionism*, *parallel distributed processing*, *neuro-computing*, natural intelligent systems, *machine learning* algorithms and ANNs. It is an attempt to simulate within specialized hardware or sophisticated software, the multiple layers of simple processing elements called neurons. Each neuron is linked to certain of its neighbors with varying coefficients of connectivity that represent the strengths of these connections. Learning is accomplished by adjusting these strengths to cause the overall network to output appropriate results.

Although currently there is a large variety of models for ANNs, they all share eight major aspects (for technical details, see Sect. 2.2 below): (i) A set of processing units, or 'neurons', represented by a set of integers; (ii) An activation for each unit, represented by a vector of time-dependent functions; (iii) An output function for each unit, represented by a vector of functions on the activations; (iv) A pattern of connectivity among units, represented by a matrix of real numbers indicating connection strength; (v) A propagation rule spreading the activations via the connections, represented by a function on the output of the units; (vi) An activation rule for combining inputs to a unit to determine its new activation, represented by a function on the current activation and propagation; (vii) A learning rule, which can be either unsupervised such as Hebbian [Heb49], or supervised, such as backpropriation [Hay98, II07b], for modifying connections based on experience, represented by a change in the 'synaptic weights' based on any number of variables; (viii) An environment which provides the system with experience, represented by sets of activation vectors for some subset of the units.

V.G. Ivancevic, T.T. Ivancevic, *Quantum Neural Computation*,
Intelligent Systems, Control and Automation: Science and Engineering 40,
DOI 10.1007/978-90-481-3350-5_1,

1.1 Neurodynamics

To give a brief introduction to classical neurodynamics, we start from the fully recurrent, N-dimensional, RC transient circuit, given by a nonlinear vector differential equation [Hay98, Kos92, II07b]:

$$C_j \dot{v}_j = I_j - \frac{v_j}{R_j} + w_{ij} f_i(v_i), \quad (i, j = 1, \dots, N), \tag{1.1}$$

where $v_j = v_j(t)$ represent the activation potentials in the jth neuron, C_j and R_j denote input capacitances and leakage resistances, synaptic weights w_{ij} represent conductances, I_j represent the total currents flowing toward the input nodes, and the functions f_i are sigmoidal.

Geometrically, (1.1) defines a smooth autonomous vector-field $X(t)$ in ND neurodynamical phase-space manifold M, and its (numerical) solution for the given initial potentials $v_j(0)$ defines the autonomous neurodynamical phase-flow $\Phi(t) : v_j(0) \to v_j(t)$ on M.

In AI parlance, (1.1) represents a generalization of three well-known recurrent NN models (see [Hay98, Kos92, II07b]):

(i) continuous Hopfield model [Hop84],
(ii) Grossberg ART-family cognitive system [CG83b], and
(iii) Hecht–Nielsen counter-propagation network [Hec87, Hec90].

Physiologically, (1.1) is based on the Nobel-awarded *Hodgkin–Huxley equation* of the neural action potential (for the single squid giant axon membrane) as a function of the conductances g of sodium, potassium and leakage [HH52a, Hod64]:

$$C\dot{v} = I(t) - g_{Na}(v - v_{Na}) - g_K(v - v_K) - g_L(v - v_L),$$

where bracket terms represent the electromotive forces acting on the ions.

The *continuous Hopfield circuit* model [Hop84]:

$$C_j \dot{v}_j = I_j - \frac{v_j}{R_j} + T_{ij} u_i, \quad (i, j = 1, \dots, N), \tag{1.2}$$

where u_i are output functions from processing elements, and T_{ij} is the inverse of the resistors connection-matrix becomes (1.1) if we put $T_{ij} = w_{ij}$ and $u_i = f_i[v_j(t)]$.

The Grossberg *analogous ART2 system* is governed by activation equation:

$$\varepsilon \dot{v}_j = -Av_j + (1 - Bv_j) I_j^+ - (C + Dv_j) I_j^-, \quad (j = 1, \dots, N),$$

where A, B, C, D are positive constants (A is dimensionally conductance), $0 \le \varepsilon \ll 1$ is the fast-variable factor (dimensionally capacitance), and I_j^+, I_j^- are excitatory and inhibitory inputs to the jth processing unit, respectively.

General *Cohen–Grossberg activation equations* [CG83b] have the form:

$$\dot{v}_j = -a_j(v_j)[b_j(v_j) - f_k(v_k)m_{jk}], \quad (j = 1, \ldots, N), \tag{1.3}$$

and the *Cohen–Grossberg theorem* ensures the global stability of the system (1.3). If

$$a_j = 1/C_j, \qquad b_j = v_j/R_j - I_j, \qquad f_j(v_j) = u_j,$$

and constant $m_{ij} = m_{ji} = T_{ij}$, the system (1.3) reduces to the Hopfield circuit model (1.2).

The Hecht–Nielsen *counter-propagation network* is governed by the activation equation [Hec87, Hec90]:

$$\dot{v}_j = -Av_j + (B - v_j)I_j - v_j \sum_{k \neq j} I_k, \quad (j = 1, \ldots, N),$$

where A, B are positive constants and I_j are input values for each processing unit.

Provided some simple conditions are satisfied, namely, say symmetry of weights $w_{ij} = w_{ij}$, non-negativity of activations v_j and monotonicity of transfer functions f_j, the system (1.1) is globally asymptotically stable (in the sense of Lyapunov energy functions). The fixed-points (stable states) of the system correspond to the fundamental memories to be stored, so it works as content-addressable memory (AM). The initial state of the system (1.1) lies inside the basin of attraction of its fixed-points, so that its initial state is related to appropriate memory vector. Various variations on this basic model are reported in the literature [Hay98, Kos92], and more general form of the vector-field can be given, preserving the above stability conditions.

1.2 Quantum Computation

Quantum computers promise to perform calculations believed to be impossible for ordinary computers. Some of those calculations are of great real-world importance. For example, certain widely used encryption methods could be cracked given a computer capable of breaking a large number into its component factors within a reasonable length of time. Virtually all encryption methods used for highly sensitive data are vulnerable to one quantum algorithm or another. The extra power of a quantum computer comes about because it operates on information represented as *qubits* (or, quantum bits, see Fig. 1.1) instead of bits of the conventional, or so-called *Von Neumann computer*.[2] Recall that an ordinary classical bit can be either a 0 or a 1, and standard microchip architectures enforce that dichotomy rigorously. A qubit,

[2] The so-called *Von Neumann architecture* [Neu58] is a design model for a stored-program digital computer that uses a *central processing unit* (CPU) and a single separate storage structure to hold both instructions and data. It is named after mathematician and early computer scientist John Von Neumann. Such a computer implements a *universal Turing machine*, and the common 'referential model' of spec-

in contrast, can be in a superposition state, which entails proportions of 0 and 1 coexisting together. One can think of the possible qubit states as *points on a sphere:* the north pole is a classical 1, the south pole a 0, and all the points in between are all the possible superpositions of 0 and 1. The freedom that qubits have to roam across the entire sphere helps to give quantum computers their unique capabilities (see [Col06]).

Quantum computers can be roughly divided into two classes: (i) discrete ones, pioneered by Richard P. Feynman several decades ago; and (ii) recently proposed topological ones. In this section we will give a brief, nontechnical description of both types. In later Chapters we will elaborate on the most interesting aspects (using the necessary technical machinery).

1.3 Discrete Quantum Computers

The concept of *quantum computing*, or *quantum computation*, was first stated by Feynman [Fey82] and Benioff [Ben82], and formalized by Deutsch [Deu85], Bernstein and Vazirani [BV93], and Yao [Yao93]. For review, see papers [Aha98, Ber97, RP98, Ste98], books [Gru99, WC98], and courses available on the web [Pre98b, Ven07].

---

ifying sequential architectures, in contrast with more recent parallel architectures. In particularly, a stored-program digital computer is one that keeps its program instructions as well as its data in read-write, random access memory. Stored-program computers were an advancement over the program-controlled computers of the 1940s, such as Colossus and ENIAC, which were programmed by setting switches and inserting patch leads to route data and control signals between various functional units. In the majority of modern computers, the same memory is used for both data and program instructions.

The separation between the CPU and memory leads to the *von Neumann bottleneck*, the limited throughput (data transfer rate) between the CPU and memory compared to the amount of memory. In modern machines, throughput is much smaller than the rate at which the CPU can work. This seriously limits the effective processing speed when the CPU is required to perform minimal processing on large amounts of data. The CPU is continuously forced to wait for vital data to be transferred to or from memory. As CPU speed and memory size have increased much faster than the throughput between them, the bottleneck has become more of a problem.

The performance problem is reduced by a *cache* between CPU and main memory, and by the development of branch prediction algorithms. Note that a cache is a collection of data duplicating original values stored elsewhere or computed earlier, where the original data is expensive to fetch (owing to longer access time) or to compute, compared to the cost of reading the cache. In other words, a cache is a temporary storage area where frequently accessed data can be stored for rapid access. Once the data is stored in the cache, future use can be made by accessing the cached copy rather than re-fetching or recomputing the original data, so that the average access time is shorter. A cache, therefore, helps expedite data access that the CPU would otherwise need to fetch from main memory.

Fig. 1.1. A qubit rotation can be visualized by a rotation of the unit vector in the so-called *Bloch sphere*. For example, here we have a spin-1/2 state of a qubit in a magnetic field (modified and adapted from [Tha99]).

Roughly, *quantum computer* is a computation device that makes direct use of distinctively quantum-mechanical phenomena, such as *superposition* and *entanglement*,[3] to perform operations on data. Whilst in a conventional computer information is stored as bits, in a quantum computer it is stored as quantum binary digits, or *qubits*. The basic principle of quantum computation is that the quantum properties can be used to represent and structure data, and that quantum mechanisms can be devised and built to perform operations with these data [GC98].

A classical computer uses strings of 0s and 1s. It can do calculations on only one set of numbers at once. A quantum computer uses quantum states which can be in a superposition of many different numbers at once. A classical computer is made up of bits while a quantum computer is made up of quantum

[3] *Quantum entanglement*, a phenomenon referred to by E. Schrödinger as 'the essence of quantum physics', is a property of quantum superpositions involving more than one system. Just as two classical bits can be in any of four states $(00, 01, 10, 11)$, the general quantum state of two qubits is a superposition of the form $c_{00}|00\rangle + c_{01}|01\rangle + c_{10}|10\rangle + c_{11}|11\rangle$ and the quantum state of N qubits can be represented by a complex-valued vector with $2N$ components. This is the basis of the exponential superiority of quantum computation: instead of N Boolean registers, one has $2N$ complex variables, even though there are only N physical switches. But to be computationally useful, the joint quantum state must be 'non-separable'. A separable state can be expressed as an abstract product of individual states:

$$|00\rangle = |0\rangle_A|0\rangle_B, \qquad |00\rangle + |01\rangle = |0\rangle_A(|0\rangle + |1\rangle)_B.$$

However, the so-called *Bell state*, $|00\rangle + |11\rangle$, cannot be factorized in this way, and is therefore non-separable. The entanglement of a state is a measure of its non-separability, and arguably represents the fundamental resource used in quantum computation [CT07].

bits, or qubits. A quantum computer can do an arbitrary reversible classical computation on all the numbers simultaneously. A quantum computer can be treated as an interacting multi-spin system. Technically, in quantum computation, the traditional Ising-like bits of a classical computer are promoted to Heisenberg-like spin-1/2 systems. Computation in an N-bit system then takes place via unitary transformations of a $2N$D Hilbert space generated by pairwise spin–spin interactions.

There are many routes to build a quantum computer. The 0 and 1 of a qubit might be the ground and excited states of an atom in a linear *ion trap* or of a *quantum dot*; they might be polarizations of photons that interact in an *optical cavity*; they might be the excess of one nuclear spin state over another in a liquid sample in an *nuclear magnetic resonance* (NMR) machine. As long as one can put the system in a *quantum superposition* and there is a way to interact multiple qubits, a system can potentially be used as a quantum computer. In order for a system to be a good choice, it should fulfill five criteria [Bar08]:

(i) be a scalable physical system with well-defined qubits;
(ii) be initializable to a simple fiducial state such as $|000\ldots\rangle$;
(iii) have much longer decoherence times (i.e., one can do many operations before losing quantum coherence);
(iv) have a universal set of quantum gates; and
(v) permit high quantum efficiency, qubit-specific measurements.

In particular, NMR is a very promising approach. The computers are molecules in a liquid, and information is encoded in atomic nuclei in the molecules. Instead of trying to coax results out of a few fragile qubits, the technique is based on manipulating, or, in effect, programming, enormous numbers of nuclei with radio-frequency pulses and then harnessing statistics to filter the right answers (about one result in a million) out of the background of noise.

If large-scale quantum computers can be built, they will be able to solve certain problems much faster than any of conventional computers, e.g., famous *Shor's algorithm*, which is a quantum algorithm for integer factorization, first introduced by mathematician Peter Shor in 1994. On a quantum computer, to factor an integer N, Shor's algorithm takes polynomial time in $\log N$, specifically $O((\log N)^3)$, demonstrating that integer factorization is in the *complexity class BQP*. This is exponentially faster than the best-known classical factoring algorithm, the general number field sieve, which works in sub-exponential time, about $O(2^{(\log N)^{1/3}})$. Shor's algorithm is important because it can, in theory, be used to 'break' the widely used public-key cryptography scheme known as RSA, which is based on the assumption that factoring large numbers is computationally infeasible. So far as is known, this assumption is valid for conventional computers; no classical algorithm is known that can factor in polynomial time in $\log N$. However, Shor's algorithm shows that factoring is efficient on a quantum computer, so an appropriately large quantum computer can 'break' RSA. It was also a powerful motivator for the design

and construction of quantum computers and for the study of new *quantum computer algorithms*.[4]

In particular, Shor's algorithm is based on the *quantum Fourier transform*, which is the discrete Fourier transform (DFT) with a particular decomposition into a product of simpler *unitary matrices* (see Appendix). Using this decomposition, the discrete Fourier transform can be implemented as a *quantum circuit*[5] consisting of *Hadamard transform gates*[6] and *controlled phase shifter gates*.[7] The quantum Fourier transform has many applications in quantum al-

[4] In 2001, Shor's algorithm was demonstrated by a group at IBM, who factored 15 into 3×5, using an NMR implementation of a quantum computer with 7 qubits [VSB01]. However, some doubts have been raised as to whether IBM's experiment was a true demonstration of quantum computation, since no entanglement was observed. Since IBM's implementation, several other groups have implemented Shor's algorithm using photonic qubits, emphasizing that entanglement was observed [LBY07].

[5] In quantum information theory, a quantum circuit is a model for quantum computation in which a computation is a sequence of reversible transformations on a quantum mechanical analog of an n bit register. This analogous structure is referred to as an n-qubit register.

To consider *quantum gates*, we need to specify the quantum replacement of an n-bit datum. The quantized version of classical n-bit space $\{0,1\}^n$ is given by $H_{\mathrm{QB}(n)} = \ell^2(\{0,1\}^n)$. This is by definition the space of complex-valued functions on $\{0,1\}^n$ and is naturally an *inner-product space*. This space can also be regarded as consisting of linear superpositions of classical bit strings. Using *Dirac bra-ket notation* (see Chap. 3), if $x_1, x_2, \ldots, x_n$ is a classical bit string, then

$$|x_1, x_2, \ldots, x_n\rangle$$

is an n-qubit; these special n-qubits (of which there are 2^n) are called *computational basis states*. All n-qubits are complex linear combinations of computational basis states. Note that HQB(n) has complex dimension 2^n.

[6] The *Hadamard transform* (also known as the Walsh–Hadamard transform) is an example of a generalized class of Fourier transforms. It performs an orthogonal, symmetric, involutary, linear operation on 2^n real numbers (or complex numbers, although the Hadamard matrices themselves are purely real). The Hadamard transform can be regarded as being built out of size-2 DFTs and is in fact equivalent to a multidimensional DFT of size $2 \times 2 \times \cdots \times 2 \times 2$. It decomposes an arbitrary input vector into a superposition of *Walsh functions*. The Hadamard transform can be computed in $n \log n$ operations, using the *fast Hadamard transform* algorithm.

Many quantum algorithms use the Hadamard transform as an initial step, since it maps n qubits initialized with $|0\rangle$ to a superposition of all 2^n orthogonal states in the $|0\rangle, |1\rangle$ basis with equal weight.

[7] Phase shifter gates operate on a single qubit. They are represented by 2×2 matrices of the form

$$R(\theta) = \begin{bmatrix} 1 & 0 \\ 0 & \mathrm{e}^{2\pi \mathrm{i} \theta} \end{bmatrix},$$

where θ is the phase shift.

gorithms as it provides the theoretical basis to the phase estimation procedure. This procedure is the key to quantum algorithms such as Shor's algorithm, the order finding algorithm and the hidden subgroup problem.

Quantum computers are different from other computers such as *DNA computers*[8] and traditional computers based on transistors. Some computing architectures such as *optical neural networks*[9] may use classical superposition

[8] DNA computing is a form of computing which uses DNA, biochemistry and molecular biology, instead of the traditional silicon-based computer technologies. DNA computing, or, more generally, molecular computing, is a fast developing interdisciplinary area. Research and development in this area concerns theory, experiments and applications of DNA computing. This field was initially developed by L. Adleman of the University of Southern California, in 1994 [Adl94]. Adleman demonstrated a proof-of-concept use of DNA as a form of computation which solved the seven-point *Hamiltonian path problem*. Since the initial Adleman experiments, advances have been made and various Turing machines have been proven to be constructible. DNA computing is fundamentally similar to parallel computing in that it takes advantage of the many different molecules of DNA to try many different possibilities at once. For certain specialized problems, DNA computers are faster and smaller than any other computer built so far. But DNA computing does not provide any new capabilities from the standpoint of computability theory, the study of which problems are computationally solvable using different models of computation.

[9] An optical neural network is a physical implementation of an artificial neural network (ANN, see Sect. 2.2 below) with optical components. Some ANNs that have been implemented as optical neural networks include the Hopfield net [YLS04] and the Kohonen self-organizing map with liquid crystals [LYG05]. While biological neural networks function on an electrochemical basis, optical neural networks use electromagnetic waves. Optical interfaces to biological neural networks can be created with optogenetics, but is not the same as an optical neural networks. In biological neural networks there exist a lot of different mechanisms for dynamically changing the state of the neurons, these include short-term and long-term synaptic plasticity. Synaptic plasticity is among the electrophysiological phenomena used to control the efficiency of synaptic transmission, long-term for learning and memory, and short-term for short transient changes in synaptic transmission efficiency. Implementing this with optical components is difficult, and ideally requires advanced photonic materials. Properties that might be desirable in photonic materials for optical ANNs include the ability to change their efficiency of transmitting light, based on the intensity of incoming light. There is one recent (2007) model of Optical Neural Network: the Programmable Optical Array/Analogic Computer (POAC). It had been implemented in the year 2000 and reported based on modified Joint Fourier Transform Correlator (JTC) and Bacteriorhodopsin (BR) as a holographic optical memory. Full parallelism, large array size and the speed of light are three promises offered by POAC to implement an optical CNN. They had been investigated during the last years with their practical limitations and considerations yielding the design of the first portable POAC version. POAC is a general purpose and programmable array computer that has a wide range of applications including: image processing; pattern recognition; target tracking; real-time video processing; document security; and optical switching.

of electromagnetic waves. However, it is conjectured that an exponential advantage over conventional computers is not possible without some specifically quantum mechanical resources such as *entanglement* [BCJ99].

In particular, a so-called *trapped ion quantum computer* is a type of quantum computer, in which ions (charged atomic particles) can be confined and suspended in free space using electromagnetic fields. Qubits are stored in stable electronic states of each ion, and quantum information can be processed and transferred through the collective quantized motion of the ions in the trap (interacting through the Coulomb force). Lasers are applied to induce coupling between the qubit states (for single qubit operations) or coupling between the internal qubit states and the external motional states (for entanglement between qubits). The fundamental operations of a quantum computer have been demonstrated experimentally with high accuracy (or 'high fidelity' in quantum computing language) in trapped ion systems and a strategy has been developed for scaling the system to arbitrarily large numbers of qubits by shuttling ions in an array of ion traps. This makes the trapped ion quantum computer system one of the most promising architectures for a scalable, universal quantum computer. As of June 2008, the largest number of entangled particles ever achieved in any quantum computer is eight calcium ions by way of the trapped ion method first achieved in 2005.

Generic Components of a Quantum Computer

A quantum computer has the following generic components:

1. Qubits: Any two-level quantum system can form a qubit, and there are two ways to form a qubit using the electronic states of an ion: (i) two ground state hyperfine levels (these are called 'hyperfine qubits'); and (ii) a ground state level and an excited level (these are called the 'optical qubits'). Hyperfine qubits are extremely long-lived (decay time of the order of thousands to millions of years) and phase/frequency stable (traditionally used for atomic frequency standards). Optical qubits are also relatively long-lived (with a decay time of the order of a second), compared to the logic gate operation time (which is of the order of microseconds). The use of each type of qubit poses its own distinct challenges in the laboratory.
2. Initialization: Ions can be prepared in a specific qubit state using a process called optical pumping. In this process, a laser couples the ion to some excited states which eventually decay to one state which is not coupled to by the laser. Once the ion reaches that state, it has no excited levels to couple to in the presence of that laser and, therefore, remains in that state. If the ion decays to one of the other states, the laser will continue to excite the ion until it decays to the state that does not interact with the laser. This initialization process is standard in many physics experiments and can be performed with extremely high fidelity ($> 99.9\%$).

3. Measurement: Measuring the state of the qubit stored in an ion is quite simple. Typically, a laser is applied to the ion that couples only one of the qubit states. When the ion collapses into this state during the measurement process, the laser will excite it, resulting in a photon being released when the ion decays from the excited state. After decay, the ion is continually excited by the laser and repeatedly emits photons. These photons can be collected by a photomultiplier tube (PMT) or a charge-coupled device (CCD) camera. If the ion collapses into the other qubit state, then it does not interact with the laser and no photon is emitted. By counting the number of collected photons, the state of the ion may be determined with a very high accuracy ($> 99.9\%$).
4. Arbitrary rotation of single qubit: One of the requirements of universal quantum computing is to coherently change the state of a single qubit. For example, this can transform a qubit starting out in 0 into any arbitrary superposition of 0 and 1 defined by the user. In a trapped ion system, this is often done using magnetic dipole transitions or stimulated Raman transitions for hyperfine qubits and electric quadrupole transitions for optical qubits. Gate fidelity can be greater than 99%.
5. Two-qubit entangling gates: Besides the controlled-NOT gate proposed by Cirac and Zoller in 1995, many equivalent, but more robust, schemes have been proposed and implemented experimentally since. Recent theoretical work has shown that there are no fundamental limitations to the speed of entangling gates, but gates in this impulsive regime (faster than 1 microsecond) have not yet been demonstrated experimentally (current gate operation time is of the order of microseconds). The fidelity of these implementations has been greater than 97%.
6. Scalable trap designs: Several groups have successfully fabricated ion traps with multiple trap regions and have shuttled ions between different trap zones. Ions can be separated from the same interaction region to individual storage regions and brought back together without losing the quantum information stored in their internal states. Ions can also be made to turn corners at a 'T' junction, allowing a two dimensional trap array design. Semiconductor fabrication techniques have also been employed to manufacture the new generation of traps, making the 'ion trap on a chip' a reality. These developments bring great promise to making a 'quantum charged-coupled device' (QCCD) for quantum computation using a large number of qubits.

Considerable interest has been generated in quantum computing since Shor [Sho97] showed that numbers can be factored in polynomial time on a quantum computer. From a practical viewpoint, Shor's result shows that a working quantum computer can violate the security of transactions that use the RSA protocol, a standard for secure transactions on the Internet. From a theoretical viewpoint, the result seemingly violates the polynomial version of the Church–Turing thesis; it is generally believed that factoring cannot be done

in polynomial time on a deterministic or probabilistic Turing machine. What makes Shor's breakthrough result possible on a quantum Turing machine is that exponentially many computations can be performed in parallel in one step and certain quantum steps enable one to extract the desired information.

Even though simple quantum computers have been built, enormous practical issues remain for larger-scale machines. The problems seem to be exacerbated with more qubits and more computation steps. In this section, we initiate the study of quantum computing within the constraints of using a polylogarithmic ($O(\log^k n)$, $k \geq 1$) number of qubits and a polylogarithmic number of computation steps. The current research in the literature has focused on using a polynomial number of qubits. Recently, researchers have initiated the study of quantum computing using a polynomial number of qubits and a polylogarithmic number of steps [MN98, Moo99, GHP00, CW00].

The concept of *quantum neural networks* (QNNs) was initially built in [GZ01] upon Deutsch's model of quantum computational network [Deu89]. The QNN model introduces a nonlinear, irreversible, and dissipative operator, called D gate, similar to the speculative operator introduced by [AL98]. We also define the precise dynamics of this operator and while giving examples in which nonlinear Schrödinger's equations are applied, we speculate on the possible implementation of the D gate.

Within a general framework of size, depth, and precision complexity, we study the computational power of QNNs. We show that QNNs of logarithmic size and constant depth have the same computational power as threshold circuits, which are used for modeling ANNs. QNNs of polylogarithmic size and polylogarithmic depth can solve the problems in NC, the class of problems that have theoretically fast parallel solutions. Thus, the new model subsumes the computation power of various theoretical models of parallel computation.

We believe that the true advantage of quantum computation lies in overcoming the communication bottleneck that has plagued the implementation of various theoretical models of parallel computation. For example, NC circuits elegantly capture the class of problems that can be theoretically solved fast in parallel using simple gates. While fast implementations of individual gates have been achieved with semiconductors and millions of gates have been put on a single chip, we do not have the implementation of full NC circuits because of the communication and synchronization costs involved in wiring a polynomial number of gates. We believe that this hurdle can be overcome using the nonlocal interactions present in quantum systems—there is no need to explicitly wire the entangled units and the synchronization is instantaneous. This advantage is manifest in the standard unitary operator, where operations on one qubit can affect probability amplitudes on all the qubits, without requiring explicit physical connections and a global clock. Thus, the new model has the potential to overcome the practical problems associated with both quantum computing as well as classical parallel computing.

There are two equivalent models for quantum computing, quantum Turing machines [Deu85, BV93] based on reversible Turing machines [Ben73, Ben89]

and quantum computational network [Deu89]. We briefly review the latter here. The basic unit in quantum computation is a *qubit*, a superposition of two independent states $|0\rangle$ and $|1\rangle$, denoted $\alpha_0|0\rangle + \alpha_1|1\rangle$, where α_0, α_1 are complex numbers such that $|\alpha_0|^2 + |\alpha_1|^2 = 1$. A system with n qubits is described using 2^n independent states $|i\rangle, 0 \leq i \leq 2^n - 1$, each associated with *probability amplitude* α_i, a complex number, as follows: $\sum_{i=0}^{2^n-1} \alpha_i|i\rangle$, where $\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1$. The direction of α_i on the complex plane is called the *phase* of state $|i\rangle$ and the absolute value $|\alpha_i|$ is called the *intensity* of state $|i\rangle$ [GZ01].

The computation unit in Deutsch's model consists of *quantum gates* whose inputs and outputs are qubits. A gate can perform any local unitary operation on the inputs. It has been shown that one-qubit gates together with two-qubit controlled NOT gates are universal [BBC95].

The quantum gates are interconnected by *wires*. A quantum computational network is a computing machine consisting of quantum gates with synchronized steps. By convention, the computation proceeds from left to right. The outputs of some of the gates are connected to the inputs of others. Some of the inputs are used as the input to the network. Other inputs are connected to *source* gates for 0 and 1 qubits. Some of the outputs are connected to *sink* gates, where the arriving qubits are discarded. An output qubit can be measured along state $|0\rangle$ or $|1\rangle$, and is observed based on the probability amplitudes associated with the qubit [GZ01].

Even though simple quantum computers have been built, enormous practical issues remain for larger-scale machines. Landauer [Lan95] exposes three main problems: *decoherence*, *localization*, and *manufacturing defects.* Decoherence is the process by which a quantum system decays to a classical state through interaction with the environment. In the best case, coherence is maintained for some 10^4 seconds, and, in the worst case, for about 10^{-10} seconds for single qubits. Some decoherence models show the coherence time declining exponentially as the number of qubits increases [Unr95]. Furthermore, the physical media that allow fast operations are also the ones with short coherence times.

The computation may also suffer from localization, that is, from reflection of the computational trajectory, causing the computation to turn around. Landauer points out that this problem is largely ignored by the research community [Lan95]. The combination of decoherence and localization makes the physical realization of quantum computation particularly difficult. On the one hand, we need to isolate a quantum computing system from the environment to avoid decoherence, and on the other hand, we need to control it externally to compel it to run forward to avoid reflection. Finally, minor manufacturing defects can engender major errors in the computations.

Introduction of the techniques of *error-correcting codes* and *fault-tolerant computation* to quantum computation has generated considerable optimism for building quantum computers, because these techniques can alleviate the problems of decoherence and manufacturing defects [GZ01]. Though this line

of research is elegant and exciting, the codes correct only local errors. For example, one qubit can be encoded into the nonlocal interactions among three qubits to correct one qubit errors. However, in principle, any (nonlocal) unitary operator can be applied to all the qubits. These nonlocal errors easily subvert the error-correcting codes. Also, nonlocal interactions provide the exponential speed-ups in quantum computing. The hope that nature might allow computational speed-ups via nonlocal interactions, while errors are constrained to occur only locally, seems unavailing. For an excellent exposition on error-correcting codes and fault-tolerant quantum computing, the reader is referred to [Pre97].

It has been shown that one-qubit gates together with two-qubit controlled NOT gates are universal [BBC95]; that is, any $2^n \times 2^n$ unitary operator can be decomposed into a polynomial number of one and two qubit operators. However, in general, any error operator can be applied in one step that cannot even be detected without observing all the involved qubits. Having the ability to operate on many qubits does not solve the problem, for error-correcting codes for k qubit errors can be subverted by a $(k+1)$-qubit error operator. Eventually, construction of a $2^n \times 2^n$ operator will itself be more time consuming than the actual computation.

There are some additional difficulties with computing using a polynomial number of qubits for a polynomial number of steps that are not discussed in the literature. For example, if $n = 1000$ and an $O(n^2)$ quantum algorithm is used, we need one million *uniquely identifiable* but *identical* carriers of quantum information. Clearly, the carriers need to be uniquely identifiable because we are not using their statistical properties, but encoding $2^{O(n^2)}$ computations in their interactions. However, the carriers need to be absolutely identical for the following reason. In describing the Hamiltonian for the whole system, there is a phase oscillation associated with each carrier. If all carriers have the same frequency, it does not affect the computation, which essentially changes the state relative to the global oscillation. But each qubit is likely to be encoded in carriers with a much larger state space, and even slight frequency differences can result in substantial errors over a polynomial number of steps. The task of preparing one million absolutely identical carriers, while exploiting the 2^{10^6} interactions, most of which are nonlocal, for speeding-up computation appears insurmountable. Controlling a polynomial number of entangled qubits for a polynomial number of steps, while compelling the computation forward, seems hard even with the help of error-correcting codes. To address the above problems, we initiate the study of quantum computation under the constraints of a poly-logarithmic number of qubits and a poly-logarithmic number of steps [GZ01].

1.4 Topological Quantum Computers

Unfortunately, quantum computers seem to be extremely difficult to build. The qubits are typically expressed as certain quantum properties of *trapped particles*, such as individual atomic ions or electrons. But their superposition states are exceedingly fragile and can be spoiled by the tiniest stray interactions with the ambient environment, which includes all the material making up the computer itself. If qubits are not carefully isolated from their surroundings, such disturbances will introduce errors into the computation. Most schemes to design a quantum computer therefore focus on finding ways to minimize the interactions of the qubits with the environment. Researchers know that if the error rate can be reduced to around one error in every 10,000 steps, then *error-correction procedures* can be implemented to compensate for decay of individual qubits. Constructing a functional machine that has a large number of qubits isolated well enough to have such a low error rate is a daunting task that physicists are far from achieving [Col06].

For this reason, a few researchers are pursuing a very different, topological way to build a quantum computer. In their approach the delicate quantum states depend on *topological properties* of a quantum system.[10] The so-called *topological quantum computer* is a theoretical quantum computer that employs 2D quasi-particles called *anyons*, whose *world lines* cross over one another to form *braids*[11] (see Fig. 1.2) in a $(1 + 2)$-space-time.

These braids form the *logic gates* that make up the quantum computer. The advantage of a quantum computer based on quantum braids over using *trapped quantum particles* is that the former is much more stable. While the smallest perturbations can cause a quantum particle to decohere and introduce errors in the computation, such small perturbations do not change the topological properties of the quantum braids (see Fig. 1.3). This is like the effort required to cut a string and reattach the ends to form a different braid, as opposed to a ball (representing an ordinary quantum particle in

[10] Recall that topology is called a rubber-sheet geometry, i.e., a geometrical study of properties that are unchanged when an object is smoothly deformed, by actions such as stretching, squashing and bending but not by cutting or joining. It embraces such subjects as *knot theory*, in which small perturbations do not change a topological property. For example, a closed loop of string with a knot tied in it is topologically different from a closed loop with no knot. The only way to change the closed loop into a closed loop plus knot is to cut the string, tie the knot and then reseal the ends of the string together. Similarly, the only way to convert a topological qubit to a different state is to subject it to some such violence.

[11] In topology, *braid theory* is an abstract geometric theory studying the everyday braid concept, and some generalizations. The idea is that braids can be organized into groups, in which the group operation is 'do the first braid on a set of strings, and then follow it with a second on the twisted strings'. Such groups may be described by explicit presentations, as was shown by E. Artin. Braid groups may also be given a deeper mathematical interpretation: as the fundamental group of certain configuration spaces (see Appendix, Sect. 7.2.1).

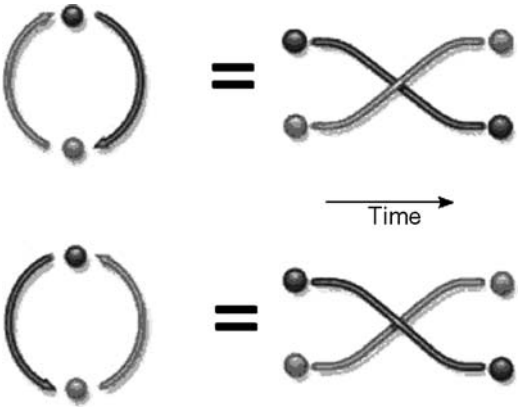

Fig. 1.2. Braiding: two anyons can encounter a clockwise swap (*top*) and a counterclockwise swap (*bottom*). These two moves in a plane generate all the possible braidings of the world lines of a pair of anyons (modified and adapted from [Col06]).

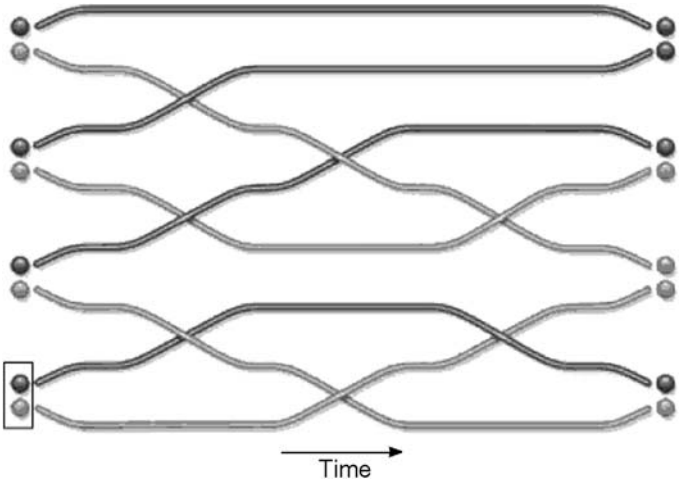

Fig. 1.3. Computing with braids of anyons. First, pairs of anyons are created and lined up in a row to represent the qubits, or quantum bits, of the computation. The anyons are moved around by swapping the positions of adjacent anyons in a particular sequence. These moves correspond to operations performed on the qubits. Finally, pairs of adjacent anyons are brought together and measured to produce the output of the computation. The output depends on the topology of the particular braiding produced by those manipulations. Small disturbances of the anyons do not change that topology, which makes the computation impervious to normal sources of errors (modified and adapted from [Col06]).

4D space-time) simply bumping into a wall. While the elements of a topological quantum computer originate in a purely mathematical realm, recent experiments indicate these elements can be created in the real world using semiconductors made of gallium arsenide near absolute zero and subjected to strong magnetic fields.

Anyons are quasi-particles in a 2D space. Anyons are not strictly fermions or bosons, but do share the characteristic of fermions in that they cannot occupy the same state. Thus, the world lines of two anyons cannot cross or merge. This allows braids to be made that make up a particular circuit. In the real world, anyons form from the excitations in an electron gas in a very strong magnetic field, and carry fractional units of magnetic flux in a particle-like manner. This phenomenon is called the *fractional quantum Hall effect*.[12] The electron 'gas' is sandwiched between two flat plates of gallium arsenide, which create the 2D space required for anyons, and is cooled and subjected to intense transverse magnetic fields.

When anyons are braided, the transformation of the quantum state of the system depends only on the topological class of the anyons' trajectories (which are classified according to the *braid group*, see Fig. 1.4, as well as Appendix, Sect. 7.2.1). Therefore, the quantum information which is stored in the state of the system is impervious to small errors in the trajectories.

In 2005, Fields Medalist Michael Freedman and collaborators from the Microsoft Station Q proposed a quantum Hall device which would realize a *topological qubit*. The original proposal for topological quantum computation is due to Alexei Kitaev from CalTex in 1997. The problem of finding specific braids for doing specific computations was tackled in 2005 by N.E. Bonesteel of Florida State University, along with colleagues from the Bell Laboratories. The team showed explicitly how to construct a so-called controlled NOT (or CNOT) gate to an accuracy of two parts in 10^3 by braiding six anyons (see Fig. 1.5). A CNOT gate takes two input qubits and produces two output qubits. Those qubits are represented by triplets (green and blue) of so-called

[12] The fractional quantum Hall effect (FQHE) is a physical phenomenon in which a certain system behaves as if it were composed of particles with charge smaller than the elementary charge. Its discovery and explanation were recognized by the 1998 Nobel Prize in Physics. The FQHE is a manifestation of simple collective behavior in a 2D system of strongly-interacting electrons. At particular magnetic fields, the electron gas condenses into a remarkable state with liquid-like properties. This state is very delicate, requiring high quality material with a low carrier concentration, and extremely low temperatures. As in the integer quantum Hall effect, a series of plateaus forms in the Hall resistance. Each particular value of the magnetic field corresponds to a filling factor (the ratio of electrons to magnetic flux quanta) $\nu = p/q$, where p and q are integers with no common factors. In particular, *fractionally charged quasi-particles* are neither bosons nor fermions and exhibit *anyonic statistics*. The FQHE continues to be influential in theories about topological order. Certain fractional quantum Hall phases appear to have the right properties for building a topological quantum computer.

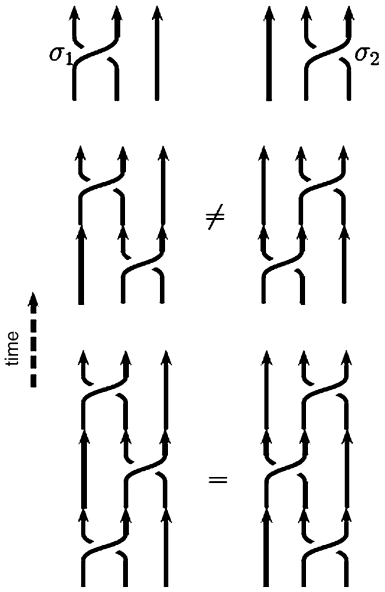

Fig. 1.4. Graphical representation of elements of the braid group. *Top*: the two elementary braid operations σ_1 and σ_2 on three anyons. *Middle*: non-commutativity is shown here as $\sigma_2\sigma_1 \neq \sigma_1\sigma_2$; hence the braid group is non-Abelian. *Bottom*: the *braid relation*: $\sigma_i\sigma_{i+1}\sigma_i = \sigma_{i+1}\sigma_i\sigma_{i+1}$ (modified and adapted from [NSS08]).

Fibonacci anyons. The particular style of braiding, leaving one triplet in place and moving two anyons of the other triplet around its anyons, simplified the calculations involved in designing the gate.

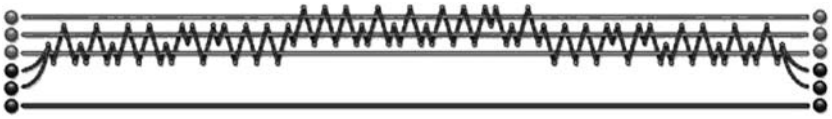

Fig. 1.5. Recently, a *quantum logic gate* known as a *CNOT-gate* has been produced by a complicated braiding of six anyons. This braiding produces a CNOT gate that is accurate to about 10^{-3} (modified and adapted from [Col06]).

Topological quantum computers are equivalent in computational power to other standard models of quantum computation, in particular to the *quantum circuit* model and to the *quantum Turing machine* model. That is, any of these models can efficiently simulate any of the others. Nonetheless, certain algorithms may be a more natural fit to the topological quantum computer model. For example, algorithms for evaluating the *Jones polynomial* were first developed in the topological model, and only later converted and extended in the discrete quantum circuit model.