We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900
Open access books available

185,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



Chapter

Quantum Algorithms for Nonlinear Equations in Fluid Mechanics

Rene Steijl

Abstract

In recent years, significant progress has been made in the development of quantum algorithms for linear ordinary differential equations as well as linear partial differential equations. There has not been similar progress in the development of quantum algorithms for nonlinear differential equations. In the present work, the focus is on nonlinear partial differential equations arising as governing equations in fluid mechanics. First, the key challenges related to nonlinear equations in the context of quantum computing are discussed. Then, as the main contribution of this work, quantum circuits are presented that represent the nonlinear convection terms in the Navier–Stokes equations. The quantum algorithms introduced use encoding in the computational basis, and employ arithmetic based on the Quantum Fourier Transform. Furthermore, a floating-point type data representation is used instead of the fixed-point representation typically employed in quantum algorithms. A complexity analysis shows that even with the limited number of qubits available on current and near-term quantum computers (<100), nonlinear product terms can be computed with good accuracy. The importance of including sub-normal numbers in the floating-point quantum arithmetic is demonstrated for a representative example problem. Further development steps required to embed the introduced algorithms into larger-scale algorithms are discussed.

Keywords: partial differential equations, fluid mechanics, nonlinear equations, quantum Fourier transform, floating-point arithmetic

1. Introduction

Quantum computing [1] and quantum communication are research areas that have seen significant developments and progress in recent years, as is apparent from the work covered in this book. In this chapter, the focus is on the development of quantum algorithms for solving nonlinear differential equations, highlighting key challenges that arise from the non-linearity of the equations to be solved. For this application of quantum computing, progress has so far been relatively limited and in this work, a promising approach to deriving efficient quantum algorithms is proposed. Although the focus is on non-linear equations related to fluid mechanics, the approach put forward here is applicable to a much wider range of problems.

Furthermore, in developing the proposed method, efficient quantum circuits involving floating-point arithmetic were created, in contrast to the more commonly used fixed-point arithmetic employed in a range of quantum algorithms. This aspect of the work described here should also be useful for a wider audience. In this work, the development of quantum algorithms for the nonlinear governing equations for fluid mechanics is described with a particular focus on representing the non-linear product terms in the equations. A key aspect of the derived quantum circuits in the present work is the (temporary) representation of the solution in the computational basis, along with the the use of a floating-point data representation in the arithmetic operations. The quantum circuits for obtaining the non-linear product terms are new developments and form the main contribution of this work. In recent years, a small number of works have considered quantum computing applications to fluid mechanics [2–8]. A brief review of this previous work will be presented in Section 2 and will provide context to the proposed approach. Related work on algorithms with representation in the computational basis is reviewed in this chapter. This chapter is structured as follows. Section 2 describes the background to the current work. Section 3 reviews the key challenges related to treating nonlinear differential equations in a quantum computing context, followed by a discussion of the nonlinear governing equations in fluids dynamics in Section 4. Section 5 then describes how nonlinear terms in governing equations can be evaluated in quantum algorithms using the computational basis. Section 6 and Section 7 discuss the quantum circuits used for computing the square of a floating-point number and the multiplication of two floating-point numbers, respectively. The simulation and verification of the derived quantum circuits is presented in Section 8. The complexity of the circuits is analyzed in Section 9. Finally, conclusions from this work and suggestions for further work are presented in Section 10.

2. Background of present work

For a small number of applications, quantum algorithms have been developed that display a significant speed-up relative to classical methods. Computational quantum chemistry is proving to be one of the key areas of application. Important developments for a wider range of applications include quantum algorithms for linear systems [9, 10] and the Poisson equation [11]. Applications to computational science and engineering problems beyond quantum chemistry have only recently begun to appear [4–6, 12–14]. Despite this research effort, progress in defining suitable engineering applications for quantum computers has been limited.

Significant progress has been made in recent years in the development of quantum algorithms for linear ordinary differential equations (ODEs) as well as linear partial differential equations (PDEs) [15–19]. However, in contract to this progress for linear equations, there has not been similar progress in the development of quantum algorithms for nonlinear ODEs and nonlinear PDEs. An early work by Leyton and Osborne [20] presented an innovative and highly ambitious algorithm. However, the computational complexity of this work involves exponential dependency on the time interval used in the time integration. A small number of more recent works have addressed nonlinear differential equations and typically algorithms for very specific problems were obtained [8]. Therefore, much research work is needed into quantum algorithms for a wider range of nonlinear problems.

Early work in quantum computing relevant to the field of Computational Fluid Dynamics (CFD) mainly involved the work on quantum lattice-gas models by

Yepez and co-workers [2, 3]. This work typically used type-II quantum computers, consisting of a large lattice of small quantum computers interconnected in nearest neighbor fashion by classical communication channels. In contrast to these quantum lattice-gas based approaches, the present study focuses on quantum algorithms designed for near-future 'universal' quantum computers. The potential of quantum computing in the context of direct numerical simulation of flows was reviewed recently by Griffin et al. [7], showing that a number of further developments are needed to make this approach viable.

Typically, there are two methods of encoding the result of a quantum algorithm: encoding within the computational basis of the quantum state and encoding within the amplitudes of the quantum state. The widely-used Quantum Fourier Transform (QFT) uses the second approach. The QFT with complexity $O(log^2(N))$ for problem size N has exponential speed-up compared to the classical fast Fourier transform (complexity $O(NlogN)$) and plays an important role in quantum computation as an essential part of many quantum algorithms. The exponential speed-up realized is due to superposition and quantum parallelism. However, in same quantum algorithms, the Fourier coefficients may be needed in the computational basis [21].

Here, the two different encoding methods are illustrated using the discrete Fourier Transform (DFT). The QFT performs the DFT in terms of amplitudes as,

$$\sum_{j=0}^{N-1} x_j|j\rangle \rightarrow \sum_{k=0}^{N-1} y_k|k\rangle \tag{1}$$

The QFT performs a DFT on a list of complex numbers, and the result is stored as amplitudes of a quantum state vector. In order to extract the individual Fourier components, measurements need to be performed on the quantum state vector. Therefore, the QFT is not directly useful for determining the Fourier-transformed coefficients of the input state. However, the QFT is widely used as a subroutine in larger algorithms. In contrast to the amplitude encoding in Eq. (1), Zhou et al. [21] presented a quantum algorithm computing the Fourier transform in the computational basis (termed QFTC). This quantum algorithm encodes Fourier coefficients with fidelity $1-\delta$ and digit accuracy ε for each Fourier coefficient. Its time complexity depends polynomially on $log\,(N)$, and linearly on $1/\delta$ and $1/\varepsilon$. The QFTC, enables the Fourier-transformed coefficient to be encoded in the computational basis as follows,

$$|k\rangle|0\rangle \rightarrow |k\rangle|y_k\rangle \tag{2}$$

where y_k corresponds to the fixed-point binary representation of $y_k \in (-1, 1)$ using two's complement format. In the algorithm proposed by Zhou et al. [21], the input vector $\vec{x}$ is provided by an oracle O_x such that,

$$O_x|0\rangle = \sum_{j=0}^{N-1} x_j|j\rangle \tag{3}$$

which can be efficiently implemented if $\vec{x}$ is efficiently computable or by using the qRAM that takes complexity $log\,(N)$ under certain conditions [21]. Comparing Eq. (1) and Eq. (2), it is clear that encoding in the computational basis requires a number of additional qubits depending on the required fixed-point representation.

3. Nonlinear problems on quantum computers

An early work by Leyton and Osborne [20] introduced a quantum algorithm to solve nonlinear differential equations with an unfavorable complexity. Since then, very few works have considered quantum algorithms for nonlinear equations. In contrast, algorithms for linear differential equations have continued to receive significant attention. As an example, advanced quantum spectral methods for differential equations were published recently by Childs and Liu [19].

A key contributing factor to the limited progress in algorithms for non-linear problems is the inherent linearity of quantum mechanics. For quantum algorithms encoding information as amplitudes of a quantum state vector, nonlinear (product) terms cannot be obtained by multiplying these amplitudes by themselves, as a result of the no-cloning theorem that prohibits the copying of an arbitrary quantum state. Furthermore, all quantum-gate operations (with the exception of measurements) in the quantum-circuit model used here need to be unitary and reversible. These requirements add further challenges to representing nonlinear terms when using the amplitude-based encoding approach. Specifically, in a normalized quantum state vector all amplitudes in the vector are ≤ 1 (unless only a single amplitude is non-zero), therefore an operator performing products of the amplitudes cannot be unitary since the resulting quantum state vector will no longer have a unit norm.

One possible way around these problems associated with nonlinear terms would be a hybrid quantum-classical approach where the nonlinear products are computed on a classical computer. However, due to the complexity introduced by measuring the quantum state (needed before each transfer of information to the classical computer) and the cost of (re-)initialization of the quantum computer with the result of these products, this is not a promising line of development. It is highly unlikely to lead to a quantum speed-up. Recently, Variational Quantum Computing (VQC) was introduced as an effective hybrid classical-quantum approach [22, 23], firstly for applications in quantum chemistry and more recently for a wider range of linear and nonlinear problems [24]. The VQC approach constructs the required solution from a layered network, as illustrated in **Figure 1**. As shown in **Figure 1(a)**, multiple layers are used (4 in the illustration), each taking as input multiple qubits (6 in example shown). Using depth 5 in the example, the quantum circuits defined

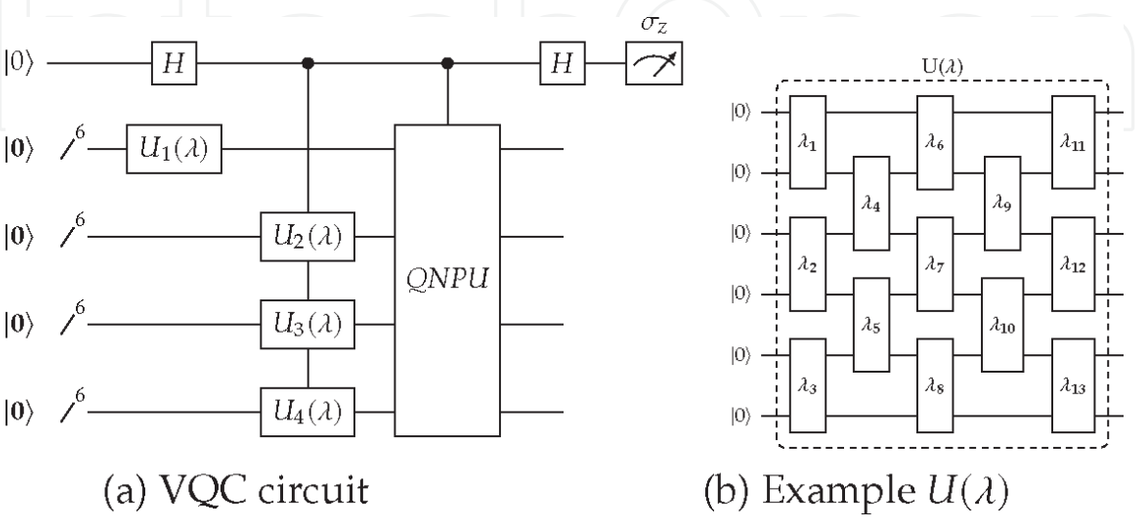


Figure 1.
Illustration of the Variational Quantum computing (VQC) approach (adapted from Lubasch et al. [24]).

by $U(\lambda)$ involve 13 two-qubit gates as shown in **Figure 1(b)**. Each of these gates has a parameter $\lambda_i \in [1, 13]$ associated with it. A classical computer is used to create optimized parameters λ employing an iterative approach that takes the measured state of the ancilla qubit as input. A further key part of the approach is the problem-specific Quantum Nonlinear Processing Unit (QNPU). Recently, Lubasch et al. [24] published an example for the QNPU for the nonlinear Burgers equation. In applications of the VQC approach, the efficiency strongly depends on the choice of the number of parameters λ used in $U(\lambda)$. The work by Lubasch et al. [24] showed that exponential speed-up is only possible if the depth of $U(\lambda)$ scales with the number of qubits and not with the overall problem size. It is clear that the proposed VQC approach is an important development toward QC applications to nonlinear problems. It therefore constitutes a leading candidate for applications to fluid dynamics. However, it is also clear that further investigation is needed to further assess its suitability for a range of applications.

4. Nonlinear governing equations in fluid mechanics

The Navier–Stokes equations for an incompressible, Newtonian fluid can be written as,

$$\frac{\partial \mathbf{U}}{\partial t} + \mathbf{U} \cdot \nabla_{\mathbf{x}} \mathbf{U} = -\frac{1}{\rho} \nabla_{\mathbf{x}} p + \nu \Delta \mathbf{U} \quad ; \quad \nabla_{\mathbf{x}} \cdot \mathbf{U} = 0 \tag{4}$$

where $\mathbf{U}$, p, ρ and ν are the velocity, pressure, density and kinematic viscosity, respectively. $\mathbf{x}$ denotes the coordinate in space. The second term on the right-hand side of Eq. (4) is the nonlinear convection term that poses a key challenge to developing efficient quantum algorithms for the Navier–Stokes equations. Efficient quantum algorithms for linear convection equations discretized on regular Cartesian meshes with periodic boundary conditions have been devised in recent years [6]. When studying numerical methods for the Navier–Stokes equations, it is often useful to switch to Burgers' model equation, to obtain a single nonlinear partial differential equation that retains a nonlinear convection term similar to the Navier–Stokes equations. Using the VQC approach, Lubasch and co-workers recently published example quantum circuits to model the Burgers equation [24]. Griffin et al. [7] discuss two approaches for treating the nonlinear term in the Navier–Stokes equations: the VCQ approach of Lubasch et al. [24] and a linearized approach. These authors conclude that, at present, VQC represents the most promising approach for Navier–Stokes equations. Their study also highlights that much further research work is needed to create efficient algorithms for fluid dynamics applications. It is relatively easy to show that the linearization approach to solving non-linear governing equations on a Quantum Computer is generally unfeasible. In applying linearization to nonlinear governing equations, the idea is to use a linearization about the present state of the solution, and then advance this linearized problem in time. This creates a linearization error, which is small if the time step is small. However, even if this linearization error can be tolerated, the linearization approach is problematic in a quantum computing context. This is due to the need for repeated measuring of the quantum state (so that the gates that implement the linear operator may be updated with the current solution) and repeated re-initialization of the quantum state. The complexity associated with repeated measuring and re-initialization is so large that any benefit of a quantum algorithm over a classical algorithm is very likely to vanish.

The development of quantum algorithms for fluid dynamics is clearly at a very early stage and therefore it is essential that different approaches are considered.

5. Representing nonlinear terms in computational basis

In the present work, an alternative approach to introducing the nonlinear terms of nonlinear differential equations into a quantum algorithm is investigated. Specifically, the assumption is made that in a large-scale quantum algorithm for the solution of the nonlinear (partial) differential equations, the solution is encoded in terms of amplitude in the quantum state vector, i.e. the approach used in a wide range of algorithms including the QFT. Then, for the nonlinear terms of the equations, the following steps are suggested. First, within the larger quantum algorithm, a quantum algorithm is embedded that converts the solution from the quantum-amplitude representation to a representation in the computational basis. Recently, quantum algorithms for this 'analog-to-digital conversion' were published by Mitarai et al. [25]. Using the representation of the solution in the computational basis, the required nonlinear terms are then efficiently evaluated using quantum circuits presented later in this chapter. Once computed, a conversion back to quantum-amplitude representation is to be used, enabling the rest of the quantum algorithm to proceed. For this 'digital-to-analog' conversion, quantum algorithms were recently studied and published by SaiToh [26]. For the representation in the computational basis, a fixed-point approach is typically employed to represent real or complex numbers in quantum algorithms. The number of additional qubits required when using computational-basis encoding depends directly on the number of qubits required to represent the real and complex numbers needed in the algorithm. In the present work, a different approach is put forward: instead of using fixed-point arithmetic, a floating-point representation is used.

In the literature, quantum arithmetic using floating-point numbers has received very little attention so far. Haener et al. [27] described an investigation into quantum circuits for floating-point addition and multiplications and compared automatically generated circuits from Verilog implementations with hand-crafted optimized circuits. Their study provides evidence that floating-point arithmetic is a viable candidate for use in quantum computing, at least for typical scientific applications, where addition operations usually do not dominate the computation. Following on from these conclusions, the present work investigates the use of floating-point arithmetic as part of evaluating nonlinear terms in the computational basis.

5.1 Previous works on algorithms in computational basis

Quantum arithmetic in the computational basis constitutes an important component of many quantum algorithms, and as a result reversible implementations of algebraic functions (addition, multiplication, inverse, square root, etc.) have been widely studied. In contrast, there is relatively little work on quantum algorithm implementation of higher-level transcendental functions, such as logarithmic, exponential, trigonometric and inverse trigonometric functions. Examples of applications of trigonometric and inverse trigonometric functions in the computational basis can be found in the famous HHL algorithm [9] and in the state preparation algorithm introduced by Grover and Rudolph [28]. More recently, a quantum

algorithm for approximating the QR decomposition of a $N \times N$ matrix in the computational basis was published by Ma et al. [29], with polynomial speed-up over the best classical algorithm.

5.2 Fixed-point and floating-point arithmetic

A fixed-point number held in an n_q qubit register can be defined as the following quantum state,

$$|w\rangle = \overbrace{\left|w^{(n_{int}-1)}\right\rangle \otimes \left|w^{(n_{int}-2)}\right\rangle \otimes \ldots \otimes \left|w^{(0)}\right\rangle}^{integer} \otimes \overbrace{\left|w^{(-1)}\right\rangle \otimes \ldots \otimes \left|w^{(n_{int}-n_q)}\right\rangle}^{fractional} \quad (5)$$

where $w^{(j)} \in 0, 1, j = n_{int} - n_q, n_{int} - n_q + 1, \ldots, 0, \ldots, n_{int-1}$ [30]. This state represents the number $w = \sum_j w^{(j)} 2^j$. The n_{int} leftmost qubits are used to represent the integer part of the number and the remaining $n_{frac} = n_q - n_{int}$ qubits represent its fractional part. In this example, no sign qubit is used so that only positive numbers can be represented (for most applications an additional sign qubit would be required). Since fewer than n_q bits may suffice for the representation of the input, a number of the leftmost qubits in the register may be set to $|0\rangle$. Clearly, the fixed point system is very limited in terms of the size of the numbers it can store. Therefore, soon after computers were introduced for numerical computing the switch to floating-point arithmetic was made. In a computer implementation of a floating point number with base 2, a non-zero signed number x, defined through a normalized representation, is expressed as,

$$x = \pm S \times 2^E, \quad \text{where} \quad 1 \leq S < 2 \quad (6)$$

where the numbers S and E are the mantissa and the exponent, respectively. The binary expansion of the mantissa is

$$S = (b_0.b_1b_2b_3 \ldots)_2 \quad \text{with} \quad b_0 = 1 \quad (7)$$

Here, it is important to note that always $b_0 = 1$ for non-zero numbers in a normalized representation. This will be used in the present work to achieve savings in the number of required qubits, as detailed later. In the binary representation, the bits following the binary point are the fractional part of the mantissa. Once floating-point numerical computation on classical computers became commonplace, the industry standard IEEE 754 was introduced [31]. A similar standard for floating-point representations on a quantum computer does not yet exist, but is desirable [30]. A key feature of the IEEE standard is that it requires correctly rounded operations: correctly rounded arithmetic operations, correctly rounded remainder and square root operations and correctly rounded format conversions. Typically, rounding to the nearest floating pointing number available in the destination (output register) is used. In the quantum circuits in the present work, rounding down to nearest is used, for reasons of simplicity. Detailed analysis of quantum-circuits developed here for squaring and multiplication operations shows that 'correctly' rounding to nearest involves a significant increase in circuit complexity (i.e. using quantum equivalents of guard and sticky bits, that are well established in arithmetic

on classical computers [31]). A key aspect of the IEEE 754 that has been incorporated in the present work is the definition of *sub-normal numbers*. To illustrate the concept of subnormal numbers, the IEEE 754 standard representation of single format numbers using a 32-bit word is considered. The first bit is the sign bit, followed by 8 bits representing the exponent. Then, 23 bits are used to store a 24-bit representation of the mantissa, i.e. b_0 is not stored. Numbers with exponent bits $(00000000)_2 = (0)_{10}$ and $(11111111)_2 = (255)_{10}$ are defined special cases. The smallest normalized number is $(1.0 \ldots 0)_2 \times 2^{-126} = 2^{-126}$. Sub-normal numbers are used to represent smaller numbers, i.e. in this case the exponent field has a zero bit string but the fraction field has a nonzero bit string. Zero is represented with a zero bit string for the fractional field. For all subnormal numbers, the (00000000) used for the exponent represents 2^{-126} and by using the 23 fractional field bits, equally-spaced numbers in the range $(0.00 \ldots 01)_2 \times 2^{-126}$ (with 22 zero bits after the binary point) to $(0.11 \ldots 11)_2 \times 2^{-126}$ (with 23 one bits after the binary point) are encoded.

5.3 Quantum floating-point format used in present work

Based on the floating point representation defined in the IEEE standard, the present work introduces a floating-point system with fewer bits (i.e. qubits in this case) than the 32 used for single format numbers. This is the direct result of the limited number of qubits available on current and near-term quantum computers. To optimize the range of floating-point numbers that can be represented with the approach used here, the following key aspects of the IEEE standard were adopted:

- For the mantissa only the fractional part is stored,
- Exponent bit strings $(00 \ldots 00)_2$ and $(11 \ldots 11)_2$ are used for special cases, i.e. dealing with 0, subnormal numbers as well as cases with overflow,
- The remaining range of exponent bit strings is used for a range of exponential centred around $2^0 = (01 \ldots 11)_2$,
- Sub-normal numbers are used to extend the range of small numbers,
- Rounding down to nearest is used as rounding mode,
- Only unsigned numbers are considered for simplicity. Signed numbers can easily be obtained by adding a further 'sign' qubit.

In this work, a floating-point number is represented as an $n_q = N_M + N_E$ quantum register. In the quantum-circuit implementation, the most significant (left-most) mantissa qubit is not stored, using the hidden-bit approach used in the IEEE 754 standard. Therefore, $N_M - 1$ qubits define the fractional part of the mantissa in the developed quantum circuits. N_E defines the number of qubits used to define the exponent. In the following, examples with $N_E = 3$ and $N_E = 4$ and $N_M \in [3, 5]$ are considered. For $N_E = 3$, the number 1.00 is defined by $|00|011\rangle$ when $N_M = 3$. Similarly, $|000|0111\rangle$ defines the number 1.000 for $N_M = 4$ and $N_E = 4$. For $N_E = 3$, the smallest normalized number that can be represented is 1/4 independent of the

$N_M = 3$	$N_M = 4$	$N_M = 5$
$\|011\|000\rangle = 3/16$	$\|0111\|000\rangle = 7/32$	$\|01111\|000\rangle = 15/64$
$\|010\|000\rangle = 1/8$	$\|0110\|000\rangle = 3/16$	$\|01110\|000\rangle = 7/32$
$\|001\|000\rangle = 1/16$	$\|0101\|000\rangle = 5/32$	$\|01101\|000\rangle = 13/64$
	$\|0100\|000\rangle = 1/8$	$\|01100\|000\rangle = 3/16$
	$\|0011\|000\rangle = 3/32$	$\|01011\|000\rangle = 11/64$
	$\|0010\|000\rangle = 1/16$	$\|01010\|000\rangle = 5/32$
	$\|0001\|000\rangle = 1/32$	$\|01001\|000\rangle = 9/64$
		⋮
		$\|00010\|000\rangle = 1/32$
		$\|00001\|000\rangle = 1/64$

Table 1.
Sub-normal numbers for floating-point numbers with 3 *qubits as exponential.*

$N_M = 4$	$N_M = 5$
$\|0111\|0000\rangle = 7/512$	$\|01111\|0000\rangle = 15/1024$
$\|0110\|0000\rangle = 3/256$	$\|01110\|0000\rangle = 7/512$
$\|0101\|0000\rangle = 5/512$	$\|01101\|0000\rangle = 13/1024$
$\|0100\|0000\rangle = 1/128$	$\|01100\|0000\rangle = 3/256$
$\|0011\|0000\rangle = 3/512$	$\|01011\|0000\rangle = 11/1024$
$\|0010\|0000\rangle = 1/256$	$\|01010\|0000\rangle = 5/512$
$\|0001\|0000\rangle = 1/512$	$\|01001\|0000\rangle = 9/1024$
	⋮
	$\|00010\|0000\rangle = 1/512$
	$\|00001\|0000\rangle = 1/1024$

Table 2.
Sub-normal numbers for floating-point numbers with 4 *qubits as exponential.*

number mantissa qubits. Then, exponent state $|000\rangle$ defines zero and sub-normal numbers, as shown in **Table 1** for $N_M = 3$, $N_M = 4$ and $N_M = 5$.

Similarly, using 4 qubits for the exponent ($N_E = 4$) means that the smallest normalized number is 1/64. For $N_M = 4$ and $N_M = 5$, **Table 2** shows the corresponding sub-normal numbers.

In line with the IEEE 754 standard, exponent state $|1 \ldots 1\rangle$ denotes numbers for which an overflow has occurred. For $N_E = 3$, the largest normalized number available is $|11 \ldots 1|110\rangle$ which equates to 14 and 15 for $N_M = 3$ and $N_M = 4$, respectively. Similarly, for $N_E = 4$, the largest normalized number available is $|11 \ldots 1|1110\rangle$ which equates to 240 and 248 for $N_M = 4$ and $N_M = 5$, respectively.

6. Quantum circuits for squaring floating-point numbers

For a floating-point number defined by N_M mantissa and N_E exponent bits, a total of $N_M - 1 + N_E$ qubits is needed to define the state in the quantum circuits

introduced here. An example with $N_M = 3$ and $N_E = 3$ will now considered, using registers $|imb1|imb0\rangle$ and $|ieb2|ieb1|ieb1\rangle$ to define the fractional part of the mantissa and the exponent of the input number, respectively. For the multiplication operation described later a second input floating-point number is defined using $|ima1|ima0\rangle$ and $|iea2|iea1|iea1\rangle$. The output of the squaring and multiplication operations is a floating-point number r defined by $|imr1|imr0\rangle$ and $|ier2|ier1|ier1\rangle$ (initialized at $|\mathbf{0}\rangle$). In addition to the input and output registers, the quantum circuits will need additional qubits to hold results of intermediate results, e.g. for $N_M = 3$ a 6-qubit sub-register $|imp5| \ldots |imp0\rangle$ is used. To facilitate the quantum-multiplication operations, a further ancilla qubit $|a0\rangle$ is used. For quantum circuits without measures to deal with sub-normal numbers and overflow, the quantum state for $N_M = 3$ and $N_E = 3$ is defined in a $2 \times (N_M - 1 + N_E) + 2 \times N_M + 1 = 17$-qubit register

$$|ieb2|ieb1|ieb0|imb1|imb0|a0|imp5| \ldots |imp0|ier2|ier1|ier0|imr1|imr0\rangle \tag{8}$$

For $N_M = 4$ and $N_E = 4$, the required number of qubits increases to $2 \times (N_M - 1 + N_E) + 2 \times N_M + 1 = 23$. The quantum circuit performing the squaring operation for $N_M = 3$ and $N_E = 3$ is detailed here as example (in realistic applications $N_M > 3$ will typically be needed). **Figure 2** shows the quantum circuit used in the first step of computing the square of a quantum floating point with $N_M = 3$ and $N_E = 3$. This step involves computing the square of the mantissa, with this result temporarily stored in $|imp5| \ldots |imp0\rangle$. In this circuit, $QFT6$ prepares this temporary register for the three subsequent product steps denoted by P_1, P_2 and $P3$, involving doubly-controlled phase operations. Specifically, three-qubit gates are used applying a phase rotation conditional on state of $|a0\rangle$ and either $|imb1\rangle$ or $|imb0\rangle$. The P_i steps are controlled-summation operations in the shift-and-add approach to computing the products, i.e. the circuits in P_i are derived from quantum adders

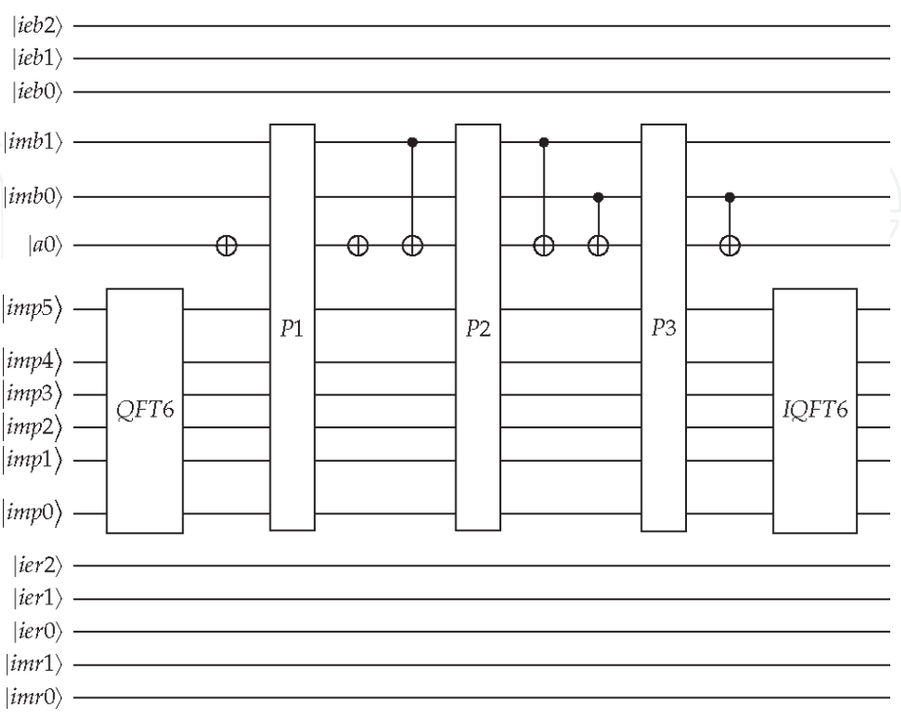


Figure 2.
Quantum circuit used to compute square of mantissa (for $N_M = 3$ and $N_E = 3$).

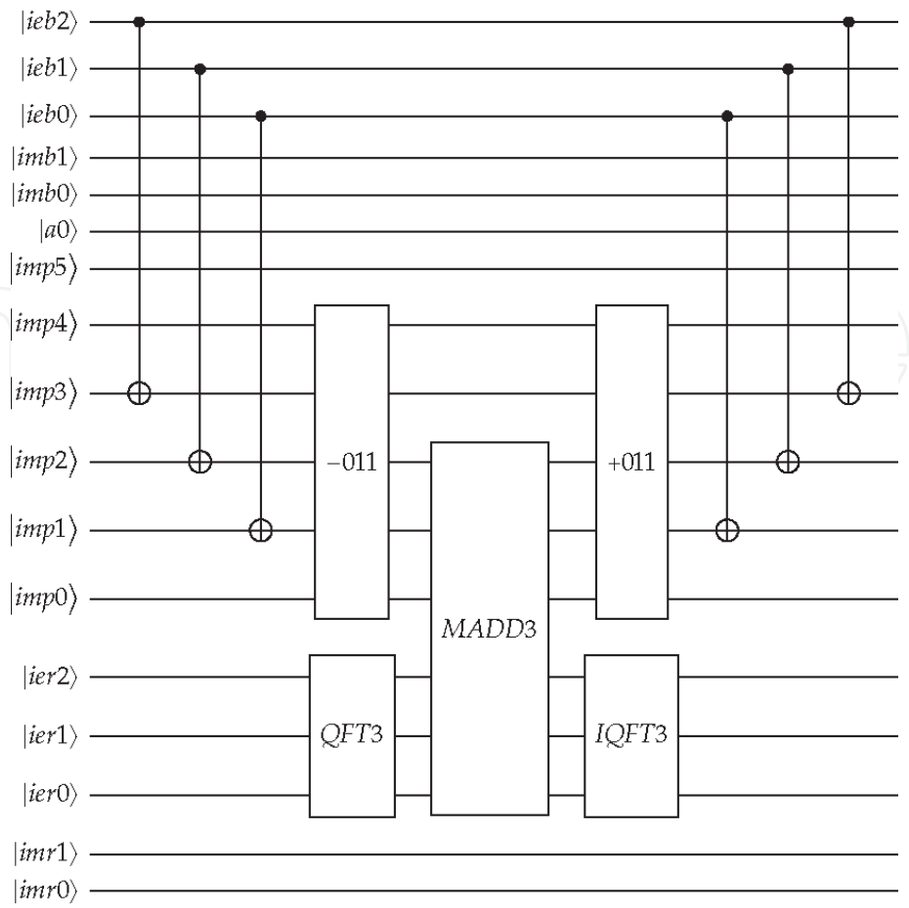


Figure 3.
Quantum circuit used to obtain exponent for squaring operation ($N_M = 3$ and $N_E = 3$).

controlled by an additional qubit. Once the controlled phase changes in the circuits P_1, P_2 and P_3 have been applied, inverse $QFT6$ on $|imp5|\ldots|imp0\rangle$ creates the desired output state. In case the square of the mantissa ≥ 2, i.e. $|imp5 = 1\rangle$, the result exponent needs to be incremented by 1. This is achieved by apply a controlled-NOT to $|ier0\rangle$ (which was initialized at $|0\rangle$) with $|imp5\rangle$ as control. In the next step, result mantissa qubits $|imr1|imr0\rangle$ are set using temporary results in $|imp5|\ldots|imp0\rangle$, where the required gate operations are conditional on the state of $|imp5\rangle$. Then the steps shown in **Figure 2** are 'uncomputed' so that the sub-register $|imp5|\ldots|imp0\rangle$ is set to $|\mathbf{0}\rangle$ again. The next step is illustrated in **Figure 3**, where the output exponent is obtained. This step involves the initialization of the temporary register $imp3|\ldots|imp0$ with $2 \times E_b$ (i.e. twice the input exponent). Then, the bias of $(011)_2 = 3$ is removed (denoted by -011). This bias removal uses two's complement to create a modified modulo-5 adder that removes a value $(011)_2 = 3$ from $|imp4|\ldots|imp0\rangle$. Then, the result exponent sub-register $|ier2|ier1|ier0\rangle$ is prepared for the subsequent modulo-3 addition (denoted by $MADD3$) by applying $QFT3$. Next, the modulo-3 adder is used to add the qubits $|imp2|imp1|imp0\rangle$ into $|ier2|ier1|ier0\rangle$. By applying the inverse QFT3 on $|ier2|ier1|ier0\rangle$ the required state is obtained. The remaining steps shown in the quantum circuit in **Figure 3** are used to 'uncompute' and clean-up the temporary register, e.g. using inverse $QFT3$ and a modified modulo-5 adder to re-apply the bias $(011)_2 = 3$. The circuits described so far do not take into account the special situation arising from creating sub-normal numbers as output as well as cases with 'overflow' results. This is discussed next.

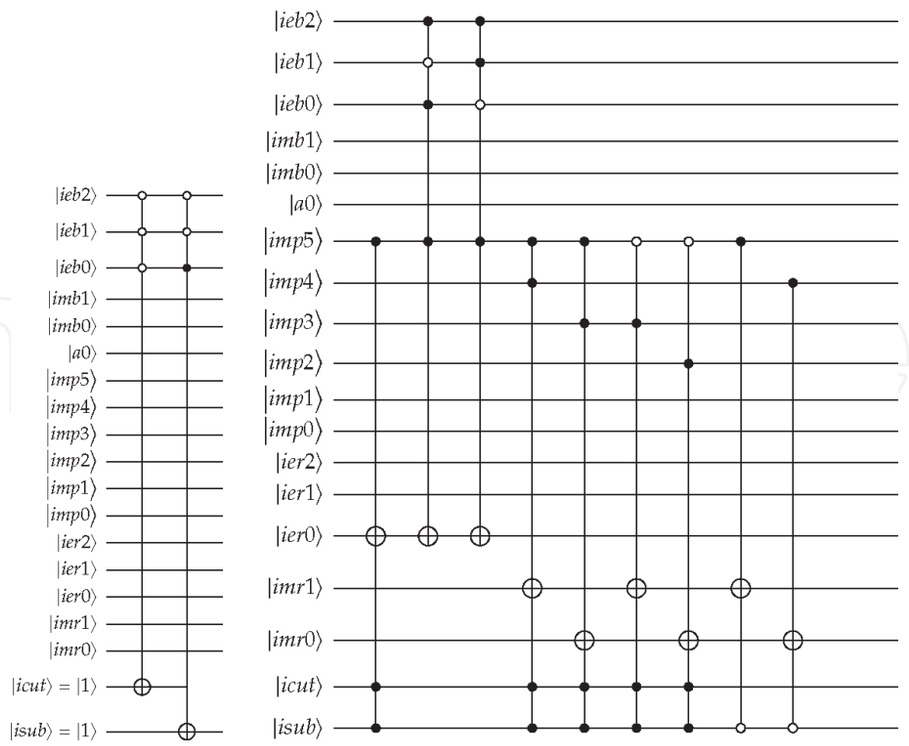


Figure 4.
Quantum circuits used in obtaining output mantissa for squaring operation, including sub-normal numbers and underflow/overflow protection ($N_M = 3$ and $N_E = 3$).

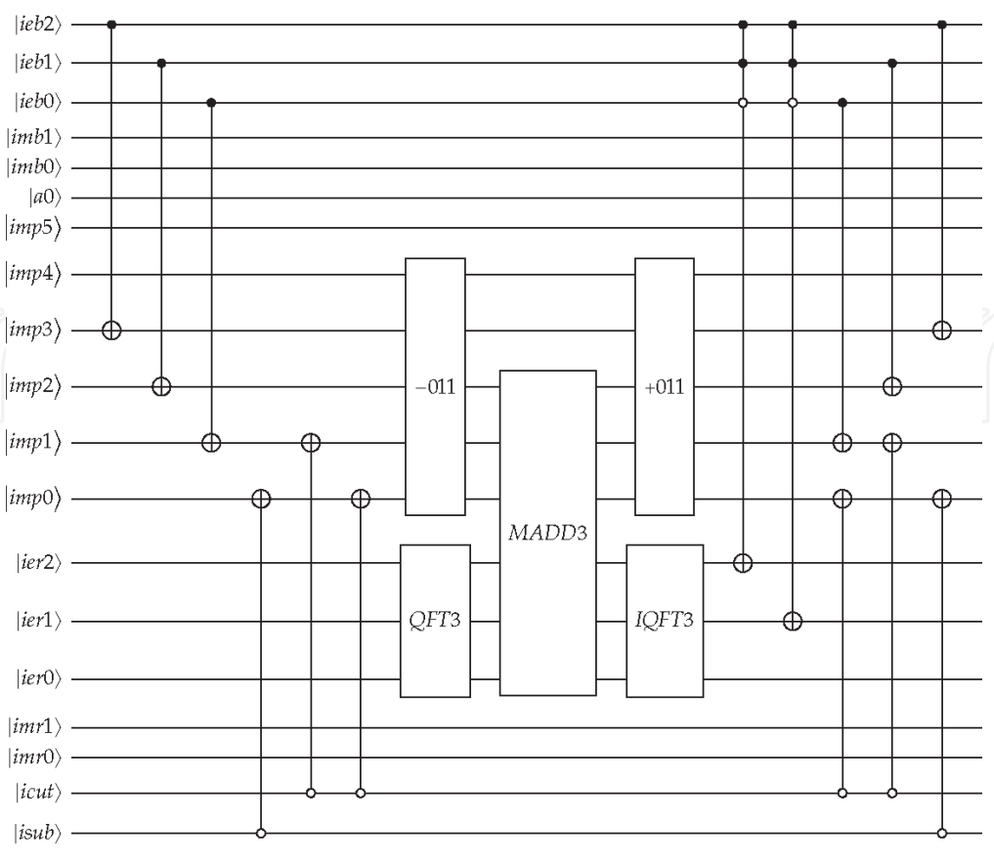


Figure 5.
Quantum circuit used to obtain exponent for squaring operation, including sub-normal numbers and under/overflow protection ($N_M = 3$ and $N_E = 3$).

For certain normalized input numbers the squaring operation leads to outputs truncated to 0 or to the non-zero sub-normal numbers discussed in Section 5.3. The quantum circuits discussed so far need to be modified in a number of ways to deal with this possible sub-normal output. **Figure 4** illustrates the required changes for $N_M = 3$ and $N_E = 3$. Two additional qubits are needed. Qubit $|isub\rangle = |0\rangle$ is used as indication that result is a sub-normal number. Qubit $|icut\rangle = |0\rangle$ is similarly used to define cases with output truncated to 0. Both qubits are initialized to $|1\rangle$. Then, before the mantissa multiplication step takes place, a first modification is introduced, shown on the left-hand side of **Figure 4**. For $N_E = 3$, only inputs with exponent $|000\rangle$ will need truncating to 0, as shown in the first 4-qubit controlled-NOT gate flipping $|icut\rangle$ to $|0\rangle$. For $N_E = 3$, inputs with exponent $|001\rangle$ are guaranteed to lead to sub-normal output (or 0), and for these cases $|isub\rangle$ is set to $|0\rangle$, using the second 4-qubit controlled-NOT gate with $|isub\rangle$ as target. The mantissa-multiplication step shown in **Figure 2** remains unchanged (i.e. qubits $|isub\rangle$ and $|icut\rangle$ are not used). The next required modification relates to the 'copying' of the result of the mantissa multiplication to output register $|imr1|imr0\rangle$ and the application of increments to the output exponent. The additional logic needed is

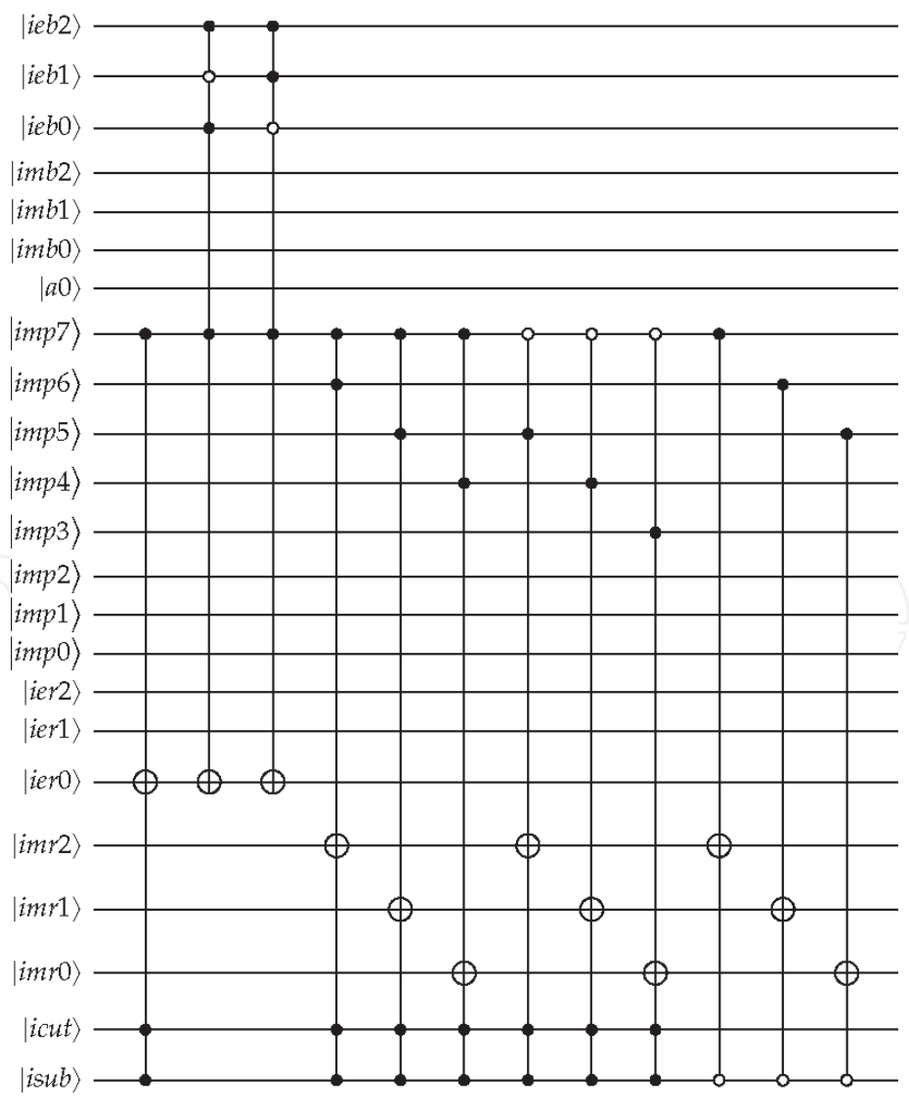


Figure 6.
Quantum circuit used to set output mantissa for squaring operation, including sub-normal numbers and underflow/overflow protection ($N_M = 4$ and $N_E = 3$).

shown on the right-hand side of **Figure 3**. First, for $|imp5\rangle = |1\rangle$, setting $|ier0\rangle = |1\rangle$ becomes conditional of both $|isub\rangle = |1\rangle$ and $|icut\rangle = |1\rangle$. The next two 4-qubit gates are used to guarantee that correct output with exponent $|111\rangle$ is created for inputs with exponents $|101\rangle$ and $|110\rangle$. The remaining gate operations perform the 'copying' of the mantissa squared into $|imr1|imr0\rangle$ taking into account the possible sub-normal output (cases with $|isub\rangle = |0\rangle$). The steps for $|isub\rangle = |1\rangle$ are the same as in the corresponding circuit for squaring without the sub-normal number modifications. A further set of circuit modifications to deal with sub-normal numbers is required in the quantum circuit used to obtain the output exponent. **Figure 5** shows the additional operations required relative to the original quantum circuit shown in **Figure 3**. Three additional CNOT operations are introduced just before performing the *QFT*3. For $|isub\rangle = |0\rangle$ and $|icut\rangle = |0\rangle$ the initialization of $|imp1|imp0\rangle$ is modified so that the subsequent steps will produce the correct result for the exponent. The three CNOT operations also appear in the 'uncompute' stage at the right-hand side of the circuit. Further changes comprise two 4-qubit controlled-NOT operations on $|ier2\rangle$ and $|ier1\rangle$ required to create $|111\rangle$ exponents for inputs with exponent $|110\rangle$.

For a fixed value of N_E it is important to note that the additional complexity introduced by increasing N_M is limited. In fact, the quantum circuit shown on the left-hand side of **Figure 4** does not depend on N_M. Similarly, the quantum circuits used to obtain the result exponent are independent of N_M. The circuit shown on the right-hand side of **Figure 4**, representing the definition of $|imr1|imr0\rangle$ for cases with normalized or sub-normal output requires modification. **Figure 6** shows how $|imr2|imr1|imr0\rangle$ are set for $N_M = 4$ using a set of gate operations that has grown

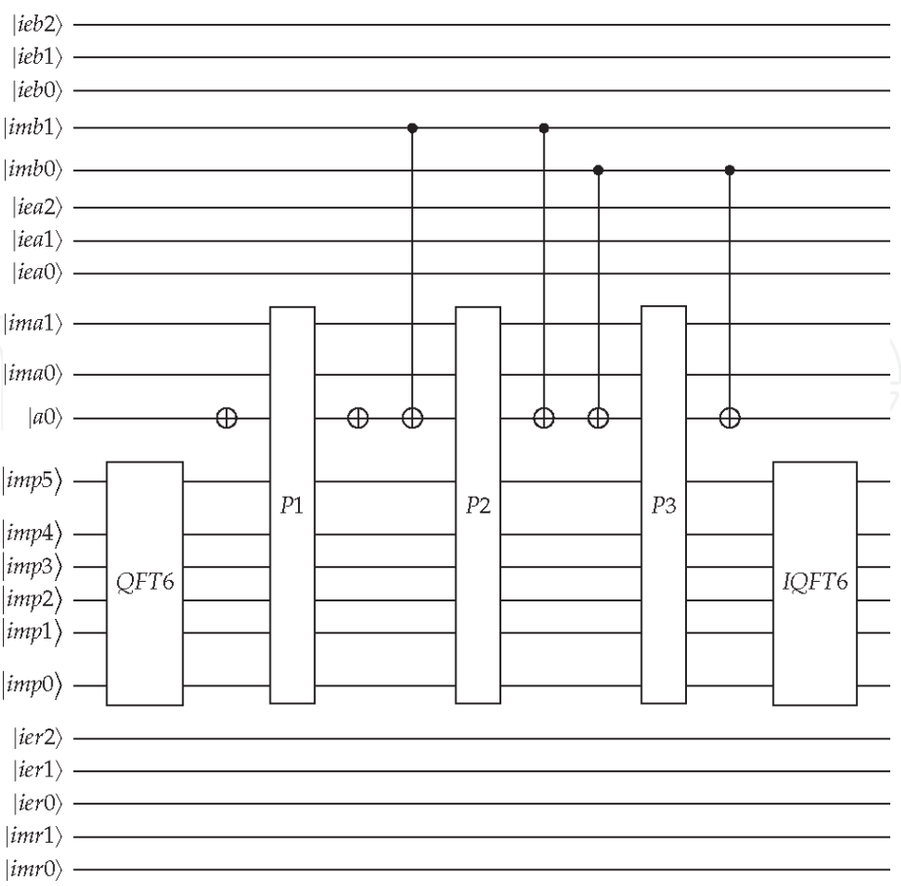


Figure 7.
Quantum circuit used in multiplying the mantissa of two input numbers ($N_M = 3$ and $N_E = 3$).

linearly with N_M. The circuit shown accounts for sub-normal numbers and includes underflow/overflow protection.

7. Quantum circuits for multiplication of floating-point numbers

In the interest of brevity, only the main features of the quantum circuits used for multiplication of two quantum floating-point numbers are summarized here. **Figure 7** illustrates the quantum circuit used to compute the product of the mantissas of two inputs. Compared to the circuit shown in **Figure 2** the main difference is that ancilla qubit $|a0\rangle$ is now set using the mantissa of a second input. A further difference relative to the squaring operation occurs in the circuit used to obtain the result exponent. Here, instead of setting $2\times$ the exponent using a bit shift, the sum of the two input exponents needs to be computed employing a quantum full adder.

8. Results of simulation and verification of quantum circuits

The proposed quantum circuits for squaring and multiplying floating-point numbers as part of the computational-basis representation, were systematically verified by gate-level simulation of the circuits for a wide range of cases with and without sub-normal numbers as well as cases with overflow results. The C++ quantum computer simulator detailed in previous work [4] was used for this purpose. To illustrate the process, the quantum algorithm used to square numbers with $N_M = 3$ and $N_E = 3$ is considered, with the following 19-qubit register (algorithm demonstrated accounts for sub-normal numbers as well as underflow/overflow protection, see Eq. (8) for reference):

$$|ieb2|ieb1|ieb0|imb1|imb0|a0|imp5| \ldots |imp0|ier2|ier1|ier0|imr1|imr0|icut|isub\rangle \quad (9)$$

where $|ieb2|ieb1|ieb0\rangle$ and $|imb1|imb0\rangle$ define the exponent and the fractional part of the mantissa of the input, respectively. Qubits $|icut\rangle$ and $|isub\rangle$ are initialized as $|1\rangle$, while all other qubits are initialized as $|0\rangle$. The quantum state in the simulation is then initialized with a single non-zero (unit) amplitude, with the index in the quantum state vector defined by the binary representation of input exponent and fractional part of mantissa. With the rounding mode fixed at rounding down to nearest, the intended output can be easily computed before the quantum circuit is simulated. In effect, this defines the index of the single non-zero (unit) amplitude of the output quantum state that should be returned in case the circuit is correct. Upon finalizing the quantum computer simulation the actual quantum state vector obtained is compared against the previously-computed required output. For this verification to be meaningful, the following range of possible inputs and outputs

Input	Initial state	Output state
7/2 (i)	$\psi_{init}\left[(1001100000000000011)_2\right] = 1$	$\psi_{out}\left[(1001100000001101011)_2\right] = 1$
7/16 (ii)	$\psi_{init}\left[(0011100000000000011)_2\right] = 1$	$\psi_{out}\left[(0011100000000001110)_2\right] = 1$
3/16 (iii)	$\psi_{init}\left[(0001100000000000011)_2\right] = 1$	$\psi_{out}\left[(0001100000000000001)_2\right] = 1$
6 (iv)	$\psi_{init}\left[(1011000000000000011)_2\right] = 1$	$\psi_{out}\left[(1011000000001110011)_2\right] = 1$

Table 3.
Results from quantum circuit simulation for representative range of inputs (squaring $N_M = 3$, $N_E = 3$).

were considered: (i) input and output are both normalized numbers, (ii) input is normalized number and output is a sub-normal number, (iii) input is a sub-normal number and result truncated to 0, (iv) input is a normalized number, with output overflow. For $N_M = 3$ and $N_E = 3$, **Table 3** summarizes the input and output states for examples of each of the 4 categories considered. For inital and output the single non-zero amplitudes are shown. Since the simulator employed here stores the full 2^{n_q} state vector for n_q qubits, only circuits with ≤ 28 qubits were considered as a result of limited computational resources and the large number of cases considered (>100). For the squaring operation, $N_M \in [3, 6]$ and $N_E \in [3, 4]$ were considered, while for multiplication the range of N_M needed to be reduced, i.e. $N_M \in [3, 4]$.

N_M	N_E	$L_2(u)$	$L_\infty(u)$	$L_2(p)$	$L_\infty(p)$
Rounding down - using sub-normal numbers					
3	3	26.805	0.124741	13.2908	0.0623342
4	3	7.69964	0.0624983	3.79396	0.0310842
5	3	1.93069	0.0312483	0.883095	0.0154592
6	3	0.477862	0.0156233	0.233542	0.00780768
7	3	0.110358	0.00781078	0.0611784	0.00390143
8	3	0.0247615	0.00390453	0.0135501	0.00194831
4	4	6.36002	0.0624983	1.57508	0.0310387
5	4	1.62679	0.0312483	0.387261	0.0154137
6	4	0.409663	0.0156233	0.10847	0.00762945
7	4	0.0958982	0.00781078	0.0296086	0.0037232
8	4	0.0209894	0.00390453	0.00647854	0.00192175
Rounding down - without sub-normal numbers					
3	3	86.8625	0.248583	111.896	0.249507
4	3	70.4413	0.248583	108.352	0.249507
5	3	65.8235	0.248583	107.359	0.249507
6	3	64.6349	0.248583	107.135	0.249507
7	3	64.3262	0.248583	107.069	0.249507
8	3	64.2529	0.248583	107.050	0.249507
4	4	6.3881	0.0624983	1.6114	0.0310387
5	4	1.65503	0.0312483	0.42405	0.0154137
6	4	0.437976	0.0156233	0.14534	0.0151248
7	4	0.124223	0.0147218	0.0665074	0.0151248
8	4	0.0493163	0.0147218	0.0433827	0.0151248

Table 4.
Approximation errors in Taylor-green vortex flow field due to reduced-precision floating-point representation. L_∞ and L_2 norms of errors relative to IEEE double-precision representation for velocity (u) and pressure (p) for different N_M and N_E. 100 × 100 uniform mesh.

9. Complexity analysis

Before analyzing the quantum circuits introduced here in terms of complexity, first the choice of N_M and N_E for representing realistic flow fields is considered.

9.1 Representing Taylor-green vortex flow

In a two-dimensional flow field, the non-linear terms appearing in the Navier–Stokes equations, shown in Eq. (4), involve the square of the velocity components in $x-$ and $-y$ directions, i.e. u^2 and v^2, as well as, the product uv. Here, the example flow field defined by the two-dimensional Taylor-Green vortex is considered, where velocity and pressure are defined in a square domain $[0, 2\pi]^2$ with periodic boundary conditions as,

$$u = cos\,(x)\,sin\,(y) \;\; ; \quad v = -sin\,(x)\,cos\,(y) \;\; ; \quad p = -\frac{1}{4}[cos\,(2x) + cos\,(2y)] \quad (10)$$

Considering a 100×100 uniform mesh, the effect of representing the flow field variables with a reduced-precision floating-point format is analyzed first.

N_M	N_E	$L_2(u^2)$	$L_\infty(u^2)$	$L_2(\|uv\|)$	$L_\infty(\|uv\|)$
Rounding down - using sub-normal numbers					
4	4	0.801596	0.0351562	0.161925	0.0146484
5	4	0.3848	0.0244141	0.0520772	0.00732422
6	4	0.101035	0.013916	0.018016	0.00378418
7	4	0.0382158	0.00738525	0.0053449	0.00186157
8	4	0.0108621	0.00379944	0.00123537	0.000919342
Rounding down - without sub-normal numbers					
4	4	0.87222	0.0351562	0.30511	0.0147705
5	4	0.461689	0.0244141	0.213756	0.0153809
6	4	0.18035	0.0151405	0.188371	0.0154495
7	4	0.119222	0.0151405	0.179671	0.0154495
8	4	0.0927176	0.0152609	0.177551	0.015553

Table 5.
Approximation errors of velocity products in Taylor-green vortex flow field due to reduced-precision floating-point representation. L_∞ and L_2 norms of errors relative to IEEE double-precision representation for velocity (u^2) and pressure (uv) for different N_M and N_E. 100×100 uniform mesh.

	CPHASE	C^2PHASE	$\theta_{\min}$
3×3	9	27	$2\pi/2^6$
4×4	14	66	$2\pi/2^8$
5×5	20	130	$2\pi/2^{10}$

Table 6.
Number of controlled-phase gates (CPHASE) and doubly-controlled-phase (C^2PHASE) for phase-addition operator in quantum-multiplier. Also, smallest rotation angle is shown.

	CPHASE	$\theta_{\min}$		CPHASE	$\theta_{\min}$
*MADD*3	6	$2\pi/2^3$	*FADD*3	9	$2\pi/2^4$
*MADD*4	10	$2\pi/2^4$	*FADD*4	14	$2\pi/2^5$
*MADD*5	15	$2\pi/2^5$	*FADD*5	20	$2\pi/2^6$
*MADD*6	21	$2\pi/2^6$			

Table 7.
Number of controlled-phase gates (CPHASE) in phase-addition step for modulo adder (MADD) and full adder (FADD). Also, smallest rotation angle is shown.

Table 4 summarizes the results, highlighting the importance of including sub-normal numbers in the floating-point representation. Since a sign bit is not used here, the absolute values of u, v, p were actually used. Flow variables defined in Eq. (10) are in the range $[-1, 1]$, so that by increasing N_E from 3 to 4, far fewer sub-normal numbers are used to represent the flow field. As a result, removing the sub-normal number capability (as shown in bottom half of table), results in smaller errors for $N_E = 4$. For realistic applications of the proposed quantum floating point format, the relatively small overhead incurred by introducing sub-normal numbers in the quantum circuits clearly suggests that sub-normal numbers should be included.

For $N_E = 4$, the representation of u^2 and $|uv|$ is considered. Specifically, the error shown is that introduced by the multiplication: the difference between the 'exact' product of the reduced-precision representation of $|u|$ and $|v|$ and the corresponding reduced precision representation of the products is shown in **Table 5**. The results highlight that although sub-normal numbers played a relatively smaller role in representing velocity components, in the computation of the nonlinear terms, the inclusion of sub-normal numbers is more important for the minimization of approximation errors.

9.2 Mantissa multiplication step

QFT and inverse *QFT* are used involving $2N_M$ qubits, so that the complexity in terms of two-qubit (controlled-phase) gates scales as N_M^2, where the well-known complexity of the standard *QFT* implementation is used. The complexity of the phase-addition steps involved in the multiplication are detailed in **Table 6**. For the two-qubit gates the number can be seen to scale as N_M^2, while the number of three-qubit gates shows a N_M^3 scaling.

9.3 Computation of exponent

QFT and inverse *QFT* are used involving N_E, $N_E + 1$ and $N_E + 2$ qubits, representing a smaller complexity than the *QFT* used in mantissa multiplications. The main contributions to complexity of exponent computation stems from the modulo and full-adders involving a number of qubits scaling linearly with N_E. The polynomial complexity in terms of qubits for the adders implemented here is shown in **Table 7**.

9.4 Discussion

The quantum circuits presented here for squaring two floating-point numbers in the format proposed show that by accounting for sub-normal numbers and

under/overflow an additional number of multi-qubit controlled-NOT gates is needed. However, for the examples analyzed a polynomial dependence on N_M and N_E was observed. This means that in terms of quantum-algorithm complexity this implementation has the desired efficiency. The relatively small complexity as compared to circuits used for mantissa multiplication highlights that for most applications it is desirable to include the capability of using sub-normal numbers and provide under/overflow protection in the quantum circuits. The analysis in this section also shows that for a realistic application, a well-considered scaling of the governing equations to $O(1)$ variables is even more important here than in classical implementations using IEEE single- or double-precision arithmetic. Using the limited number of qubits available on current and near-term quantum computers (<100), the proposed approach to introducing non-linearity is a good candidate in cases where N_M and N_E can be chosen significantly smaller than in equivalent classical floating-point representations.

10. Conclusions

The challenges associated with representing non-linear differential equations in terms of quantum circuits were discussed in this chapter. In this work, a new approach for representing product-terms in nonlinear equations suitable for near-term (e.g. NISQ generation) quantum computers was proposed. A key aspect discussed is the (temporary) representation of the variables in the computational basis. Furthermore, the use of a suitably-chosen floating-point format was detailed. The importance of including sub-normal numbers, such as defined in the IEEE 758 standard for floating-point arithmetic on classical computers, was demonstrated. Based on the current findings, a number of suggestions for further work can be put forward. The presented circuits performed arithmetic for a single set of input data, i.e. equivalent to data for a single point in a computational domain. Extending the approach to a multi-dimensional computational mesh is a first step to consider. A complexity analysis will be needed to assess the potential speed-up relative to classical discretization approaches for the considered equations. A further step involves investigating how the proposed approach can be made part of a larger quantum algorithm, where a mix of amplitude-based encoding and computational-basis encoding occurs. A key aspect is therefore the development of efficient quantum circuits to perform the required conversions between the two different encoding approaches. Finally, further work is needed to establish how the approach presented here can be used in a wider range of quantum computing applications.

IntechOpen

IntechOpen

Author details

Rene Steijl
James Watt School of Engineering, University of Glasgow, Glasgow, United Kingdom

*Address all correspondence to: rene.steijl@glasgow.ac.uk

IntechOpen

© 2020 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. CC BY

References

[1] Nielsen MA, Chuang IL. Quantum Computation and Quantum Information: 10th Anniversary Edition. 2nd ed. Cambridge: Cambridge University Press; 2010.

[2] Yepez J. Quantum lattice-gas model for computational fluid dynamics. Phys. Rev. E. 2001;63(4):046702. DOI: 10.1103/PhysRevE.63.046702

[3] Berman GP, Ezhov AA, Kamenov DI, Yepez J. Simulation of the diffusion equation on a type-II quantum computer. Phys. Rev. A. 2002;66(1): 012310. DOI:10.1103/PhysRevA. 66.012310

[4] Steijl R, Barakos GN. Parallel evaluation of quantum algorithms for computational fluid dynamics. Computers&Fluids 2018;173:22-28. DOI: 10.1016/j.compfluid.2018.03.080

[5] Steijl R. Quantum Algorithms for Fluid Simulations. In: Bulnes F, Stavrou VN, Morozov O, Bourdine AV, editors. Advances in Quantum Communication and Information. IntechOpen; 2020. DOI: 10.5772/ intechopen.86685

[6] Todorova BN, Steijl R. Quantum Algorithm for the collisionless Boltzmann equation. J. Comp. Phys. 2020;409:109347. DOI:10.1016/j. jcp.2020.109347

[7] Griffin KP, Jain SS, Flint TJ, Chan WHR. Investigations of quantum algorithms for direct numerical simulation of the Navier-Stokes equations. Center for Turbulence Research Annual Research Briefs. 2019; 347-363.

[8] Gaitan F. Finding flows of a Navier–Stokes fluid through quantum computing. npj Quantum Information. 2020; 6:61. DOI:10.1038/s41534-020-00291-0

[9] Harrow AW, Hassidim A, Lloyd S. Quantum algorithm for linear systems of equations. Phys. Rev. Lett. 2009;103 (15):150502. DOI:10.1103/ PhysRevLett.103.150502

[10] Clader BD, Jacobs BC, Sprouse CR, Preconditioned quantum linear system algorithm. Phys. Rev. Lett. 2013;110 (25):25054. DOI:10.1103/ PhysRevLett.110.250504

[11] Cao Y, Papageorgiou A, Petras I, Traub J, Kais S. Quantum algorithm and circuit design solving the Poisson equation. New J. Phys. 2013;15:013021. DOI:10.1088/1367-2630/15/1/013021

[12] Scherer A, Valiron B, Mau S-C, Alexander S, van den Berg E, Chapuran TE. Concrete resource analysis of the quantum linear-system algorithm used to compute the electromagnetic scattering cross section of a 2D target. Quantum Inf. Proc. 2017;16(3):60. DOI: 10.1007/s11128-016-1495-5

[13] Montanaro A, Pallister S. Quantum Algorithms and the finite element method. Phys. Rev. A. 2016;93(3): 032324. DOI:10.1103/ PhysRevA.93.032324

[14] Xu G, Daley AJ, Givi P, Somma RD. Turbulent mixing simulation via a quantum algorithm. AIAA J. 2018;56(2): 687-699. DOI:10.2514/1.J055896

[15] Berry DW. High-order quantum algorithm for solving linear differential equations. J. Phys. A. 2014;47(10): 105301. DOI:10.1088/1751-8113/47/10/ 105301

[16] Berry DW, Childs AM, Ostrander A, Wang G. Quantum Algorithm for Linear Differential Equations with Exponentially Improved Dependence on Precision. Comm. Math. Phys. 2017;356 (3):1057-1081. DOI:10.1007/ s00220-017-3002-y

[17] Fillion-Gourdeau F, Lorin E. Simple digital quantum algorithm for symmetric first-order linear hyperbolic systems. Numerical Algorithms. 2019; 82:1009-1045. DOI:10.1007/s11075-018-0639-3

[18] Costa PCS, Jordan S, Ostrander *A. quantum* algorithm for simulating the wave equation. Phys. Rev. A. 2019;99 (1):012323. DOI:10.1103/ PhysRevA.99.012323

[19] Childs AM, Liu J-P. Quantum spectral methods for differential equations. Comm. Math. Phys. 2020; 375(2):1427-1457. DOI:10.1007/ s00220-020-03699-z

[20] Leyton SK, Osborne TJ. A quantum algorithm to solve nonlinear differential equations. arXiv.org 2008;0812.4423.

[21] Zhou SS, Loke T, Izaac JA, Wang JB. Quantum Fourier transform in computational basis. Quantum Inf. Proc. 2017; 16(3):82. DOI:10.1007/ s11128-017-1515-0

[22] Peruzzo A, McClean J, Shadbolt P, Yung M-H, Zhou X-Q, Love PJ, Aspuru-Guzil A, O'Brien JL. A variational eigenvalue solver on a photonic quantum processor. Nature Comms 2014; 5:4213. DOI:10.1038/ncomms5213

[23] McClean JR, Romero J, Babbush R, Aspuru-Guzik A. The theory of variational hybrid quantum-classical algorithms. New J. Phys. 2016; 18:023023. DOI:10.1088/1367-2630/18/2/023023

[24] Lubasch M, Joo J, Moinier P, Kiffner M, Jaksch D. Variational quantum algorithms for nonlinear problems. Phys. Rev. A. 2020;101(1): 010301. DOI:10.1103/ PhysRevA.101.010301

[25] Mitarai K, Kitagawa M, Fijii K. Quantum analog-digital conversion. Phys. Rev. A. 2019;99(1):012301. DOI: 10.1103/PhysRevA.99.012301

[26] SaiToh *A. quantum* digital-to-analog conversion algorithm using decoherence. Quantum Inf. Proc. 2015; 14(8):2729-2748. DOI:10.1007/ s11128-015-1033-x

[27] Haener T, Soeken M, Roetteler M, Svore KM. Quantum circuits for floating-point arithmetic. In: Kari J, Ulidowski I, editors. Reversible Computation. RC 2018. Lecture Notes in Computer Science, vol 11106. Springer; 2018. DOI:doi.org/10.1007/978-3-319-99498-7-11

[28] Grover L, Rudolph T. Creating superpositions that correspond to efficiently integrable probability distribution. arXiv 2002;0208112

[29] Ma G, Li H, Zhao J. Quantum QR decomposition in the computational basis. Quantum Inf. Proc. 2019; 19:271. DOI:10.1007/s11128-020-2777-4

[30] Bhaskar MK, Hadfield S, Papageorgiou A, Petras I. Quantum algorithms and circuits for scientific computing. Quantum Info. Comput. 2016;16(3-4):197-236. DOI:10.5555/ 3179448.3179450

[31] Overton M.L. Numerical Computing with IEEE Floating Point Arithmetic. 1st ed. Philadelphia: SIAM; 2001. 97p.