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

Simple digital quantum algorithm for symmetric first order linear hyperbolic systems

F. Fillion-Gourdeau · E. Lorin

the date of receipt and acceptance should be inserted later

Abstract This paper is devoted to the derivation of a digital quantum algorithm for the Cauchy problem for symmetric first order linear hyperbolic systems, thanks to the reservoir technique. The reservoir technique is a method designed to avoid artificial diffusion generated by first order finite volume methods approximating hyperbolic systems of conservation laws. For some class of hyperbolic systems, namely those with constant matrices in several dimensions, we show that the combination of i) the reservoir method and ii) the alternate direction iteration operator splitting approximation, allows for the derivation of algorithms only based on simple unitary transformations, thus perfectly suitable for an implementation on a quantum computer. The same approach can also be adapted to scalar one-dimensional systems with non-constant velocity by combining with a non-uniform mesh. The asymptotic computational complexity for the time evolution is determined and it is demonstrated that the quantum algorithm is more efficient than the classical version. However, in the quantum case, the solution is encoded in probability amplitudes of the quantum register. As a consequence, as with other similar quantum algorithms, a post-processing mechanism has to be used to obtain general properties of the solution because a direct reading cannot be performed as efficiently as the time evolution.

Keywords First order hyperbolic systems, quantum algorithms, quantum information theory, reservoir method.

1 Introduction

Quantum computing is a new paradigm in information science which benefits from quantum mechanics to perform some computational tasks. In the last few decades, it has attracted a lot of attention because it promises efficient solutions to a large class of problems deemed unsolvable on classical computers. Shor's algorithm, for the prime number factorization of integers, is the foremost example of the strength of quantum computing [52]. This algorithm runs in polynomial time, i.e. the computation time scales like a polynomial function of the input size, while the same task runs in (sub-)exponential time on a classical computer. This quantum speedup has motivated the development of many other algorithms for the solution of problems in the BQP complexity class but outside the P class, i.e. problems with bounded error for which the amount of quantum resources is a polynomial function and having an exponential speedup over classical computations [47, 56].

One of the promising applications of quantum computing is the simulation of quantum systems. Inspired from Feynman's quantum simulator [24], it has been demonstrated that *universal quantum*

F. Fillion-Gourdeau
Université du Québec, INRS-Énergie, Matériaux et Télécommunications, Varennes, Canada, J3X 1S2
Institute for Quantum Computing, University of Waterloo, Waterloo, Ontario, Canada, N2L 3G1
E-mail: francois.fillion@emt.inrs.ca

E. Lorin
Centre de Recherches Mathématiques, Université de Montréal, Montréal, Canada, H3T 1J4.
School of Mathematics and Statistics, Carleton University, Ottawa, Canada, K1S 5B6.
E-mail: elorin@math.carleton.ca

computers [22] can simulate efficiently the dynamics of any local quantum Hamiltonian with a number of quantum operations scaling polynomially with the system size [43]. Following these seminal results, other algorithms have been developed for other types of quantum physical systems. Some examples include algorithms for the simulation of non-relativistic single-particle quantum mechanics [64,65,12], relativistic mechanics [27], many-body physics [17,64,65,61,55], quantum field theory [35], quantum chemistry [37, 63,36,8] and many others [18,28].

Generally the simulation of quantum systems on a quantum computer is based on two main ingredients: (i) the encoding of the quantum state on the quantum register, and (ii) the existence of a set of operations that modify a quantum register according to the dynamics of the physical system under study. When both are available, it is possible to use the quantum computer to emulate another quantum physical system.

In contrast to analogue quantum simulators [28], based on a direct analogy between two Hamiltonians and thus restricted to a certain category of systems, the digital quantum computers (DQC) considered in this article are built from a set of entangled two-level quantum systems, the qubits, forming the quantum register. Similar to bits in classical computing, qubits are discrete entities calling for the discretization of the physical system under consideration. However, due to their inherent quantum nature allowing them to be in a superposition of states, the quantum register is characterized by $N = 2^n \in \mathbb{N}^*$ complex coefficients, where n is the number of qubits. Then, a possible strategy consists in mapping the coefficients of the physical system wave function expressed in some suitable basis into the probability amplitudes of the quantum register, as in the aforementioned algorithms. This process is called amplitude encoding. Thus, DQC allows for the simulation of quantum systems in their discretized form, analogously to classical computers.

Furthermore, the dynamics of the quantum computer and the simulated system proceed by unitary operations. In a quantum computer, the state of the quantum register is modified by simple unitary operations: the quantum logic gates [47]. In turn, a quantum system evolves according to unitary operations given by the evolution operator, according to the laws of quantum mechanics. A mapping of the evolution operator onto quantum gates can be implemented via a unitary decomposition, often performed using Trotter(splitting)-like approximations [60]. However, these mappings are not unique, each one corresponding to a different numerical method: the encoding is determined by the basis choice while the chosen unitary decomposition sets the numerical scheme used to evolve the system in time.

Using these two mappings, i.e. the wave function on the quantum register and the evolution operator on quantum gates, it is possible to simulate quantum systems. Moreover, for many cases of interest, these simulations are more efficient than their classical counterparts, in the sense that computing resources are scaling polynomially with the size of the system.

For classical systems, such as fluids, plasmas and electromagnetic fields, the analogy described above between the quantum computer and the physical system is not as explicit and may even be non-existing as the time evolution may be non-unitary. Mathematically, this implies that in contrast to quantum mechanics, the L^2-norm is not always preserved by the dynamics, complicating the mapping on quantum computers which are based on unitary operations. Nevertheless, it is possible to devise algorithms for the quantum simulation of some classical systems. For instance, fluid-like mechanics equations have been considered in [45], but they required the implementation of non-unitary operations. The latter can be implemented on a quantum computer, but the algorithm becomes nondeterministic and the probability of success depends on the time step. For these reasons, algorithms for the simulation of classical systems are scarce, with some notable exceptions where classical thermal state [62], classical diffusion [44], electromagnetism [53] and the Poisson equation [19] have been considered. In addition, there exists algorithms for the solution of ordinary differential equations, which may be relevant for many physical applications [14].

In this article, we consider the quantum simulation of an important class of classical partial differential equations: linear hyperbolic systems. Our approach is similar to a scheme developed for the Dirac equation where an analogy between the split operator method and quantum walks was explicitly constructed [27]. The Dirac equation is in fact a particular hyperbolic system, where the mass and the electromagnetic potential are local source terms. Therefore, it was expected that techniques for the quantum Dirac equation can be adapted to more general hyperbolic systems. This was noticed in [7], where a quantum algorithm for constant linear hyperbolic systems with rational eigenvalues have been investigated. Here, we are proposing a quantum implementation of the reservoir numerical scheme [3,5,40,6,4], extending the results in [7] to more general hyperbolic systems. The reservoir method was developed to avoid

spurious diffusion generated by low order finite volume methods, by adding a reservoir and a Courant-Friedrichs-Lewy (CFL [57]) counter at the interface between discretization volumes. This numerical scheme is then particularly well-suited for a quantum implementation because in some cases, it reduces to simple streaming steps which preserve the L^2-norm and also, can be implemented efficiently on a quantum computer. Notice that the assumption that matrices in the hyperbolic system are symmetric is not indispensable in the reservoir method, but is required in order to have unitary operations. At the same time, it guarantees that the system of equations is hyperbolic.

The main result of this paper, stated precisely in Propositions 51 and 52, is that by combining these ideas and assuming that the quantum register can be initialized in $O(\text{polylog } N)$ operations, it is possible to solve a symmetric hyperbolic system on a DQC with a quantum speedup

$$S = O\left(\frac{m^2}{\log^2 m}\frac{N^d}{\log^2 N}\right),$$

for a large number of grid points N, where d is the number of dimensions and m is the number of characteristic fields. This corresponds to an exponential speedup, i.e. the number of operations in the quantum algorithm increases logarithmically with the number of grid points while it increases linearly in the classical implementation of the same algorithm. This is clearly an interesting advantage of the quantum approach, although there is an important caveat: as the solution is encoded into probability amplitudes, the measurement/reading of the solution on the DQC requires $O(N)$ repetitions of the algorithm, potentially eliminating the exponential speedup. This is a typical problem in the field of quantum simulations (see [14] for instance) which is usually solved partially by post-processing the solution to obtain some observables or some general properties of the solution. To preserve the exponential speedup, one requires that this post-processing is performed using $O(\text{polylog } N)$ operations, which is not possible if one wants the whole solution. Of course, this is an important but standard limitation of the quantum approach.

Although the problem under consideration, with constant symmetric matrices, seems relatively simple from a mathematical point of view, there still exist some complex open problems in physics involving this type of systems. For instance, the advection equation for transport problem, linear acoustic equations for sound waves, Maxwell's equations in electrodynamics, linear elasticity equation and the wave equation are some examples of homogeneous symmetric hyperbolic systems with considerable importance in physics. As mentioned above, the Dirac equation is a first order hyperbolic system with constant coefficients. In quantum electrodynamics, it is well known that accurate computations of electron-positron pair production from Schwinger's effect [26] requires the solution to a given Dirac equation with many initial conditions, which is not tractable on classical computers. Such calculations require fundamental breakthrough from the computing point of view, which could be provided by efficient quantum algorithms. In addition, our work can be considered as a first step towards the development of quantum algorithms for more general hyperbolic equations. We give possible directions to generalize our approach at the end of this article.

This article is organized as follows. In Section 2, we first give some basic notions on quantum computing. In Section 3, we review the reservoir method for linear systems. We then show in Section 4, how to derive a quantum algorithm for the reservoir method for one-dimensional first order hyperbolic systems and then, for multidimensional first order systems. The computational complexity and quantum speedup are discussed in Section 5. In Section 6, we propose some possible avenues to extend the ideas presented in this paper, first to scalar first order hyperbolic equations with non-constant velocity, then using quantum algorithms based on the method of lines. We conclude in Section 7.

2 Survival kit on quantum computing

In this section, we recall some basic definitions in quantum computing, and quantum simulation theory for readers without or with a very limited knowledge of these fields. The objective is to present the mathematical objects and tools which are necessary to construct or understand a quantum algorithm. The reader who already has some advanced knowledge in quantum computing can skip this section. More information on this topic can be found in Ref. [47].

First, we recall that the qubit in QC is the analogue of the bit on digital computers, that is the basic unit of information. More specifically, a qubit is a quantum mechanical two-level system. In principle, it

can be realized by any quantum physical systems having two degrees of freedom, such as single photon polarization, electron spin, superconducting qubits, and many others. In practice, some physical systems are more suitable for quantum computing because they can be controlled more easily and have a larger decoherence time owing to a weaker interaction with the environment.

The state for all two-level systems is described quantum mechanically by a unit vector, denoted here by $|u\rangle$, defined in a two-dimensional Hilbert space $\mathcal{H}$. It can then be written in the form : $|u\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$, where $|0\rangle$, $|1\rangle$ denote the computational basis vectors of $\mathcal{H}$ and α_0, α_1 are complex numbers representing the quantum amplitudes normalized as $|\alpha_0|^2 + |\alpha_1|^2 = 1$.

A quantum register is a set of ℓ two-level entangled systems. In this case, the quantum state of the total system $|u_\ell\rangle$, according to quantum mechanics principles, is a vector in the Hilbert space $\mathcal{H}_\ell = \otimes_{n=1}^{\ell}\mathcal{H}$, the tensor product of ℓ two-dimensional spaces. Then, any $|u_\ell\rangle \in \mathcal{H}_\ell$ reads

$$|u_\ell\rangle = \sum_{s_1=0}^{1} \cdots \sum_{s_\ell=0}^{1} \alpha_{s_1\cdots s_\ell} \otimes_{i=1}^{\ell} |s_i\rangle, \tag{1}$$

where $\alpha_{s_1\cdots s_\ell}$ (for all $s_1 \cdots s_\ell$) are complex numbers representing the coefficients or amplitudes of the quantum state, and $\{|s_i\rangle\}_{1\leqslant i\leqslant \ell}$ are the ℓ qubit basis functions of the ℓ two-dimensional spaces $\{\mathcal{H}_i\}_{1\leqslant i\leqslant \ell}$. We note that $|u_\ell\rangle$ is a unit vector ($\langle u_\ell|u_\ell\rangle = 1$), implying that the amplitudes should be normalized as

$$\sum_{s_1=0}^{1} \cdots \sum_{s_\ell=0}^{1} |\alpha_{s_1\cdots s_\ell}|^2 = 1.$$

This normalization is introduced to have a probability interpretation of the quantum state.

Here, it is important to note that although we only have ℓ qubits, the number of amplitude coefficients is 2^ℓ owing to the tensorial structure of the vector space. Information can be stored into these coefficients. However, reading all of them is challenging because it requires $O(2^\ell)$ measurements. This occurs because each amplitude $\alpha_{s_1\cdots s_\ell}$ is related to the probability of finding the system in some state $\otimes_{i=1}^{\ell}|s_i\rangle$ as $P_{s_1\cdots s_\ell} = |\alpha_{s_1\cdots s_\ell}|^2$. Then, according to the measurement postulate in quantum mechanics, a Von Neumann measurement on a quantum system characterized by the Hilbert space $\mathcal{H}_\ell$ outputs a classical value $s_1 \cdots s_\ell$ with probability $P_{s_1\cdots s_\ell}$. After such measurement, the system has collapsed to the state $\otimes_{i=1}^{\ell}|s_i\rangle$, i.e. any subsequent measurement will obtain $s_1 \cdots s_\ell$ with probability 1. Then, measuring all coefficients $\alpha_{s_1\cdots s_\ell}$ entails reconstructing the probability distribution for all possible states, a process called quantum tomography, which usually requires $O(2^\ell)$ measurements.

Realizing physically a quantum register by entangling a certain number of qubits is a very challenging experimental task. Nevertheless, this has been achieved using several physical systems such as superconducting circuits [39,10,11], trapped ions [41], nuclear magnetic resonance [46] photons [59] and cavity quantum electrodynamics [50]. The typical size for quantum registers varies presently up to ~ 50 qubits, but this number is likely to increase in the future.

2.1 Quantum logic gates

Quantum logic gates are analogues to logical gates in classical computing. More precisely, the quantum gates are operators acting on a quantum register, modifying its state according to the laws of quantum mechanics. Thus, they must be unitary reversible operators. One of the most simple, but also important quantum gate is the *Hadamard gate*. The latter acts on a single qubit and corresponds mathematically to one rotation of π around the x-axis and another rotation of $\pi/2$ around the y-axis, so that

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

Hence, any qubit $|u\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$, is transformed by H as

$$H|u\rangle = \frac{\alpha_0 + \alpha_1}{\sqrt{2}}|0\rangle + \frac{\alpha_0 - \alpha_1}{\sqrt{2}}|1\rangle.$$

Another elementary quantum gate, is the NOT-gate which is the analogue of digital NOT-gate (or inverter), which reads

$$\mathrm{NOT} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

There exists an infinite number of possible $2^\ell \times 2^\ell$-dimensional unitary operations U_ℓ, each of them corresponding to a quantum logic gate. However, it can be demonstrated that any unitary operations applied on a quantum register with ℓ-qubits can be approximated to a certain accuracy ε by a finite sequence of elementary quantum gates taken from a *universal set* [47]. As any real quantum device used for computation will be able to implement a certain universal set, any quantum algorithm needs to be decomposed as a sequence of these elementary gates.

A typical example is the *standard universal set* formed of two single qubit gates: Hadamard and $\pi/8$-gates ($= R_{\pi/4}$), and one two-qubit gate: the controlled-NOT-gate. They are explicitly given by

$$R_\phi = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix}, \ \mathrm{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

along with the Hadamard gate defined previously. The controlled-gates will be important in the design of our algorithm. They operate on at least two qubits, where one of the qubit serves as a control: if the control qubit is in a certain state ($|0\rangle$ or $|1\rangle$), then some operation X is applied on other qubits. For example, if X is an arbitrary gate acting on a single qubit, that is

$$X = \begin{bmatrix} X_{00} & X_{01} \\ X_{10} & X_{11} \end{bmatrix},$$

the corresponding $C(X)$-gate which acts of two qubits and which perform the operation when the first qubit is in the state $|1\rangle$, reads

$$\mathrm{C}(X) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & X_{00} & X_{01} \\ 0 & 0 & X_{10} & X_{11} \end{bmatrix}.$$

We refer again to [47] for more details about quantum gates.

One of the main challenges in quantum computation is to determine efficient decompositions of unitary transformations in terms of elementary gates defined above. A given decomposition has an exponential speedup when the ratio of the cost (number of operations) for the classical algorithm and the cost for the quantum algorithm is an exponential function in the amount of resources. In a quantum algorithm, such sequence of operations forms a quantum circuit, discussed in more details in the next section. The purpose of this paper is to decompose a classical numerical solver for hyperbolic systems into quantum logic gates which are applied to a quantum register.

2.2 Quantum circuits

As mentioned above, the quantum algorithm has to be written as a sequence of elementary quantum gates. A convenient way to represent this decomposition visually is to use what is commonly called a quantum circuit or quantum diagram. These circuits allow for a visual representation of operations on $\ell-$qubits, where ℓ is called the width of circuit. The most simple quantum circuit is depicted in Fig. 1(a). The line represents the qubit under consideration $|u\rangle$, and H is the Hadamard matrix/operator. The circuit is read from the left to the right: first, the qubit is initialized to $|u\rangle = |0\rangle$ and then the Hadamard gate is applied, yielding the qubit in the state $|u\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$, at the right of the circuit.

A more complex example for 2-qubits is the one shown in Fig. 1(b), where the two qubits are initialized to arbitrary states. The top (resp. bottom) line represents the first (resp. second) qubit. The first part

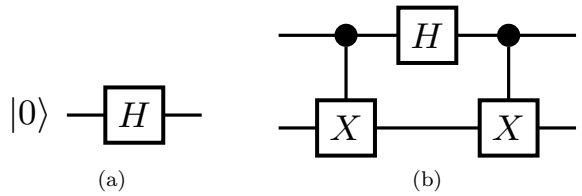


Fig. 1 (a) Hadamard gate applied to a qubit initialized to the state $|u\rangle = |0\rangle$. (b) Diagram of rotation operator in x-direction, with qubits initialized in arbitrary states.

Fig. 2 (a) Circuit diagram for the CNOT-gate. (b) Circuit diagram for the Toffoli-gate.

of the circuit represents $C(X)$-gate applied to the 2-qubits, then a Hadamard gate on the first qubit, then a $C(X)$-gate again. The vertical lines represent the control: here, if the first qubit is in the state $|1\rangle$ (represented by a black dot), then the operation X is performed on the second qubit. A white dot in the controlled operation can also be used when one needs the control qubit to be in the state $|0\rangle$ to perform the operation X.

In this paper, the controlled-not(also called CNOT-gate) and controlled-controlled-not gates (also known as Toffoli gate), are used to implement the translation operators. For 2 qubits $|s_1\rangle$, $|s_2\rangle$, the controlled-not gate transforms $|s_1\rangle \mapsto |s_1\rangle$ and $|s_2\rangle \mapsto |s_1 \oplus s_2 \mod 2\rangle$; the corresponding quantum circuit is represented in Fig. 2(a), where the "plus dot" represents the NOT-gate. These controlled gates can be generalized to any number of control qubits.

The Toffoli gates transform 3 qubits $|s_1\rangle$, $|s_2\rangle$, $|s_3\rangle$ with s_1, s_2, s_3 in $\{0,1\}$ as follows: $|s_1\rangle \mapsto |s_1\rangle$, $|s_2\rangle \mapsto |s_2\rangle$ and $|s_3\rangle \mapsto |s_3 \oplus (s_1 s_2) \mod 2\rangle$ and is represented in Fig. 2(b).

To sum-up, quantum computing works in the following way. First, the qubits forming the quantum register are initialized to a certain state, a vector in $\mathcal{H}_\ell$ of dimension 2^ℓ. Then, a sequence of unitary operations is applied to this quantum register, modifying its quantum state. The quantum circuits illustrate this procedure: each horizontal line represent a qubit, vertical lines represent control and boxes represent unitary operators. Finally, a measurement is performed where the state of the quantum system is observed.

As discussed in the following sections, the most important operations of the classical algorithms considered in this paper are the changes of basis and the translations by upwinding. At the quantum level, these operations can be performed using rotation and translation operators. The latter can themselves be decomposed as elementary quantum gates (e.g. NOT,CNOT, Hadamard, Toffoli). The scaling of the number of fundamental quantum gates as a function of quantum resources yields the computational complexity of the algorithm. This will be discussed in Section 5.

3 Reservoir method for symmetric linear first order hyperbolic systems

In this section, we recall the principle of the reservoir method for solving one-dimensional first order linear hyperbolic systems with constant matrices. This method was initially proposed in [3,5,6] and studied analytically in [40]. It basically consists of introducing flux-difference reservoirs and CFL counters in order to ensure a "CFL=1" or diffusion-less behavior of the approximate solution to hyperbolic systems of conservation laws at any time, on any characteristic field, and any location.

We start by considering the one-dimensional case because this is the main building block to treat systems with many dimensions. Indeed, our strategy for the multi-dimensional case is the use of alternate direction iteration, whereas the multi-dimensional problem is reduced to a sequence of one-dimensional problems.

3.1 Reservoir method for one-dimensional linear systems

We first present some general remarks on finite volume discretization for first order hyperbolic systems in 1-D with constant matrices and then, describe how their discretization can be improved with the reservoir method.

Specifically, we consider the following system:

$$\begin{cases} \partial_t u + A\partial_x u = 0, & (x,t) \in \mathbb{R} \times (0,T), \\ u(\cdot,0) = u_0, & x \in \mathbb{R}, \end{cases} \tag{2}$$

where $u_0 : \mathbb{R} \mapsto \mathbb{R}^m$ is a given function in $\big(L^2(\mathbb{R})\big)^m$ with compact support, so that $u_0 \in \big(L^1(\mathbb{R})\big)^m$, and $A \in S_m(\mathbb{R})$ has m real eigenvalues ordered as $|\lambda_1| \leqslant \cdots \leqslant |\lambda_m|$. The fact that u_0 has compact support allows for avoiding boundary conditions issues on finite domain and for preserving the L^2-norm of the solution. The latter can be checked using an elementary calculation: in particular, it can be shown that $\partial_t \|u(t)\|_{L^2} = 0$ if A is symmetric and if $u(t)$ has compact support. As we will see later, the conservation of the L^2-norm is important to get simple quantum algorithms. We also assume that $\|u_0\|_{L^2} = 1$.

We denote by $\{s_k\}_{k=1,\cdots,m}$ the corresponding orthonormalized right-eigenvectors, and we denote by S=col$(s_1, \cdots, s_m)$ the corresponding transition matrix. To simplify the presentation, we will assume that the eigenvalues are all non-zero. Such a system is called hyperbolic [51, 54, 29, 30]. Notice that in the basis of eigenvectors, the system is diagonal, and can then be rewritten as an uncoupled system

$$\begin{cases} \partial_t w + \Lambda\partial_x w = 0, & (x,t) \in \mathbb{R} \times (0,T), \\ w(\cdot,0) = S^{-1}u_0, & x \in \mathbb{R}, \end{cases}$$

where Λ =diag$(\lambda_1, \cdots, \lambda_m)$. The exact solution to this problem, for $k = 1, \cdots, m$, is given by

$$w_k(x,t) = w_k(x - \lambda_k t, 0). \tag{3}$$

This solution can be obtained by the method of characteristics and corresponds to a streaming with velocities $(\lambda_k)_{k=1,\cdots,m}$.

We can now look how such systems are discretized on a grid to obtain a numerical scheme. For this purpose, we introduce a sequence of nodes $\{x_j\}_{j\in\mathbb{Z}}$ (resp. $\{x_{j\pm1/2}\}_{j\in\mathbb{Z}}$), defined by $x_j = j\Delta x$ (resp. $x_{j\pm1/2} = (j \pm 1/2)\Delta x$) for a given lattice spacing $\Delta x > 0$. We also define finite volumes $\big\{\omega_j\big\}_{j\in\mathbb{Z}}$, where $\omega_j = (x_{j-1/2}, x_{j+1/2})$. We introduce a sequence of times $\{t_n\}_{n\in\mathbb{N}}$ and time steps $\{\Delta t_n\}_{n\in\mathbb{N}}$ ulteriorly determined, as well as a sequence of vectors $\big\{U_j^n\big\}_{(j,n)\in\mathbb{Z}\times\mathbb{N}}$ of $\mathbb{R}^m$ approximating $u(x_j, t_n)$, for any $j \in \mathbb{Z}$ and $n \in \mathbb{N}$. Initially, for any $j \in \mathbb{Z}$, the solution is projected on the mesh

$$U_j^0 = \frac{1}{\Delta x}\int_{\omega_j} u_0(x)dx,$$

corresponding to a finite volume formulation. For first order linear systems, the natural finite volume method consists of upwinding the solution on each characteristics field. In other words, the corresponding finite volume scheme reads

$$U_j^{n+1} = U_j^n - \frac{\Delta t_n}{\Delta x}\big(\Phi_{j+1/2}^n - \Phi_{j-1/2}^n\big), \qquad j \in \mathbb{Z},\ n \geqslant 0 \tag{4}$$

where the interface flux is given by

$$\Phi_{j-1/2}^n = \frac{1}{2}\Big\{A\big(U_j^n + U_{j-1}^n\big) - VA\big(U_j^n - U_{j-1}^n\big)\Big\},$$

with $V = S\mathrm{diag}_{k=1,\cdots,m}(\mathrm{sgn}(\lambda_k))S^{-1}$ the so-called *sign matrix*. Thus

$$U_j^{n+1} = U_j^n - \frac{\Delta t_n}{2\Delta x} A\Big\{ \big(I - V\big)(U_{j+1}^n - U_j^n) + \big(I + V\big)(U_j^n - U_{j-1}^n) \Big\}. \tag{5}$$

This is a straightforward generalization of the upwind scheme for transport equations. This is equivalent in the basis of eigenvectors to solve

$$W_j^{n+1} = W_j^n - \frac{\Delta t_n}{\Delta x} S^{-1}\big(\Phi_{j+1/2} - \Phi_{j-1/2}\big),$$

where $W_j^n = S^{-1}U_j^n$ with $W_j^n = \big(w_{1;j}^n, \cdots, w_{m;j}^n\big)^T$, which simply becomes per characteristic field

$$w_{k;j}^{n+1} = w_{k;j}^n - \frac{\lambda_k \Delta t_n}{\Delta x}(w_{k;j+1}^n - w_{k;j}^n), \qquad \lambda_k < 0, \tag{6}$$

$$w_{k;j}^{n+1} = w_{k;j}^n - \frac{\lambda_k \Delta t_n}{\Delta x}(w_{k;j}^n - w_{k;j-1}^n), \qquad \lambda_k > 0\,. \tag{7}$$

This scheme is stable iff the CFL-condition

$$\mathrm{CFL} = \frac{\Delta t_n}{\Delta x} \max_{k=1,\cdots,m} |\lambda_k| = \frac{\Delta t_n}{\Delta x} |\lambda_m| \leqslant 1$$

is satisfied. In practice, we choose CFL=1, which allows for avoiding numerical diffusion on the m'th characteristic fields, but creating some on the other ones. However, if the eigenvalues have all the same modulus, the scheme (4) provides the exact solution on the grid even if it is only first order accurate. This is typically the case with the one-dimensional Dirac equation where eigenvalues are related to the speed of light (see [25] for instance). The reservoir method [5,40,6] was precisely introduced as a tool to avoid numerical diffusion for all characteristics fields in first order finite volume schemes for hyperbolic systems of conservation laws.

We now turn to the principle of the reservoir method. The main two ingredients of the reservoir techniques are i) the CFL counters and ii) the flux difference reservoirs at the finite volume interfaces $\{x_{j+1/2}\}_{j\in\mathbb{Z}}$. Indeed, when the matrix is non-constant $A(x)$ and the system is well-posed, the reservoir technique requires the introduction of m time-dependent vectorial reservoirs at each interface $R_{1;j-1/2}, \cdots, R_{m;j-1/2} \in \mathbb{R}^m$ and initially taken null, as well as m scalar time-dependent counters $c_{1;j-1/2}, \cdots, c_{m;j-1/2} \in [0,1)$ also initialized to 0. In the specific cases discussed in this paper with a constant A matrix, the reservoirs and counters are actually interface independent, greatly simplifying the notation/implementation of the scheme. Making this assumption allows us to consider only m reservoirs $R_1, \cdots, R_m \in \mathbb{R}^m$ and m counters $c_1, \cdots, c_m$. Furthermore, we denote by $U_{\mathrm{R}}^k(U,W)$ the solution of the Riemann problem with left (resp. right) state U (resp. W), which lies between the k'th and $k+1$st wave, where we have set: $U_{\mathrm{R}}^0(U,W) = U$ and $U_{\mathrm{R}}^m(U,W) = W$. For linear systems, computing the solution of Riemann problems is almost straightforward (see [30,5] for details). Additionally, we introduce a temporary variable:

$$\mathcal{C}_k^{n+1} := c_k^n + |\lambda_k| \frac{\Delta t_n}{\Delta x}.$$

Then, we update the solution as follows

$$U_j^{n+1} = U_j^n + \sum_{k=1}^{m} \big(\widetilde{U}_{k;j-1/2}^{n+1} + \widetilde{U}_{k;j+1/2}^{n+1}\big), \tag{8}$$

where we have (if $\lambda_k < 0$)

$$\begin{pmatrix} \widetilde{U}_{k;j+1/2}^{n+1} \\ c_k^{n+1} \\ R_k^{n+1} \end{pmatrix} = \begin{cases} \begin{pmatrix} 0 \\ c_k^n + |\lambda_k| \dfrac{\Delta t_n}{\Delta x} \\ R_k^n - \dfrac{\Delta t_n}{\Delta x} A\big(U_{\mathrm{R}}^k(U_j^n, U_{j+1}^n) - U_{\mathrm{R}}^{k-1}(U_j^n, U_{j+1}^n)\big) \end{pmatrix}, & \text{when } \mathcal{C}_k^{n+1} < 1, \\ \begin{pmatrix} R_k^n - \dfrac{\Delta t_n}{\Delta x} A\big(U_{\mathrm{R}}^k(U_j^n, U_{j+1}^n) - U_{\mathrm{R}}^{k-1}(U_j^n, U_{j+1}^n)\big) \\ 0 \\ 0 \end{pmatrix}, & \text{when } \mathcal{C}_k^{n+1} = 1 \end{cases}$$

and if $\lambda_k > 0$, we have

$$\begin{pmatrix} \widetilde{U}^{n+1}_{k;j-1/2} \\ c^{n+1}_k \\ R^{n+1}_k \end{pmatrix} = \begin{cases} \begin{pmatrix} 0 \\ c^n_k + |\lambda_k| \dfrac{\Delta t_n}{\Delta x} \\ R^n_k - \dfrac{\Delta t_n}{\Delta x} A\big(U^k_{\mathrm{R}}(U^n_{j-1}, U^n_j) - U^{k-1}_{\mathrm{R}}(U^n_{j-1}, U^n_j)\big) \end{pmatrix}, & \text{when } \mathcal{C}^{n+1}_k < 1, \\ \begin{pmatrix} R^n_k - \dfrac{\Delta t_n}{\Delta x} A\big(U^k_{\mathrm{R}}(U^n_{j-1}, U^n_j) - U^{k-1}_{\mathrm{R}}(U^n_{j-1}, U^n_j)\big) \\ 0 \\ 0 \end{pmatrix}, & \text{when } \mathcal{C}^{n+1}_k = 1\,. \end{cases}$$

The time steps are finally chosen as

$$\Delta t_n = \min_{k=1,\cdots,m} \left[\big(1 - c^n_k\big) \frac{\Delta x}{|\lambda_k|} \right].$$

Although, the scheme may look complicated, it simply consists in updating the k'th components of the solution in the basis of eigenvectors, when the corresponding local counter reaches 1 or *any* prescribed value less or equal to 1.

The analysis of convergence is addressed in [40]. In particular, it was proven that the reservoir method provides exact solutions at the discrete level for linear hyperbolic systems with constant rational eigenvalues. This latter assumption was only technical, and the reservoir method is still applicable beyond this condition. More specifically, it is proven that at time say $T > 0$, the reservoir method combined with an order 1 finite volume method provides a L^1-error ε, which is bounded by the product of the maximal time step with the initial L^1-norm error (L^1-norm of the error between the exact initial data and its projection on the finite volume mesh):

$$\varepsilon = \|u(\cdot, T) - U^{n_T}\|_{L^1} < \big\|u(\cdot, t_0) - U^0\big\|_{L^1} + C \max_{k=1,\cdots,n_T} (\Delta t_k),$$

where $C > 0$ is a real constant. As a consequence in one dimension, the error remains bounded for large T's, unlike usual finite volume methods (including higher order ones, in general) for which the error grows linearly in T.

A priori, it is challenging to design a quantum algorithm that implements this numerical scheme. However, for the case considered here, where A is constant, the reservoir method can be formulated in a very simple way, more suitable for a quantum implementation. In the diagonal basis, the upwind scheme reads Eqs. (6) and (7). Then, at each time step, the reservoir technique consists of freezing the components of the solution, until the corresponding CFL counter reaches 1. In practice, we proceed as follows, assuming that we are at time t_n:

– The time step is evaluated from

$$\Delta t_{n+1} = \min_{k=1,\cdots,m} \left[\big(1 - c^n_k\big) \frac{\Delta x}{|\lambda_k|} \right].$$

– Then, the CFL counter reaches 1 for some set of components $\mathcal{K}^n = \{k^*_1, \cdots, k^*_a\}$ with $k^*_1, \cdots, k^*_a \in \{1, \cdots, m\}$, where a is the number of components that needs to be updated. We also define a set which stores the sign of the corresponding eigenvalues, that is $\mathcal{Q}^n = \{\sigma(\lambda_{k^*_1}), \cdots, \sigma(\lambda_{k^*_a})\}$, where σ is the sign function. Then, for all $k \in \mathcal{K}^n$

$$\begin{aligned} w^{n+1}_{k;j} &= w^n_{k;j-1}, \qquad \text{if } \lambda_k > 0, \\ w^{n+1}_{k;j} &= w^n_{k;j+1}, \qquad \text{if } \lambda_k < 0. \end{aligned}$$

This step corresponds to a simple translation of the solution, similar to the analytical solution given in (3). Therefore, the reservoir method reproduces the exact solution on the grid, even if the scheme is first order. Meanwhile, the other components are frozen, that is for any $k \notin \mathcal{K}^n$

$$w^{n+1}_{k;j} = w^n_{k;j}\,.$$

– The counters are updated as follow. For $k \neq k^*$

$$c_k^{n+1} = c_k^n + |\lambda_{k^*}| \frac{\Delta t_n}{\Delta x}, \text{ and } c_{k^*}^{n+1} = 0.$$

– At the end of the calculation, we can use the transition matrix S to obtain the approximate solution U^{n_T}.

The reservoir method can be simply reformulated as a list of operations. Let us denote by n_T the total number of iterations, such that $\sum_{n=1}^{n_T} \Delta t_n = T$. Next, we denote by $\mathcal{I}^n, \mathcal{S}^n$ the ordered sets of indices and signs, respectively, corresponding to the characteristic fields which have been updated up to time t_n. Generally, we have $\mathcal{I}^n := \big(\mathcal{K}^1, \cdots, \mathcal{K}^n\big)$ and $\mathcal{S}^n := \big(\mathcal{Q}^1, \cdots, \mathcal{Q}^n\big)$, and we denote the k'th element of $\mathcal{I}^{n_T}, \mathcal{S}^{n_T}$ with $k \leqslant n_T$, by $\mathcal{I}_k$ and $\mathcal{S}_k$, respectively. The only relevant information required by the quantum algorithm for linear systems with constant coefficients, is then the sets $\mathcal{I}^{n_T}$ and $\mathcal{S}^{n_T}$. In particular, there will be no need for explicitly creating and updating reservoirs or even CFL counters. In practice, a classical algorithm can be run to determine the sets $\mathcal{I}^{n_T}$ and $\mathcal{S}^{n_T}$.

Basic example. We now propose as an illustration, a simple example to construct $\mathcal{I}^{n_T}$. We consider $A \in S_3(\mathbb{R})$ with eigenvalues $\lambda_1 = 10^{-1}$, $\lambda_2 = 1$ and $\lambda_3 = 1 + 10^{-1}$. Numerically, we take $\Delta x = 10^{-2}$, $T = 10^{-1}$ and $n_T = 23$. We then find

$$\mathcal{I}^{n_T} = (3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 1, 3, 2, 3), \tag{9}$$

while the set $\mathcal{S}^{n_T}$ is trivial, containing only positive signs. The first time steps are given by $\Delta t_1 = 9.09 \times 10^{-3}$, $\Delta t_2 = 9.09 \times 10^{-4}$, $\Delta t_3 = 8.18 \times 10^{-3}$, $\Delta t_4 = 1.82 \times 10^{-3}, \cdots$.

For $A = \text{diag}(\lambda_1, \lambda_2, \lambda_3)$ and with a domain given by $[0, 1]$ (with Dirichlet boundary conditions), and an initial data $u_0(x) = 1$ for $x \leqslant 0.1$ and $u_0(x) = 0$, for $x \in (0.1, 1]$, we report the solution components at the final time in Fig. 3, and compare to the "CFL=1"-solution. As expected, the "CFL=1"-solution displays some spurious diffusion for components with the lowest eigenvalues (u_1 and u_2), seen as a smoothing of the discontinuity. On the other hand, the component with the largest eigenvalue (u_3) does not show this effect as the CFL condition is exactly one for this particular component, in stark contrast with u_1 and u_2, for which the CFL condition is lower than 1, inducing diffusion in the numerical solution. The solution obtained from the reservoir technique is very few diffused because the CFL condition is 1 at all time and space points.

3.2 Alternate direction iteration for multi-dimensional systems

We can now generalize the ideas presented in the previous section for multi-dimensional linear symmetric hyperbolic systems. In particular, we consider, for $u = u(x, t)$ and $x = (x_1, \cdots, x_d)$, the following system

$$\begin{cases} \partial_t u + \sum_{j=1}^d A^{(j)} \partial_{x_j} u = 0, & (x, t) \in \mathbb{R}^d \times (0, T), \\ u(\cdot, 0) = u_0, & x \in \mathbb{R}^d \end{cases} \tag{10}$$

where the matrices $A^{(j)} \in S_m(\mathbb{R})$ and such that for any $n = (n^{(1)}, \cdots, n^{(d)})^T \in \mathbb{R}^d$, $\sum_{j=1}^d A^{(j)} n^{(j)}$ has real eigenvalues with linearly independent eigenvectors, and u_0 has compact support and L^2-norm equal to 1. Again, the L^2-norm is conserved under these conditions. Then, for any $1 \leqslant j \leqslant d$, the eigenvalues of $A^{(j)}$ are denoted $|\lambda_1^{(j)}| < \cdots < |\lambda_m^{(j)}|$. By symmetry-assumption, for any $j \in \{1, \cdots, d\}$ there exists $S^{(j)} \in U_n(\mathbb{R})$ such that $A^{(j)} = S^{(j)} \text{diag}\big(\lambda_1^{(j)}, \cdots, \lambda_d^{(j)}\big)(S^{(j)})^T$. The corresponding reservoir-method for constant matrices and with directional splitting can be proven to be L^2-stable by construction.

As mentioned earlier, it is convenient to use the alternate direction iteration method in order to apply the 1-D reservoir method for each dimension. The alternate direction iteration method proceeds

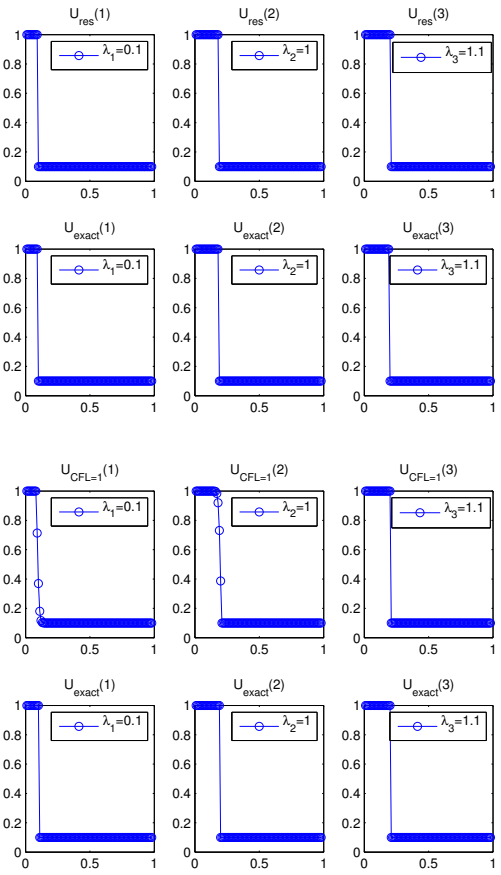


Fig. 3 Reservoir method (on the left) and upwind scheme with CFL=1 (on the right) solutions at final time $T = 0.1$.

as follows [42]. From time t_n to t_{n+1}, and assuming $u(\cdot, t_n)$ known, we successively solve

$$\left\{\begin{array}{l} \left\{\begin{array}{ll} \partial_t u^{(1)} + A^{(1)}\partial_{x_1} u^{(1)} = 0, & (x,t) \in \mathbb{R}^d \times \in (t_n, t_{n_1}), \\ u^{(1)}(\cdot, t_n) = u(\cdot, t_n), & x \in \mathbb{R}^d \end{array}\right. \\ \\ \left\{\begin{array}{ll} \partial_t u^{(2)} + A^{(2)}\partial_{x_2} u^{(2)} = 0, & (x,t) \in \mathbb{R}^d \times \in (t_{n_1}, t_{n_2}), \\ u^{(2)}(\cdot, t_n) = u^{(1)}(\cdot, t_{n_1}), & x \in \mathbb{R}^d \end{array}\right. \\ \\ \ldots \\ \\ \ldots \\ \\ \left\{\begin{array}{ll} \partial_t u^{(d)} + A^{(d)}\partial_{x_d} u^{(d)} = 0, & (x,t) \in \mathbb{R}^d \times \in (t_{n_{d-1}}, t_{n+1}), \\ u^{(d)}(\cdot, t_n) = u^{(d-1)}(\cdot, t_{n_{d-1}}), & x \in \mathbb{R}^d \end{array}\right. \end{array}\right. \tag{11}$$

Finally, we define the approximate solution at time t_{n+1} by $u(\cdot, t_{n+1}) = u^{(d)}(\cdot, t_{n+1})$. This corresponds to a first order splitting scheme and therefore, has an error that scales like $O(\Delta t_n^2)$. We note here that just like in the 1-D case, the set of all $(\Delta t_n)_{n=1,\cdots,n_T}$ is still to be determined.

We can now discretize the solution on a space grid using finite volumes to obtain the full numerical scheme. This process is very similar to the 1-D case. For this purpose, we introduce, for each dimension $i = 1, \cdots, d$, a sequence of nodes $\{x_{i;j}\}_{j\in\mathbb{Z}}$ (resp. $\{x_{i;j\pm1/2}\}_{j\in\mathbb{Z}}$), defined by $x_{i;j} = j\Delta x$ (resp. $x_{i;j\pm1/2} = (j \pm 1/2)\Delta x$) for a given $\Delta x > 0$. Then, we can define d-dimensional cubic cells as $\omega_{j_1,\cdots,j_d} = \otimes_{i=1}^d \omega_{i;j_i}$ where $\omega_{i;j} = (x_{i;j-1/2}, x_{i;j+1/2})$. The discretization then proceeds as follows. Let us introduce a sequence of vectors $\left\{U^n_{j_1,\cdots,j_d}\right\}_{(j_1,\cdots,j_d,n)\in\mathbb{Z}^d\times\mathbb{N}}$ in $\mathbb{R}^m$, approximating the mean of $u(x, t_n)$

over $\omega_{i;k}$ for any $(j_1, \cdots, j_d) \in \mathbb{Z}^d$, $n \in \mathbb{N}$. The initial data is set to

$$U^0_{j_1,\cdots,j_d} = \frac{1}{(\Delta x)^d} \int_{\omega_{j_1,\cdots,j_d}} u_0(x) dx_1 \cdots dx_d \,.$$

Then, the update of the solution parallels the 1-D case. This is possible because the splitting has transformed the multi-dimensional system to a sequence of 1-D systems, using alternate direction iteration. Therefore, we can introduce reservoirs and CFL counters at the interface between the finite volumes to implement the reservoir method. However, it was demonstrated in Section 3.1 that for constant matrices $A^{(j)}$, the reservoir are interface dependent, allowing for the introduction of $d \times m$ reservoirs $\{R_{i;k}\}$ and $d \times m$ counters $\{c_{i;k}\}$, for $i \in \{1, \cdots, d\}$ and $k \in \{1, \cdots, m\}$. However, the method amounts to the generation of a list of updated components where the reservoirs and CFL counters can be disregarded. This can be carried to the multi-dimensional case, except for one new ingredient: before each update, the system has to be diagonalized thanks to unitary operators $\{S_i\}_{1 \leqslant i \leqslant d} \in U_m(\mathbb{R})$, where i is the direction of the streaming. Setting $w^{(i)} = S^{(i)T} u^{(i)}$ for any $i \in \{1, \cdots, d\}$, the equations in Eq. (11) are transformed to uncoupled systems of transport equations in directions x_i for $1 \leqslant i \leqslant d$, of the form

$$\partial_t w^{(i)} + \Lambda^{(i)} \partial_{x_i} w^{(i)} = 0,$$

which can be solved as in the 1-D case. In particular, it is necessary to create the lists $\mathcal{I}^{n_T}$ and $\mathcal{S}^{n_T}$, similar to the ones defined in Section 3.1, which allows for sorting the translation operators. The numerical scheme at time t_n reads as follows with $U^n_{j_1,\cdots,j_d} = \left(u_{1;j_1}, \cdots, u^n_{m;j}\right)^T$ and $W^n_{j_1,\cdots,j_d} = \left(w_{1;j_1}, \cdots, w^n_{m;j}\right)^T$.

– The time step is evaluated from

$$\Delta t_{n+1} = \min_{k=1,\cdots,m} \min_{i=1,\cdots,d} \left[\left(1 - c^n_{i;k}\right) \frac{\Delta x}{|\lambda^{(i)}_k|} \right] .$$

– Then, the CFL counter reaches 1 for some set of pairs $\mathcal{K}^n = \{(k^*_1, i^*_1), \cdots, (k^*_a, i^*_a)\}$ with components $k^*_1, \cdots, k^*_a \in \{1, \cdots, m\}$ and dimensions $i^*_1, \cdots, i^*_a \in \{1, \cdots, d\}$, where a is the number of components to be updated. We also define $\mathcal{Q}^n$ as in the one-dimensional case. Then for all pairs $(k, i) \in \mathcal{K}^n$, the solution update proceeds in three steps:

1. We transform to the diagonal basis using $W^n_{j_1,\cdots,j_d} = (S^{(i)})^T U^n_{j_1,\cdots,j_d}$.
2. We evaluate the streaming step as

$$\begin{aligned} w^{n+1}_{k;j_1,\cdots,j_i,\cdots j_d} &= w^n_{k;j_1,\cdots,j_{i-1},j_i-1,j_{i+1},\cdots j_d}, \qquad \text{if } \lambda^{(i)}_k > 0, \\ w^{n+1}_{k;j_1,\cdots,j_i,\cdots j_d} &= w^n_{k;j_1,\cdots,j_{i-1},j_i+1,j_{i+1},\cdots j_d}, \qquad \text{if } \lambda^{(i)}_k < 0. \end{aligned}$$

Meanwhile, the other components are frozen, that is for any pairs for which $k' \neq k$, we have

$$w^{n+1}_{k';j_1,\cdots,j_d} = w^n_{k';j_1,\cdots,j_d} \,.$$

3. We transform back to the canonical basis using $U^n_{j_1,\cdots,j_d} = S^{(i)} W^n_{j_1,\cdots,j_d}$.

– Finally, the CFL counters are updated as follows. For $k' \neq k$ with any $i \in \{1, \cdots, d\}$, and for $k = k'$ with $i' \neq i$

$$c^{n+1}_{i';k'} = c^n_{i';k'} + |\lambda^{(i)}_k| \frac{\Delta t_n}{\Delta x}, \text{ and } c^{n+1}_{i;k} = 0 \,.$$

The list $\mathcal{I}^n$ is now a set of pairs of indices, corresponding to the characteristic fields and dimension which have been updated. It is given by $\mathcal{I}^n = \left(\mathcal{K}^1, \cdots, \mathcal{K}^n\right)$. We also have the list $\mathcal{S}^n$ which stores the sign of the eigenvalues. With these two lists, it is possible to evolve the solution in time and this is equivalent to the reservoir method combined with alternate direction iteration.

Basic example. We illustrate this approach with a two-dimensional test (another example can be found in Appendix A, for a diagonal system):

$$\begin{cases} \partial_t u + A^{(\mathrm{x})} \partial_x u + A^{(\mathrm{y})} \partial_y u = 0, & (x, y, t) \in [0, L]^2 \times (0, T), \\ u(x, y, 0) = u_0, & (x, y) \in [0, L]^2 \end{cases}$$

The matrices are defined as $A^{(\mathrm{x})} = S^{(\mathrm{x})}\Lambda^{(\mathrm{x})}S^{(\mathrm{x})T}$, and $A^{(\mathrm{y})} = S^{(\mathrm{y})}\Lambda^{(\mathrm{y})}S^{(\mathrm{y})T}$ with $\lambda_1^{(1)} = 1, \lambda_2^{(1)} = 2$ and $\lambda_1^{(2)} = 1, \lambda_2^{(2)} = 4$ and with unitary transition matrices defined by

$$S^{(\mathrm{x})} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}, \qquad S^{(\mathrm{y})} = \frac{1}{\sqrt{5}}\begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}.$$

Notice that in this case, the matrices $A^{(\mathrm{x})}$ and $A^{(\mathrm{y})}$ do not commute, but $A^{(\mathrm{x})}A^{(\mathrm{y})} - \left(A^{(\mathrm{y})}A^{(\mathrm{x})}\right)^T = 0$ and $[S^{(\mathrm{x})}, S^{(\mathrm{y})}] = 0$.

The computational domain is $[0, L = 10]^2$ and $T = 1$, $m = 2$, $d = 2$, $\Delta x_1 = \Delta x_2 = 0.1$. The initial data is $u_{0,1}(x_1, x_2) = \exp\left(-4\big((x_1-5/2)^2+(x_2-5/2)^2\big)\right)$, $u_{0,2}(x_1, x_2) = \exp\left(-4\big((x_1-5/2)^2+(x_2-5/2)^2\big)\right)$. The reservoir solution components (first and second components at time $T = 1$) are represented in Fig. 4, showing no numerical diffusion, unlike the "CFL=1"-solutions also reported in Fig. 4.

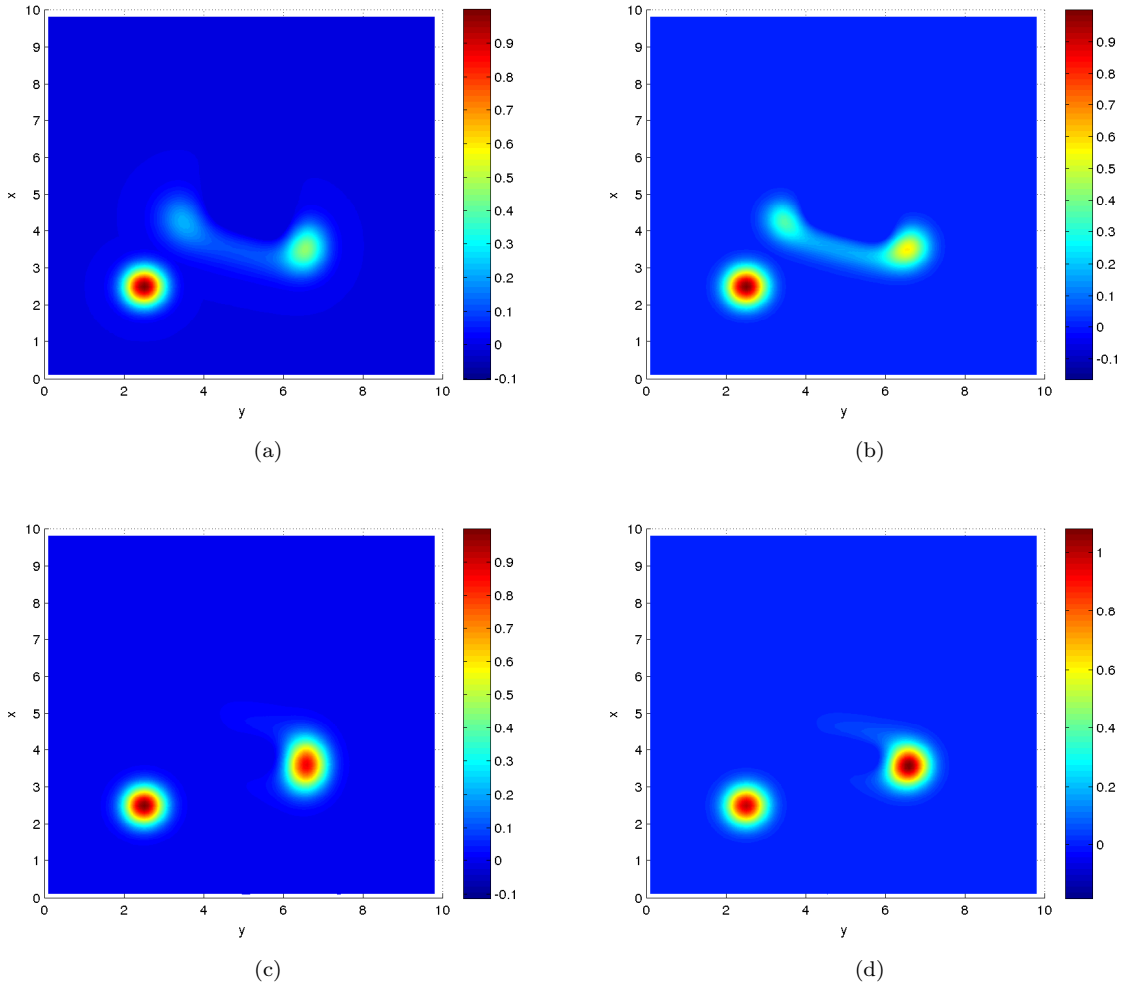


Fig. 4 first (bottom-left corner) and second component solution at time $T = 1$. (Left column) CFL=1 solution. (Right column) Reservoir method.

4 Quantum algorithm for linear first order hyperbolic systems

In this section, a quantum algorithm, i.e. an algorithm which can be implemented on a quantum computer, is developed for solving linear symmetric first order hyperbolic systems. The algorithm will be

formulated as a quantum circuit which is equivalent to the reservoir method described in the previous section.

4.1 Amplitude encoding of the solution

To develop a quantum algorithm and to benefit of the quantum nature of the computation, the first step is the encoding of the solution into the quantum register. Here, we use the same notations as in Ref. [27], and we map $U^n_{j_1,\cdots,j_d}$ into the probability amplitudes. First, we assume that the quantum register is made of $\ell = p + \sum_{i=1}^{d} n_i$ qubits. Therefore, its state is given by Eq. (1). However, for our purpose, it is more convenient to split the Hilbert space of the register in different parts as $\mathcal{H}_\ell = \mathcal{H}_p \otimes_{i=1}^{d} \mathcal{H}_{n_{x_i}}$ and to relabel the states such that

$$|u_\ell\rangle = \sum_{k=1}^{m} \sum_{j_1=1}^{N_{x_1}} \cdots \sum_{j_d=1}^{N_{x_d}} \alpha_{k,j_1,\cdots,j_d} |k\rangle \otimes |j_1,\cdots,j_d\rangle,$$

where $m = 2^p$ and $N_i = 2^{n_i}$ and where the relabeling is performed as $(k-1)_{10} = (s_1 \cdots s_p)_2$ and $(j_i - 1)_{10} = (s_{p+\sum_{l=1}^{i-1} n_l+1} \cdots s_{p+\sum_{l=1}^{i} n_l})_2 := (s_{i,1} \cdots s_{i,n_x})_2$. This means that the first p qubits label the $m = 2^p$ components of the solution, while the other $\tilde{n} := \sum_{i=1}^{d} n_i$ qubits serves to label the coordinate space positions. Because this is a tensor product, there are $N := \prod_{i=1}^{d} N_{x_i} = 2^{\tilde{n}}$ available amplitudes to store lattice coordinates.

Once the states of the quantum register are properly relabeled, the mapping of the solution using amplitude encoding is straightforward:

$$u^n_{k;j_1,\cdots,j_d} \mapsto \alpha_{k,j_1,\cdots,j_d}\,,$$

for a given time n. In other words, the coefficients of the discretized solution are mapped into the probability amplitudes of the quantum register.

4.2 Updating the solution in the quantum algorithm

For the updating of the solution in the quantum algorithm, we are seeking a sequence of unitary transformations that are equivalent to the reservoir method described in the last section. Precisely, we are looking for a unitary transformation $\widehat{V}$, such that, for all $k, j_1, \cdots, j_d$

$$u^n_{k;j_1,\cdots,j_d} \mapsto \alpha_{k,j_1,\cdots,j_d} \xrightarrow{\widehat{V}|u_\ell\rangle} \alpha'_{k,j_1,\cdots,j_d} \mapsto u^{n+1}_{k;j_1,\cdots,j_d},$$

where $u^{n+1}_{k;j_1,\cdots,j_d}$ is the same as the one obtained by the reservoir method. We will discuss the complexity of these mappings in the next section. Notice that here and in the following, we denote unitary operations on the quantum register with the "hat" notation ($\widehat{V}$).

We saw in Section 3.2 that the reservoir method, for constant matrices, amounts to a sequence of three operations: transformation to the diagonal basis, streaming and transformation to the canonical basis. The components and dimension which are subjected to these transformations are given by the list $\mathcal{I}^{n_T}$ while the streaming direction is in $\mathcal{S}^{n_T}$. We would like to perform the same operations on the quantum register and therefore, we need to construct two types of operator: translation operators and rotation operators.

4.2.1 Translation operators

The translation operators perform the streaming operations on the quantum register. Therefore, they are defined, for any (k, l) in $\mathcal{I}^{n_T}$, by

$$\begin{aligned}
\widehat{T}\big[(k,l)\big]|k\rangle \otimes |i_1,\cdots,i_d\rangle &= |k\rangle \otimes |i_1,\cdots,i_{l-1}, i_l \ominus 1 \bmod(N_l+1), i_{l+1},\cdots i_d\rangle, && \text{if } \lambda^{(l)}_k > 0,\\
\widehat{T}\big[(k,l)\big]|k\rangle \otimes |i_1,\cdots,i_d\rangle &= |k\rangle \otimes |i_1,\cdots,i_{l-1}, i_l \oplus 1 \bmod(N_l+1), i_{l+1},\cdots i_d\rangle, && \text{if } \lambda^{(l)}_k < 0.
\end{aligned}$$

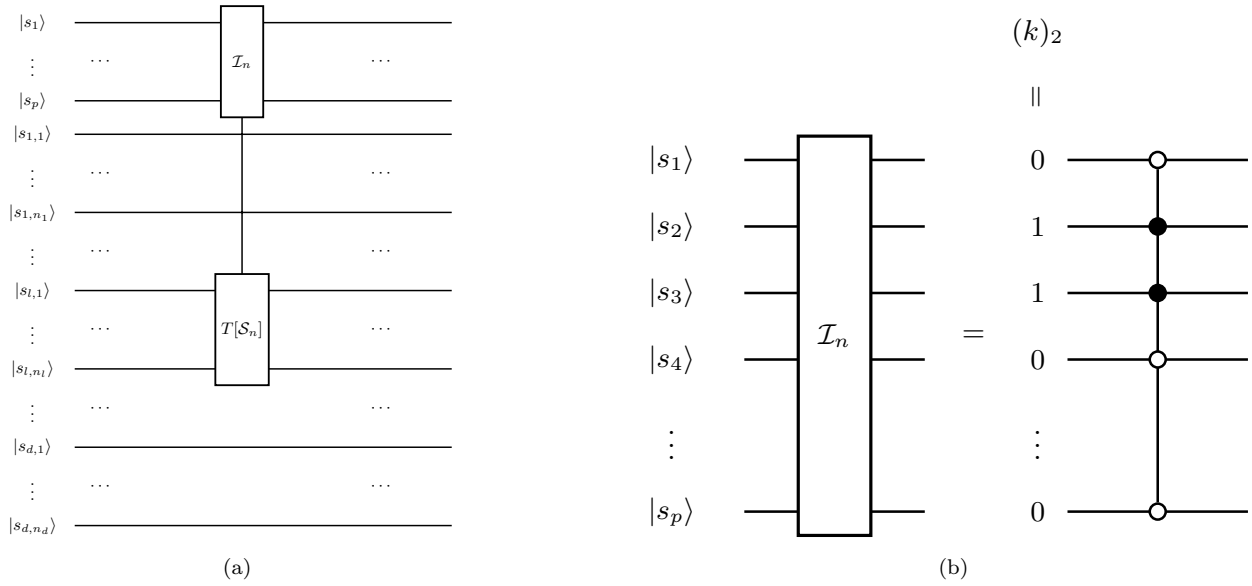


Fig. 5 (a) Circuit diagram for the implementation of the translation operator at time t_n, for the element in the list $\mathcal{I}_n = (k, l)$. The box $T[\mathcal{S}_n]$ is the shift operator on the n_l-qubits (its decomposition is depicted in Fig. 6) and the box $\mathcal{I}_n$ is a control on the first p-qubits (its explicit circuit is depicted in (b)). (b) Explicit implementation of the controlled operations, where the control is set by the component value k in the pair $\mathcal{I}_n$, expressed as a binary string.

and when $k \neq k'$

$$\widehat{T}\big[(k,l)\big]|k'\rangle \otimes |i_1, \cdots, i_d\rangle = |k'\rangle \otimes |i_1, \cdots, i_d\rangle.$$

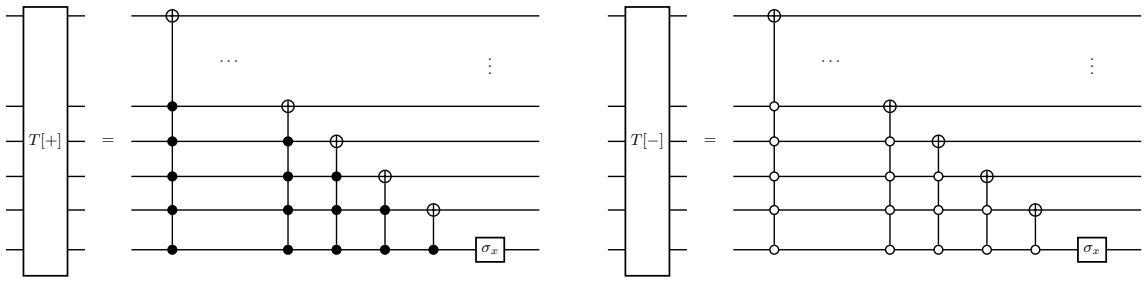
This is a unitary operation which can be represented by a quantum circuit that includes a shift operator $\widehat{T}[\pm]$ controlled by some qubits. In particular, to translate a specific component k, the shift operator has to be controlled by the first quantum register $\mathcal{H}_p$. The quantum operation is a fully controlled gate on this register and the explicit controls are determined by the value k expressed as a binary string with p digits. In particular, we have $(k)_{10\mapsto 2} = (s_1, \cdots, s_p)_2$. If the value of the digit $s_i = 0$, the control is on state $|s_i\rangle = |0\rangle$ (white dot). Conversely, if the value of the digit $s_i = 1$, the control is on state $|s_i\rangle = |1\rangle$ (black dot). These controlled operations allow for selecting and translating the proper component. Then, the sign of $\mathcal{S}_n$ determines which shift operator $\widehat{T}[\pm]$ is used. The corresponding circuit is displayed in Fig. 5.

In practice, the computations are performed on finite domains. As u_0 has compact support, for finite time, the support of the solution to the considered system also has compact support. As a consequence, the boundary conditions imposed have no influence on the solution assuming the domain is large enough such that the solution is not scattered on the boundary. Here, we use periodic boundary conditions, which can easily be implemented within the quantum algorithms developed above, thus explaining the appearance of the mod$(N_l + 1)$.

The shift operators $\widehat{T}[\pm]$ can be decomposed as a sequence of simpler unitary operations. One possible decomposition is displayed in Fig. 6 where controlled operations are used. The left (resp. right) quantum circuit corresponds to a shift from the left (resp. right) to the right (resp. left). This decomposition was first studied in [23] and used for the solution of the Dirac equation in [27]. It has a complexity which scales like $O(n_l^2)$.

4.2.2 Rotation operators

A unitary operation (diagonalization) is applied to transform the system from the canonical basis of eigenvectors of $A^{(l)}$ to the diagonal basis before and after any increment or decrement operator. In the classical algorithm, these unitary operations were denoted by $\{S^{(l)}\}_{l\in\{1,\cdots,d\}}$. We are now looking for the corresponding operators $\{\widehat{S}[l]\}_{l\in\{1,\cdots,d\}}$ which implement the same operations on the quantum register. Because the transition operators are unitary operations that rotates the characteristic fields,



(a) (b)

Fig. 6 Circuit diagram for the (a) increment and (b) decrement operator acting on qubits storing the space data of the solution (the set of n_x qubits). This implementation was first considered in [23].

the quantum operations are simply unitaries applied in the Hilbert space $\mathcal{H}_p$, as $(\widehat{S}[l]^T|k\rangle) \otimes |i_1, \cdots, i_d\rangle$. Because of the tensor product, this operation is automatically carried over all grid points, i.e. at every grid points, the fields are transformed to the diagonal basis. The explicit value of the rotation operators in the computational basis is the same as the classical $\{S^{(l)}\}_{l\in\{1,\cdots,d\}}$ and thus, they should be determined from the eigenvectors of $\{A^{(l)}\}_{l\in\{1,\cdots,d\}}$.

The next fundamental issue is the decomposition of unitary operators $\{\widehat{S}[l]\}_{l\in\{1,\cdots,d\}}$ in terms of quantum elementary gates. This question can be treated using existing techniques: $m\times m$ unitary matrices can be performed/simulated exactly using explicit quantum circuits with a quadratic complexity $O(m^2)$ using a Gray code approach [58]. In this paper and for the sake of simplicity, we will present only examples where the unitary transition matrices are relatively simple, but existing decomposition techniques for more complex matrices can perfectly be coupled to the method presented. A simple example for a rotation operator is given in Appendix B.

4.2.3 Full quantum algorithm

The quantum circuit corresponding to the full quantum algorithm for one time step is displayed in Fig. 7. The first p qubits (from top) label the components of the solution, and the next qubits label the coordinate space positions in each of the d dimensions. For $\mathcal{I}^n = (k, l)$, i) we apply a rotation $\widehat{S}[l]^T$ (change of basis, to diagonalize $A^{(l)}$) to the system, followed ii) by a translation operator $\widehat{T}[\pm k]$ of the kth component of the solution (upwinding), and finally iii) we apply $\widehat{S}[l]$ (back to canonical basis).

The resulting quantum algorithm is relatively simple because it does not use sophisticated quantum techniques such as amplitude amplification, phase estimation or the quantum solution of linear systems [34]. This is possible because hyperbolic systems with constant symmetric matrices conserve the L^2-norm. Therefore, the algorithm relies on straightforward mappings from the classical unitary operators to the quantum operators.

5 Efficiency and resource requirements

In this section, the efficiency of the preceding algorithms is analyzed. This is performed by comparing the number of required operations on a quantum computer and on a classical computer, for the same numerical method, i.e. the reservoir scheme discussed in previous sections. In particular, we are interested in the quantum speedup defined by the asymptotic behavior of [48, 49]

$$S \sim \frac{N_{\text{classical}}(\mathcal{N})}{N_{\text{quantum}}(\mathcal{N})} \quad \text{as } \mathcal{N} \to \infty$$

where $\mathcal{N}$ characterizes the system size, $N_{\text{classical}}(\mathcal{N})$ is the computational cost (or complexity) of the classical algorithm and $N_{\text{quantum}}(\mathcal{N})$ is the cost of the quantum algorithm. These costs are evaluated by counting the number of operations required to evolve the solution in time.

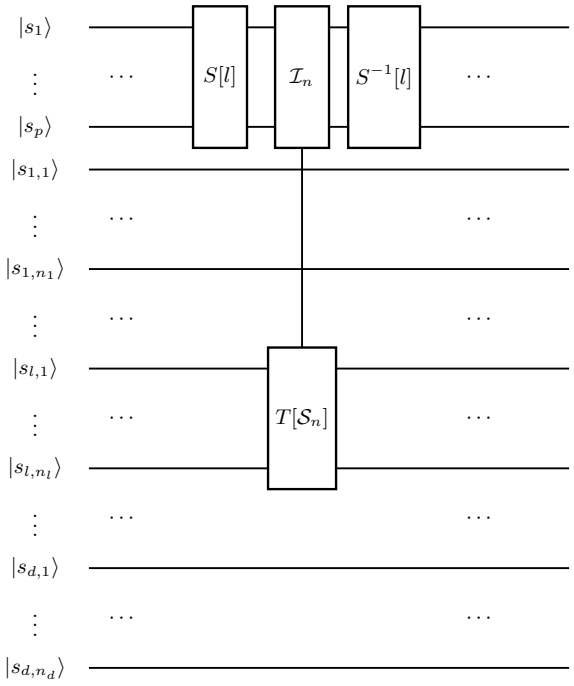


Fig. 7 Circuit diagram for the implementation of the algorithm for the d-dimensional linear system at time t_n and where $\mathcal{I}_n = (k, l)$. Here, $S[l]$ is the unitary operation that implements the diagonalization of the system. The latter has to be decomposed into a set of fundamental gates.

Proposition 51 *Let us denote by N the number of grid points in each dimension, d the number of dimensions, n_T the number of iterations, and m the number of characteristic fields. Let consider a hyperbolic system of equations with constant symmetric matrices. Then, the quantum speedup for the numerical resolution of this system using the reservoir method is*

$$S = O\left(\frac{m^2}{\log^2 m}\frac{N^d}{\log^2 N}\right), \quad \text{for } O\left(\tfrac{m^2}{\log^2 m}\left[1 + \tfrac{d}{m} + \tfrac{dm^2}{n_T}\right]\right) = O(\log^2 N),$$

corresponding to an exponential quantum speedup, when N and n_T are much larger than d and m.

Proof. The number of operations can be evaluated as follows. On a classical computer, the shift operations defined in Fig. 6 requires $O(N^d)$ swap operations to translate the solution along a given axis. Accessing the array for the component that is translated, given from the list $\mathcal{I}^{n_T}$, requires $O(1)$ operations. On the other hand, the rotation operation is a matrix-vector multiplication at all grid points, thus requiring $O(m^2 N^d)$ operations. Finding the rotation operators is equivalent to solving an eigenvalue problem for each matrix $\{A^{(j)}\}_{j=1,\cdots,d}$, typically requiring $O(dm^3)$ operations. Finally, constructing the list $\mathcal{I}^{n_T}$ requires a classical algorithm with a scaling of $O(n_T dm)$. Then, the total number of operations for n_T iterations can be written as

$$N_{\text{classical}} = O\left(n_T m^2 N^d + dm^3 + n_T dm\right).$$

For the quantum algorithm, the shift operator along an axis can be implemented using $O(\tilde{n}^2) = O(\log^2 N)$ quantum logic gates [27], where $\tilde{n}$ is the number of qubits labeling the grid points in the given direction. However, these gates are controlled by the register p: controlled gates need $O(p^2) = O(\log^2 m)$ operations [9]. As discussed earlier, the rotation operator needs $O(m^2)$ gates. If a classical algorithm is used to find the rotation matrices, the number of operations is also $O(dm^3)$, as in the classical case. This however could be improved by using a quantum algorithm, such as the Abrams-Lloyd technique [1]. Therefore, the total number of quantum logic gates after n_T iterations is

$$N_{\text{quantum}} = O\left(n_T \log^2 m \log^2 N + n_T m^2 + dm^3 + n_T dm\right).$$

Finally, taking the ratio of $N_{\text{classical}}$ and N_{quantum}, and using the fact that $n_T = O(N)$, we can evaluate the quantum speedup which proves the proposition. □

The last proposition is an important result of this article, stating that in some regimes, for a large number of grid points, our algorithm which generates a state representing the solution of a hyperbolic system of equations using the quantum algorithm is much more efficient than its classical counterparts. In terms of computational complexity, the quantum implementation has an exponential speedup over the classical implementation for the time evolution of the solution.

Here, we have neglected the initialization and reading phases of the quantum register. In particular, it is assumed that the initialization of U^0 can be implemented in polylog(N) operations. However, this is not true in the general case. As a matter of fact, the initialization of the quantum register to a general initial state U^0 requires the implementation of diagonal unitary gates, which can be implemented with uniformly controlled gates having a computational cost scaling like $O(N^2)$ [13]. For a certain class of functions, which can be integrated analytically, this can be improved to $O(\log N)$ by using the algorithm described in [65,38,33], recovering the exponential quantum speedup.

In addition, the reading of the solution is not more efficient in the quantum case in general because U^{n_T} is stored in $O(N)$ quantum amplitudes. Reconstructing these amplitudes, a process called quantum tomography [21], necessitates that the quantum algorithm is performed at least $O(N)$ times, an exponential number of operations. Therefore, to keep the efficiency of the algorithm, the final solution stored on the quantum register should either be post-processed with other efficient quantum algorithms or the measurement should be performed on some given observable $\langle u_\ell|\widehat{O}|u_\ell\rangle$, where the operator $\widehat{O}$ allows for extracting some relevant informations on the solution. In some particular cases, it may be possible to reconstruct the quantum state with some polynomial speedup using the method given in [20].

For a given error $\varepsilon > 0$, we estimate the computational resources and complexity necessary to implement the algorithm proposed above. We consider a *d-dimensional* $m = 2^p$-equation systems. We also assume that u_0 smooth with compact support, and the problem is set on a cubic domain Ω of size L^d. For a N^d-point lattice, we define $\Delta x := L/N$. The analysis of convergence for $d = 1$ is provided in [40], Theorem 2.4. We notice that the projection error of the initial data on the lattice $\|u_0 - u_h^0\|_{\ell^1}$ is bounded by $\leqslant C\|\nabla u_0\|_\infty \Delta x^d$, for some $C > 0$. In addition the directional splitting, when the matrices do not commute, creates an error in $O(\Delta t^2)$ per time iteration. For a total of n_T time iterations and using that Δt is proportional to Δx (CFL or stability condition), there exists a constant $E = E(u_0, c, m, d, \Omega) > 0$, $F > 0$, such that

$$\|u(\cdot, t_{n_T}) - u_h^{n_T}\|_1 \leqslant \varepsilon_{\text{reservoir}} + \varepsilon_{\text{splitting}}, \tag{12}$$

$$\leqslant \|u_0 - u_h^0\|_{\ell^1} + E\Delta x + F(n_T\Delta t)\Delta x, \tag{13}$$

where u (resp. $u_h^{n_T}$) denotes the exact (resp. approximate) solution, at $T = t_{n_T}$. The main feature of the reservoir method for linear systems with constant coefficients, is that the error remains bounded in time. Thus, the first term ($\varepsilon_{\text{reservoir}}$) on the right-and-side of (12) and (13) is independent of $n_T = T/\Delta t$. However, the splitting error $\varepsilon_{\text{splitting}}$ grows linearly with time. As a consequence, we have the condition $r := \varepsilon_{\text{reservoir}}/\varepsilon_{\text{splitting}} \ll 1$, i.e. the error due to reservoir is negligible compared to the splitting, for a long enough final time T, when $T = \Omega(L^{d-1}/N^{d-1})$. Then, for a given error $\varepsilon > 0$, the necessary resources are such that

$$\varepsilon = F(n_T\Delta t)\Delta x(1 + r) = FT\frac{L}{N}(1 + r).$$

As a consequence, we obtain that the number of grid points, for given error, final time and domain size should be

$$N = F(1 + r)\frac{TL}{\varepsilon} = O\left(\frac{TL}{\varepsilon}\right). \tag{14}$$

As $\log_2(N) = \tilde{n}$, the total number of qubits necessary for representing the solution after n_T iterations, with an error ε and for T large enough, is given by

$$\tilde{n} = \log\left(F(1 + r)\frac{TL}{\varepsilon}\right) = O\left[\log\left(\frac{TL}{\varepsilon}\right)\right]. \tag{15}$$

Notice that in the case of diagonal matrices, N is "only" a $O(L/\varepsilon)$, due to the commutation of the matrices. Reporting these results for N and $\tilde{n}$ in the cost of the algorithm, we can evaluate the speedup in terms of the problem parameters. We finally deduce the following proposition:

Proposition 52 *Let us denote by T the final time, d the number of dimensions, L the size of the domain in each dimension, m the number of characteristic fields, and ε the numerical error. We consider a hyperbolic system of equations with constant symmetric matrices. Then, the quantum speedup for the numerical computation of this system using the reservoir method is*

$$S = O\left(\frac{m^2}{\log^2 m}\frac{T^d L^d}{\varepsilon^d}\frac{1}{\log^2\left(\frac{TL}{\varepsilon}\right)}\right),$$

which corresponds to an exponential quantum speedup.

The proof essentially follows the same logic as for Proposition 51, but replacing N and $\tilde{n}$ by (14) and (15), respectively.

The previous results considered the asymptotic computational complexity of the classical and quantum algorithms. However, it is interesting to look at minimal examples to verify if a proof-of-principle calculation could be performed on actual quantum computers. Two such examples are described in Appendix C where explicit gate decompositions are carried out with `Quipper`.

6 Some possible extensions of the proposed quantum algorithms

In this section, we propose some ideas to extend the algorithms proposed above. First, we discuss the extension of the above quantum algorithms to linear hyperbolic equations with space-dependent velocity. In the second part of this section, we discuss method-of-line based quantum algorithms for linear hyperbolic systems.

6.1 Reservoir method for linear hyperbolic equations with non-constant velocity

As introduction to this problem, we consider the following one-dimensional transport equation

$$\begin{cases} \partial_t u + \partial_x\big(A(x)u\big) = 0, & (x,t) \in \mathbb{R}\times(0,T), \\ u(\cdot,0) = u_0, & x \in \mathbb{R} \end{cases} \tag{16}$$

where A is assumed smooth, with a derivative denoted $a(x) := \partial_x A(x)$. We also assume that u_0 is assumed smooth with compact support. For the sake of simplicity of the presentation and notation, we will assume that $a(x) > 0$.The L^2-norm of the solution to (16) is not preserved in general, instead:

$$\frac{d}{dt}\int_{\mathbb{R}} A(x)|u(x,t)|^2 dx = 0.$$

However, it is still possible to implement a quantum algorithm preserving the $\ell^2(\mathbb{Z})-$norm of the quantum register. This problem can be reduced to a constant velocity transport equation by using a change of variable $y = f(x) = \int^x a^{-1}(x')dx'$, allowing for an analytical solution when f can be obtained in analytical form. If this is not available, one should resort to a numerical approach like the reservoir method. In this case however, the reservoirs and counters are space-dependent. Quantum algorithms are based on unitary transformations. By default, the reservoir method for non-constant velocity on uniform mesh, is a priori not based on unitary operations. We then propose a version of the reservoir method on non-uniform mesh. More specifically, we define a sequence grid points $\left\{x_{j+1/2}\right\}_{j\in\mathbb{Z}}$, and one-dimensional volumes $\{\omega_j\}_{j\in\mathbb{Z}}$, with $\omega_j := (x_{j-1/2}, x_{j+1/2})$ of lengths $\Delta x_j := x_{j+1/2} - x_{j-1/2}$. We denote, for any $j \in \mathbb{Z}$

$$u_j^0 := \frac{1}{\Delta x_j}\int_{\omega_j} u_0(x)dx$$

and we denote $\{u_j^n\}_{(j,n)\in\mathbb{Z}\times\mathbb{N}}$ the sequences approximating

$$\left\{\frac{\Delta t_n}{\Delta x_j}\int_{\omega_j} u(x,t)dx\right\}_{(j,n)\in\mathbb{Z}\times\mathbb{N}}.$$

We also denote for $j \in \mathbb{Z}$, $a_{j-1/2} := a(x_{j-1/2})$.

For $a(x)$ assumed fixed (and positive), it is convenient to consider a non-uniform mesh as follows: $\omega_j = (x_{j-1/2}, x_{j+1/2})$ with non-constant $\Delta x_j = x_{j+1/2} - x_{j-1/2}$ such that

$$\Delta x_j = \begin{cases} \dfrac{a_{j-1/2}}{1 + \lfloor a_{j-1/2} \rfloor}, & \text{if } a_{j-1/2} > 1, \\ a_{j-1/2}, & \text{if } a_{j-1/2} \leqslant 1\,. \end{cases}$$

Notice that by construction, for any $j \in \mathbb{Z}$

$$\ell_j := \frac{a_{j-1/2}}{\Delta x_j} \in \mathbb{N}^*\,.$$

Thus, the reservoir method can then be rewritten

$$\begin{pmatrix} u_j^{n+1} \\ c_{j-1/2}^{n+1} \\ r_{j-1/2}^{n+1} \end{pmatrix} = \begin{cases} \begin{pmatrix} 0 \\ c_{j-1/2}^n + \ell_j \Delta t_n \\ r_{j-1/2}^n - \ell_j \Delta t_n (u_j^n - u_{j-1}^n) \end{pmatrix}, & \text{when } c_{j-1/2}^n + \ell_j \Delta t_n < 1, \\ \begin{pmatrix} u_j^n + r_{j-1/2}^n - \Delta t_n \ell_j (u_j^n - u_{j-1}^n) \\ 0 \\ 0 \end{pmatrix}, & \text{when } c_{j-1/2}^n + \ell_j \Delta t_n = 1\,. \end{cases}$$

The counters can then be defined by $c_{j-1/2}^n = \ell_j \sum_{k=p_j}^{n-1} \Delta t_k$, for some $0 \leqslant p_j \leqslant n$ and

$$\Delta t_n = \min_j \left[(1 - c_{j-1/2}^n) \frac{1}{\ell_j} \right]\,.$$

Notice that if a is small enough, we can simplify even more the algorithm, and we can determine a priori the time steps and the list of space indices to be updated per iteration, $\mathcal{I}^{n_T}$ as in the case of constant velocity. In order to achieve this procedure, we construct the grid nodes $\{x_{j+1/2}\}_{j\in\mathbb{Z}}$ such that for all $j \in \mathbb{Z}$, $a_{j-1/2}/\Delta x_j = \ell$ with ℓ constant. Initially, the counters and reservoirs are as usual, empty. The counters are of the form $c_{j-1/2}^n = \ell \sum_{k=p}^{n-1} \Delta t_k$, for some $0 \leqslant p \leqslant n$ (for $p = n$, the counters are null). In other words, the counters as well as the time steps are spatially independent:

$$\Delta t_n = \frac{1}{\ell}\Big(1 - \ell \sum_{k=p}^{n-1} \Delta t_k\Big)\,.$$

In practice, we then have for any $n \in \mathbb{N}$:

$$\Delta t_n = \frac{1}{\ell}, \qquad \text{and } u_j^{n+1} = u_{j-1}^n\,.$$

At the quantum level for $a(x) > 0$, the algorithm is simply given by $T_x|i\rangle = |i \ominus 1 \bmod(N)\rangle$. Recall though that this diffusion-less approach only works for very specific velocities, and is a priori not conservative.

This algorithm should also be efficient on a quantum computer, as it is based on the application of the shift operator, as in the constant velocity cases, and therefore, scales like $O(n_T \log^2 N)$. However, finding the grid point positions and the time step may require an exponential amount of resources. A naive classical algorithm for this task requires $O(N)$ operations. Therefore, the method is more efficient on a quantum computer as long as it is possible to find a quantum algorithm that can evaluate these grid positions more efficiently than $O(N)$.

Example. We illustrate the above approach with the following simple example.

$$\partial_t u + a(x)\partial_x u = 0,\ (x,t) \in (0,70) \times (0,T)$$

where a velocity is randomly constructed, using a uniform probability density function, $\mathcal{U}(0,1)$. It is defined $\{a_{j-1/2}\}_j$ with $a_{j-1/2} = a(x_{j-1/2})$ see Fig. 8 (Left), and where $x_{j+1} = x_j + \Delta x_j$ with $\Delta x_j = a_{j-1/2}$. In particular $a_{j-1/2}/\Delta x_j = \ell$ with $\ell = 1$ in the following, and $T = 400$. We report in Fig. 8

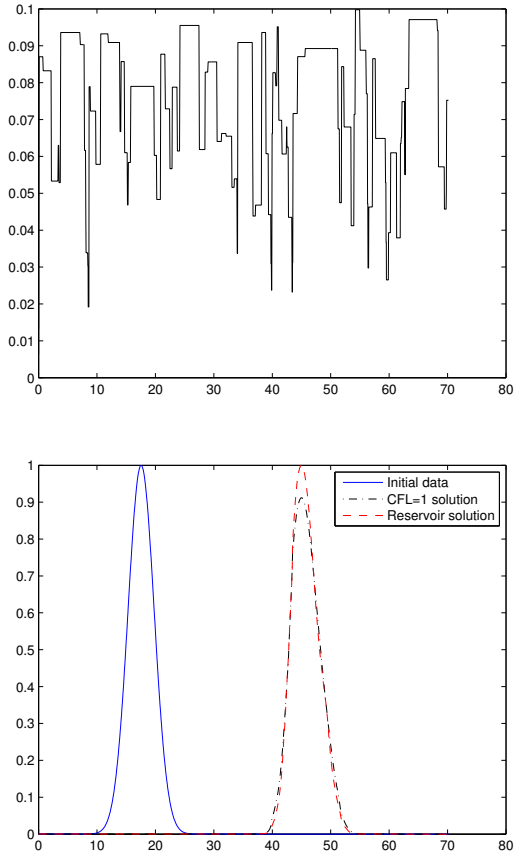


Fig. 8 (Left) Non-constant velocity $\{(x, a(x)),\, x \in (0, 70)\}$. (Right) CFL=1 (on uniform mesh) and reservoir (CFL=1 on non-uniform mesh) solutions.

(Right), the solution reservoir solution[1] $\{u_j^n\}_{(j,n)\in\mathbb{Z}\times\mathbb{N}}$

$$u_j^{n+1} = u_{j-1}^n$$

The $\ell^2(\mathbb{Z})-$ norm of $\{u_j^n\}_j$ is trivially well-preserved, while $\ell^2(\Delta x\mathbb{Z})$ is not. The CFL=1 solution $\{v_j^n\}_{(j,n)\in\mathbb{Z}\in\mathbb{N}}$ is constructed on uniform mesh $\Delta x_j = \Delta x$, given by (as the $a(x) > 0$):

$$v_j^{n+1} = v_j^n - a_{j-1/2}\frac{\Delta t}{\Delta x}(v_j^{n+1} - v_j^n)$$

at CFL=1, corresponding to a time step satisfying

$$\Delta t = \frac{\Delta x}{\max_j |a_{j-1/2}|}.$$

Notice that using a directional splitting, it is easy to extend this idea to d-dimensional transport equations for $u := u(x_1, \cdots, x_d, t)$, of the form

$$\partial_t u + \sum_{i=1}^{d} a_i(x_i)\partial_{x_i} u = 0,\ (x_1 \cdots, x_d, t) \in \mathbb{R}^d \times \mathbb{R}_+$$

where $\{a_i(x_i)\}_{1\leqslant i\leqslant d}$ is a sequence of given functions. In this case, the space steps are simply defined directionally as follows: $\{\Delta x_{i;j}\}_{j\in\mathbb{Z}} = \{a_i(x_{i;j+1/2})\}_{j\in\mathbb{Z}}$.

[1] this terminology is here a bit abusive, as this is nothing but a "CFL=1"-solution on a non-uniform mesh.

6.2 Generalization to other numerical scheme: Method of lines

Other numerical schemes could also be considered. For instance, the method of lines is particularly interesting [14]. In this case, the hyperbolic system is discretized in space only, and can be written as

$$\frac{dU}{dt} = \mathcal{A}U, \tag{17}$$

where $\mathcal{A}$ is a sparse matrix obtained from the space discretization. When the matrix $\mathcal{A}$ is skew-symmetric the solution to the initial value problem is given by an orthogonal transformation as

$$U^{n_T} = e^{\mathcal{A}T}U^0. \tag{18}$$

The orthogonal operation $e^{\mathcal{A}T}$ can then be implemented efficiently for sparse matrices on a quantum computer, see for instance [15,2]. However, the construction of the matrix $\mathcal{A}$ may be efficient only for a subclass of all matrices. This promising avenue will be also explored in a forthcoming paper.

7 Conclusion

In this paper, we have proposed an efficient quantum scheme for the solution of first order linear hyperbolic systems by combining the diffusion-less reservoir method, and operator splitting. We demonstrated that in some cases, namely for a large class of multi-dimensional hyperbolic systems with constant symmetric matrices and for one-dimensional scalar hyperbolic equations with non-constant velocity, the reservoir method, along with alternate direction iteration, can be simplified significantly and becomes a set of streaming steps $\mathcal{I}^{n_T}$. These steps can be implemented efficiently on a quantum computer. We have also shown that the combination of these techniques yields a speedup over the classical implementation of the same numerical scheme, promising interesting performance. However, as with other similar quantum algorithms, the measurement of the solution, encoded in the probability amplitudes, requires $O(N)$ operations which is not more efficient than a classical algorithm. To be useful, the algorithm has to be combined with some efficient post-processing procedure that either computes some observables or that yields some general properties of the solution.

The generalization of this quantum algorithm to more general hyperbolic systems is certainly harder to obtain. First, for other types of matrices (non-constant and non-symmetric), the L^2-norm is not preserved, implying that non-unitary operations has to be implemented on the quantum computer. It is possible in principle to use non-unitary operations by using projective measurements at every time steps [16] but this is certainly more challenging to implement. Second, the space-dependent matrices has to be diagonalized at every grid points, which requires $O(N^d m^3)$ operations for a classical algorithm, instead of $O(dm^3)$ for constant matrices. It is unclear how efficient this operation can be on a quantum computer. Finally, when the matrices depends on space, the reservoir and CFL counters has to be considered explicitly at each volume interface and updated according to (8). Again, it is unclear if an efficient quantum algorithm can be formulated for this purpose.

The techniques and ideas developed in this paper, can be generalized to the derivation of efficient algorithms solving other types of linear differential systems. However, the required key element is the use of an efficient implementation of translation operators which allow for an efficient overall algorithm. We are currently investigating the extension of some of the ideas presented in this paper to first order nonlinear hyperbolic equations. In principle, it is possible to derive explicit quantum versions of simple first order finite difference/volume schemes for any linear partial differential equation; however in addition to the poor accuracy, deriving efficient quantum versions of these classical algorithms, that is having exponential speedups w.r.t. classical algorithms, is far from trivial and is even the source of many open problems, which could certainly be investigated by numerical analysts.

References

1. Daniel S. Abrams and Seth Lloyd. Quantum algorithm providing exponential speed increase for finding eigenvalues and eigenvectors. *Phys. Rev. Lett.*, 83:5162–5165, Dec 1999.
2. Dorit Aharonov and Amnon Ta-Shma. Adiabatic quantum state generation and statistical zero knowledge. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 20–29. ACM, 2003.

3. F. Alouges, F. De Vuyst, G. Le Coq, and E. Lorin. A process of reduction of the numerical diffusion of usual order one flux difference schemes for nonlinear hyperbolic systems [un procédé de réduction de la diffusion numérique des schémas à différence de flux d'ordre un pour les systèmes hyperboliques non linéaires]. *Comptes Rendus Mathematique*, 335(7):627–632, 2002.
4. F. Alouges, F. De Vuyst, G. Le Coq, and E. Lorin. The reservoir scheme for systems of conservation laws. In *Finite volumes for complex applications, III (Porquerolles, 2002)*, pages 247–254. Hermes Sci. Publ., Paris, 2002.
5. F. Alouges, F. De Vuyst, G. Le Coq, and E. Lorin. The reservoir technique: a way to make godunov-type schemes zero or very low diffuse. application to Colella-Glaz solver. *European Journal of Mechanics, B/Fluids*, 27(6):643–664, 2008.
6. F. Alouges, G. Le Coq, and E. Lorin. Two-dimensional extension of the reservoir technique for some linear advection systems. *J. of Sc. Comput.*, 31(3):419–458, 2007.
7. Pablo Arrighi, Vincent Nesme, and Marcelo Forets. The Dirac equation as a quantum walk: higher dimensions, observational convergence. *Journal of Physics A: Mathematical and Theoretical*, 47(46):465302, 2014.
8. Alán Aspuru-Guzik, Anthony D. Dutoi, Peter J. Love, and Martin Head-Gordon. Simulated quantum computation of molecular energies. *Science*, 309(5741):1704–1707, 2005.
9. A. Barenco, C.H. Bennett, R. Cleve, D.P. Divincenzo, N. Margolus, P. Shor, T. Sleator, J.A. Smolin, and H. Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 52(5), 1995.
10. R Barends, L Lamata, J Kelly, L García-Álvarez, AG Fowler, A Megrant, E Jeffrey, TC White, D Sank, JY Mutus, et al. Digital quantum simulation of fermionic models with a superconducting circuit. *Nature communications*, 6, 2015.
11. Rami Barends, Alireza Shabani, Lucas Lamata, Julian Kelly, Antonio Mezzacapo, Urtzi Las Heras, Ryan Babbush, AG Fowler, Brooks Campbell, Yu Chen, et al. Digitized adiabatic quantum computing with a superconducting circuit. *Nature*, 534(7606):222–226, 2016.
12. Giuliano Benenti and Giuliano Strini. Quantum simulation of the single-particle schroedinger equation. *American Journal of Physics*, 76(7):657–662, 2008.
13. Ville Bergholm, Juha J. Vartiainen, Mikko Möttönen, and Martti M. Salomaa. Quantum circuits with uniformly controlled one-qubit gates. *Phys. Rev. A*, 71:052330, May 2005.
14. Dominic W Berry. High-order quantum algorithm for solving linear differential equations. *Journal of Physics A: Mathematical and Theoretical*, 47(10):105301, 2014.
15. Dominic W. Berry, Graeme Ahokas, Richard Cleve, and Barry C. Sanders. Efficient quantum algorithms for simulating sparse hamiltonians. *Communications in Mathematical Physics*, 270(2):359–371, 2007.
16. Andreas Blass and Yuri Gurevich. Ancilla-approximable quantum state transformations. *Journal of Mathematical Physics*, 56(4):042201, 2015.
17. Bruce M Boghosian and Washington Taylor. Simulating quantum mechanics on a quantum computer. *Physica D: Nonlinear Phenomena*, 120(1):30–42, 1998.
18. Katherine L. Brown, William J. Munro, and Vivien M. Kendon. Using quantum computers for quantum simulation. *Entropy*, 12(11):2268, 2010.
19. Yudong Cao, Anargyros Papageorgiou, Iasonas Petras, Joseph Traub, and Sabre Kais. Quantum algorithm and circuit design solving the poisson equation. *New Journal of Physics*, 15(1):013021, 2013.
20. Marcus Cramer, Martin B Plenio, Steven T Flammia, Rolando Somma, David Gross, Stephen D Bartlett, Olivier Landon-Cardinal, David Poulin, and Yi-Kai Liu. Efficient quantum state tomography. *Nature Communications*, 1:149, 2010.
21. G Mauro D'Ariano, Matteo GA Paris, and Massimiliano F Sacchi. Quantum tomography. *Advances in Imaging and Electron Physics*, 128:206–309, 2003.
22. D. Deutsch. Quantum theory, the church-turing principle and the universal quantum computer. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 400(1818):97–117, 1985.
23. B. L. Douglas and J. B. Wang. Efficient quantum circuit implementation of quantum walks. *Phys. Rev. A*, 79:052335, May 2009.
24. Richard P Feynman. Simulating physics with computers. *International journal of theoretical physics*, 21(6):467–488, 1982.
25. F. Fillion-Gourdeau, E. Lorin, and A. D. Bandrauk. Numerical solution of the time-dependent Dirac equation in coordinate space without fermion-doubling. *Comput. Phys. Comm.*, 183(7):1403 – 1415, 2012.
26. F. Fillion-Gourdeau, E. Lorin, and A.D. Bandrauk. Resonantly enhanced pair production in a simple diatomic model. *Phys. Rev. Lett.*, 110(1), 2013.
27. François Fillion-Gourdeau, Steve MacLean, and Raymond Laflamme. Algorithm for the solution of the dirac equation on digital quantum computers. *Phys. Rev. A*, 95:042343, Apr 2017.
28. I. M. Georgescu, S. Ashhab, and Franco Nori. Quantum simulation. *Rev. Mod. Phys.*, 86:153–185, Mar 2014.
29. E. Godlewski and P.-A. Raviart. *Hyperbolic systems of conservation laws*, volume 3/4 of *Mathématiques & Applications (Paris) [Mathematics and Applications]*. Ellipses, Paris, 1991.
30. E. Godlewski and P.-A. Raviart. *Numerical approximation of hyperbolic systems of conservation laws*, volume 118 of *Applied Mathematical Sciences*. Springer-Verlag, New York, 1996.
31. A.S. Green, P.L. Lumsdaine, N.J. Ross, P. Selinger, and B. Valiron. An introduction to quantum programming in quipper. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7948 LNCS:110–124, 2013.
32. A.S. Green, P.L. Lumsdaine, N.J. Ross, P. Selinger, and B. Valiron. Quipper: A scalable quantum programming language. pages 333–342, 2013.
33. Lov Grover and Terry Rudolph. Creating superpositions that correspond to efficiently integrable probability distributions. *arXiv preprint quant-ph/0208112*, 2002.
34. A. W. Harrow, A. Hassidim, and S. Lloyd. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.*, 103(15):150502, 4, 2009.
35. Stephen P Jordan, Keith SM Lee, and John Preskill. Quantum algorithms for quantum field theories. *Science*, 336(6085):1130–1133, 2012.

36. Ivan Kassal, Stephen P Jordan, Peter J Love, Masoud Mohseni, and Alán Aspuru-Guzik. Polynomial-time quantum algorithm for the simulation of chemical dynamics. *Proceedings of the National Academy of Sciences*, 105(48):18681–18686, 2008.
37. Ivan Kassal, James D Whitfield, Alejandro Perdomo-Ortiz, Man-Hong Yung, and Aln Aspuru-Guzik. Simulating chemistry using quantum computers. *Annual review of physical chemistry*, 62:185207, 2011.
38. Phillip Kaye and Michele Mosca. Quantum networks for generating arbitrary quantum states. *arXiv preprint quant-ph/0407102*, 2004.
39. Julian Kelly, R Barends, AG Fowler, A Megrant, E Jeffrey, TC White, D Sank, JY Mutus, B Campbell, Yu Chen, et al. State preservation by repetitive error detection in a superconducting quantum circuit. *Nature*, 519(7541):66–69, 2015.
40. S. Labbé and E. Lorin. On the reservoir technique convergence for nonlinear hyperbolic conservation laws. I. *J. Math. Anal. Appl.*, 356(2):477–497, 2009.
41. B. P. Lanyon, C. Hempel, D. Nigg, M. Müller, R. Gerritsma, F. Zähringer, P. Schindler, J. T. Barreiro, M. Rambach, G. Kirchmair, M. Hennrich, P. Zoller, R. Blatt, and C. F. Roos. Universal digital quantum simulation with trapped ions. *Science*, 334(6052):57–61, 2011.
42. Randall J LeVeque. *Finite volume methods for hyperbolic problems*, volume 31. Cambridge university press, 2002.
43. S. Lloyd. Universal Quantum Simulators. *Science*, 273:1073–1078, August 1996.
44. David A. Meyer. Quantum computing classical physics. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 360(1792):395–405, 2002.
45. A Mezzacapo, M Sanz, L Lamata, IL Egusquiza, S Succi, and E Solano. Quantum simulator for transport phenomena in fluid flows. *Scientific reports*, 5, 2015.
46. C. Negrevergne, T. S. Mahesh, C. A. Ryan, M. Ditty, F. Cyr-Racine, W. Power, N. Boulant, T. Havel, D. G. Cory, and R. Laflamme. Benchmarking quantum control methods on a 12-qubit system. *Phys. Rev. Lett.*, 96:170501, May 2006.
47. Michael A Nielsen and Isaac L Chuang. *Quantum computation and quantum information*. Cambridge university press, 2010.
48. Anargyros Papageorgiou and Joseph F. Traub. Measures of quantum computing speedup. *Phys. Rev. A*, 88:022316, Aug 2013.
49. Troels F. Rønnow, Zhihui Wang, Joshua Job, Sergio Boixo, Sergei V. Isakov, David Wecker, John M. Martinis, Daniel A. Lidar, and Matthias Troyer. Defining and detecting quantum speedup. *Science*, 345(6195):420–424, 2014.
50. Y. Salathé, M. Mondal, M. Oppliger, J. Heinsoo, P. Kurpiers, A. Potočnik, A. Mezzacapo, U. Las Heras, L. Lamata, E. Solano, S. Filipp, and A. Wallraff. Digital quantum simulation of spin models with circuit quantum electrodynamics. *Phys. Rev. X*, 5:021027, Jun 2015.
51. D. Serre. *Systèmes de lois de conservation. I.* Fondations. [Foundations]. Diderot Editeur, Paris, 1996. Hyperbolicité, entropies, ondes de choc. [Hyperbolicity, entropies, shock waves].
52. P. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.
53. Siddhartha Sinha and Peter Russer. Quantum computing algorithm for electromagnetic field simulation. *Quantum Information Processing*, 9(3):385–404, 2010.
54. J. Smoller. *Shock waves and reaction-diffusion equations*, volume 258 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Science]*. Springer-Verlag, New York-Berlin, 1983.
55. R. Somma, G. Ortiz, J. E. Gubernatis, E. Knill, and R. Laflamme. Simulating physical phenomena by quantum networks. *Phys. Rev. A*, 65:042323, Apr 2002.
56. Andrew Steane. Quantum computing. *Reports on Progress in Physics*, 61(2):117, 1998.
57. J. C. Strikwerda. *Finite Difference Schemes and Partial Differential Equations*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition, 2004.
58. J.J. Vartiainen, M. Mtinen, and M.M. Salomaa. Efficient decomposition of quantum gates. *Physical Review Letters*, 92(17):177902–1, 2004.
59. Xi-Lin Wang, Luo-Kan Chen, W. Li, H.-L. Huang, C. Liu, C. Chen, Y.-H. Luo, Z.-E. Su, D. Wu, Z.-D. Li, H. Lu, Y. Hu, X. Jiang, C.-Z. Peng, L. Li, N.-L. Liu, Yu-Ao Chen, Chao-Yang Lu, and Jian-Wei Pan. Experimental ten-photon entanglement. *Phys. Rev. Lett.*, 117:210502, Nov 2016.
60. Nathan Wiebe, Dominic Berry, Peter Hyer, and Barry C Sanders. Higher order decompositions of ordered operator exponentials. *Journal of Physics A: Mathematical and Theoretical*, 43(6):065203, 2010.
61. Stephen Wiesner. Simulations of many-body quantum systems by a quantum computer. *arXiv preprint quant-ph/9603028*.
62. Man-Hong Yung, Daniel Nagaj, James D. Whitfield, and Alán Aspuru-Guzik. Simulation of classical thermal states on a quantum computer: A transfer-matrix approach. *Phys. Rev. A*, 82:060302, Dec 2010.
63. Man-Hong Yung, James D. Whitfield, Sergio Boixo, David G. Tempel, and Aln Aspuru-Guzik. *Introduction to Quantum Algorithms for Physics and Chemistry*, pages 67–106. John Wiley & Sons, Inc., 2014.
64. Christof Zalka. Efficient simulation of quantum systems by quantum computers. *Fortschritte der Physik*, 46(6-8):877–879, 1998.
65. Christof Zalka. Simulating quantum systems on a quantum computer. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 454(1969):313–322, 1998.

A Numerical example for the reservoir method with diagonal system

We now illustrate this approach with a simple classical diagonal two-dimensional test, with $\lambda_1^{(1)} = 1, \lambda_2^{(1)} = 4, \lambda_3^{(1)} = 8$ and $\lambda_1^{(2)} = 1, \lambda_2^{(2)} = 2, \lambda_3^{(2)} = 4$. The computational domain is $[0, 10]^2$, and $T = 0.75$, $m = 3$, $d = 2$, $\Delta x_1 = \Delta x_2 = 0.1$. The time step is given by $\Delta t_n = 0.012488$ for all $n \leqslant n_T = 60$. The initial data is $U_{0;1}(x_1, x_2) = \exp\Big(-2\big((x_1 - 5/2)^2 + (x_2 -$

$5/2)^2)\Big)$, $U_{0;2}(x_1,x_2) = \exp\Big(-4\big((x_1-5/2)^2+(x_2-5/2)^2\big)\Big)$, $U_{0;3}(x_1,x_2) = \exp\Big(-8\big((x_1-5/2)^2+(x_2-5/2)^2\big)\Big)$. The reservoir solution components are represented in Fig. 9 (2nd column), showing no numerical diffusion whatsoever, unlike the "CFL=1"-solutions also represented in Fig. 9 (3rd column).

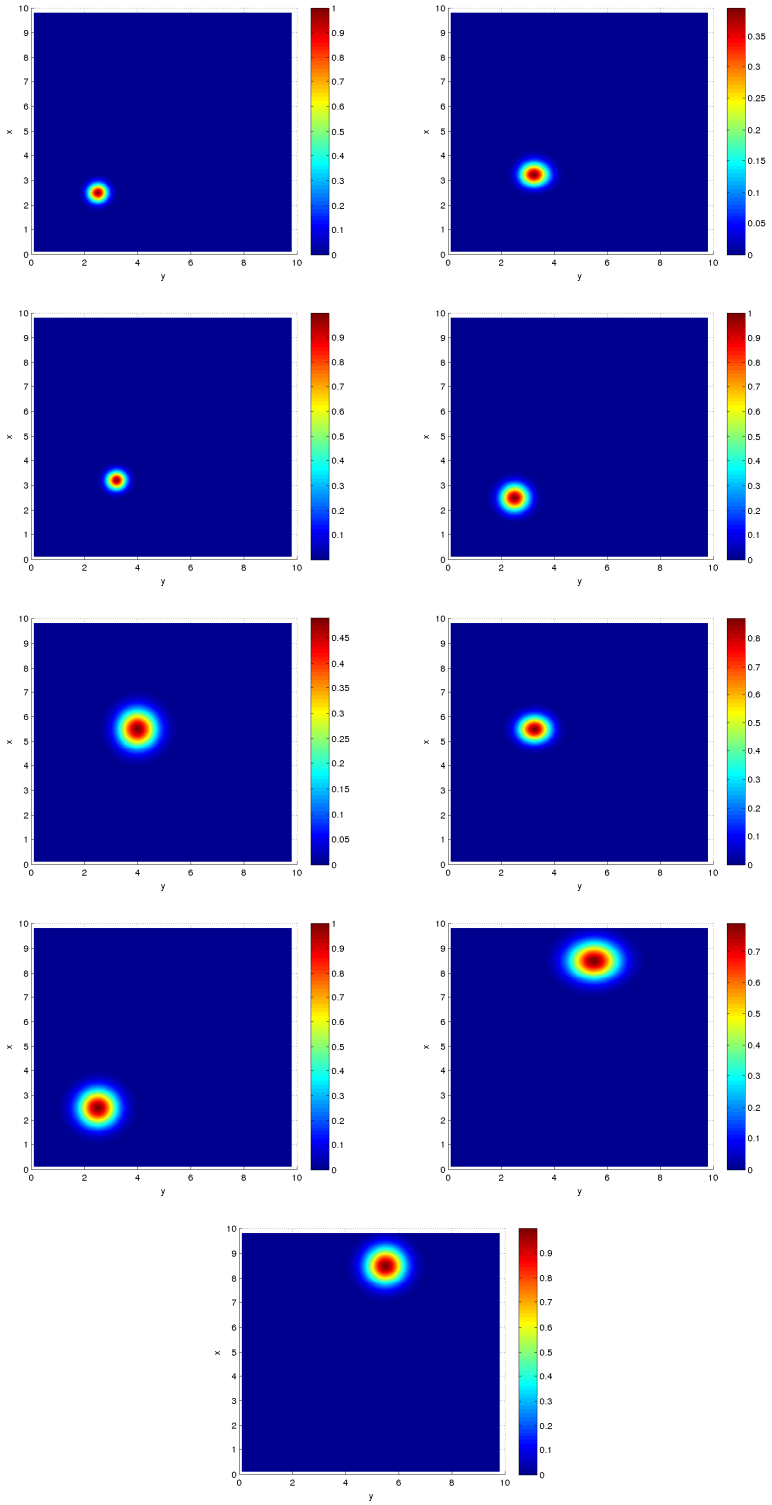


Fig. 9 Solution components (rows 1 to 3): initial data (1st column), CFL=1 solution (2nd column), and reservoir method solution (3rd column), at final time $T = 0.75$.

B Example for a rotation operator

A simple explicit example for a rotation operator can easily be constructed, inspired from [27]. We consider for $d = 3$, $m = 2^2$ the following system

$$\begin{cases} \partial_t u + A^{(\mathrm{x})}\partial_x u + A^{(\mathrm{y})}\partial_y u + A^{(\mathrm{z})}\partial_z u = 0, & (x,y,z,t) \in \mathbb{R}^3 \times (0,T), \\ u(\cdot,0) = u_0, & (x,y,z) \in \mathbb{R}^3 \end{cases} \tag{19}$$

with $A^{(\mathrm{x})} = S^{(\mathrm{x})}\Lambda^{(\mathrm{x})}S^{(\mathrm{x}T)}$ (resp. $A^{(\mathrm{y})} = S^{(\mathrm{y})}\Lambda^{(\mathrm{y})}S^{(\mathrm{y})T}$, $A^{(\mathrm{z})} = S^{(\mathrm{z})}\Lambda^{(\mathrm{z})}S^{(\mathrm{z})T}$), where $S^{(\gamma)}$ with $\gamma = x, y, z$, are defined by

$$S^{(\gamma)} = \frac{1}{\sqrt{2}}\begin{pmatrix} \mathrm{I}_2 & \sigma_\gamma \\ \sigma_\gamma & -\mathrm{I}_2 \end{pmatrix} \tag{20}$$

and where $\left\{\sigma_\gamma\right\}_{\gamma=x,y,z}$ are the Pauli matrices. From a quantum algorithm point of view, these rotations operators can be decomposed [27] as

$$S[\gamma] = \mathrm{C}(\sigma_\gamma)\mathrm{H} \otimes \mathrm{I}_2\mathrm{C}(\sigma_\gamma)$$

where $C(\sigma_\gamma)$ (resp. H) is σ_γ-controlled (resp. Hadamard) gate, see Fig. 1 in [27]. In this case the construction of the quantum circuit can directly be deduced from [27], thanks to a simple decomposition in elementary quantum gates of $S^{(\gamma)}$. More elaborated cases corresponding to more complex $S^{(\gamma)}$ can be considered as discussed above, but then necessitate more complex quantum circuits.

C Determining minimal resource requirements of the quantum algorithm with `Quipper`

`Quipper` is a `Haskell`-based embedded functional language whose purpose is to emulate *the implementation of quantum algorithms on realistic quantum computers* by providing explicit gate decompositions of quantum algorithms [31,32]. In particular, functions for transforming complex quantum circuits into elementary gates (Hadamard, CNOT, Clifford,...) are included in `Quipper`, along with many other functionalities allowing for circuit assembly and for resource requirement diagnosis. In this section, we will implement some of the algorithms presented above.

Throughout, it is assumed that all logical operations are implemented without error. In a real quantum device, the quantum system interacts with its environment, generating some noise and error in each operation. In this sense, the gate counts given below represent a lower bound estimates for the "true" algorithm, which may require error-correcting steps.

The algorithm for the solution of hyperbolic systems was formulated in the abstract Hilbert space of the quantum register. Therefore, it is independent of the computer architecture and thus, is amenable to any digital quantum devices. For example, quantum computers based on superconducting circuits [39,10,11], trapped ions [41] and cavity quantum electrodynamics [50], have been used with some success for other problems and could be used in principle to implement our algorithm. The main limitations however are i) the number of available qubits in current register and ii) the coherence time of these devices that restricts the number of logic quantum gates. State-of-the-art quantum computations on actual digital computers reach ≈ 1000 quantum logic gates on ≈ 9 qubits [11].

In all the examples considered in the following, minimal requirements are studied in order to assess the feasibility of simulations on these real quantum devices. In particular, we count the number of quantum gates (circuit depth) and the total number of qubits (circuit width) required to evolve the initial condition data to a given final time. Due to physical limitations in terms of the number of qubits and the coherence time, only systems with overly small meshes are investigated. As emphasized in [27], this may be enough for proof-of-principle calculations but is far from outperforming classical computations. Nevertheless, given the amount of effort and resources devoted to the development of these quantum devices, these numbers will likely be improved in the future.

C.1 One-dimensional hyperbolic system

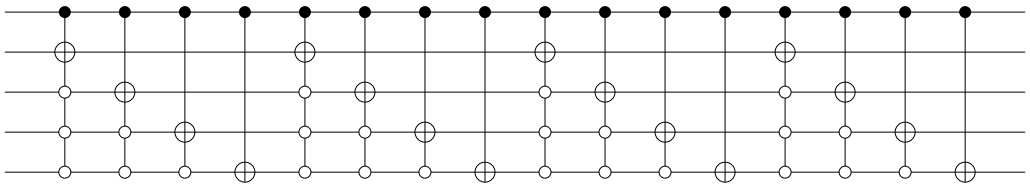
We consider a one-dimensional system with a computational domain as $[0,1] \times [0,T]$:

$$\partial_t u + \Lambda\partial_x u = 0, \qquad (x,t) \in [0,1] \times [0,T]$$

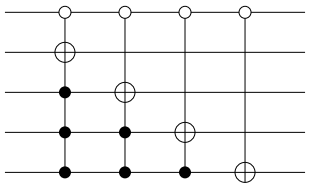
where $A = \Lambda$ is a diagonal matrix in $M_2(\mathbb{R})$, with eigenvalues $\lambda_1 = 1, \lambda_2 = -3$. If A were not diagonal, it would be necessary to add a transition operator at the beginning and the end of the circuit. For $T = 10^{-1}$ and $n_T = 40$ we can determine the sets

$$\begin{aligned} \mathcal{I}^{n_T} &= (2,2,2,1,2,2,2,1,2,2,2,1,2,2,2,1,2,2,2,1,2,2,2,1,\\ &\quad 2,2,2,1,2,2,2,1,2,2,2,1,2,2,2,1)\,, \\ \mathcal{S}^{n_T} &= (-,-,-,+,-,-,-,+,-,-,-,+,-,-,-,+,-,-,-,+,\\ &\quad -,-,-,+,-,-,-,+,-,-,-,+,-,-,-,+,)\,. \end{aligned}$$

We then implement the quantum algorithm described in Subsection 6.1, with `Quipper`. In this case, it is only necessary to implement the decrement and increment operators, thanks to the preliminarily established list $\mathcal{I}^{n_T}$ of characteristics to be updated. Following Table I from [27], we report the minimal requirements for simulating the one-dimensional system



(a)



(b)

Fig. 10 Circuit diagram with $N_x = 16$. (a) first 4 iterations of the quantum reservoir method and first component. (b) Second component.

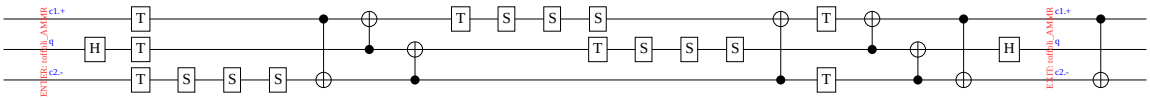


Fig. 11 Circuit diagram obtained from the gate decomposition with $N_x = 4$ in 1-D for one iteration.

described above, for proof-of-principle calculations. More specifically using `Quipper`, we can evaluate precisely the number of elementary gates necessary to implement the quantum algorithm. Say for respectively $n_x = 2$, $n_x = 4$, and $n_x = 8$, the circuit depth computed by `Quipper` is 780, 5520, 27960, and the circuit width is respectively 3, 7, 15. For instance for $n_x = 4$ that is $N_x = 16$, the respective number of Hadamard, Clifford, Toffoli, CNOT gates, is found to be 540, 1890, 1350, 1560. We report in Fig. 10 the quantum circuit for the first 4 iterations with $N_x = 16$. This quantum circuit has a width equal to 5, that is 4 qubits for labeling the coordinate space positions, and 1 qubit for the labeling of the component. The circuits which are represented corresponds to the first 4 time iterations. We notice that the circuit for the first component (top) has 4 times the same pattern (elementary circuit), corresponding to 4 translations from the left to the right, while the circuit for the second component (bottom) corresponds to only 1 iteration from the right to the left, for the same lapse of time. This is due to the fact that $\lambda_2 = -3 \times \lambda_1$, so that after 4 iterations, 4 translations to the right are applied to the first component, while the second component is only translated once to the left. The quantum circuit for 1 iteration is generated by `Quipper`.

As an illustration for the same test, but with $N_x = 4$, we also report in Fig. 11 the quantum circuit for the corresponding gate decomposition and which is still generated by `Quipper`. This time we only have 3 qubits, 1 for the component index, and 2 for the positions.

The case with the smallest circuit depth, for the lower number of lattice points ($n_x = 2, N_x = 4$), could possibly be implemented on actual device for a proof-of-principle calculation, as long as the initialization phase requires less than approximately 300 gates. However, the results obtained from the gate decomposition demonstrate that it would be a challenging task to perform quantum simulations of hyperbolic systems on actual quantum device with larger lattice size. Moreover, performing a relevant quantum simulation outperforming classical computation demands much improvement from both the coherence time and the number of qubits.

C.2 Three-dimensional hyperbolic system

In this subsection, we implement with `Quipper` the quantum version of the reservoir method for a three-dimensional linear hyperbolic system (19) with $\lambda_1^{(x)} = \lambda_2^{(x)} = -1$, $\lambda_3^{(x)} = \lambda_4^{(x)} = \sqrt{2}$, $\lambda_1^{(y)} = \lambda_2^{(y)} = -2$, $\lambda_3^{(y)} = \lambda_4^{(y)} = 2\sqrt{2}$, $\lambda_1^{(z)} = \lambda_2^{(z)} = -4$, $\lambda_3^{(z)} = \lambda_4^{(z)} = 4\sqrt{2}$. Notice that we have associated the following upper indices: (x) $\leftrightarrow$ (1), (y) $\leftrightarrow$ (2), (z) $\leftrightarrow$ (3). We then select $S^{(\gamma)}$ as in (20), where we recall that Pauli's matrices are defined by:

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad , \quad \sigma_y = \begin{bmatrix} 0 & -\mathtt{i} \\ \mathtt{i} & 0 \end{bmatrix} \quad \text{and} \quad \sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} . \tag{21}$$

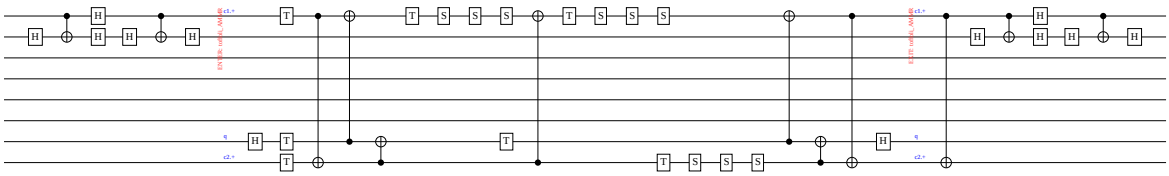


Fig. 12 Circuit diagram obtained from the gate decomposition with $N_x = 16$ in 3-D for the first iteration.

The motivation for considering such "simple" transition matrices is to design simple quantum circuit portion for diagonalization. As mentioned in Section 4.2.2, more complex transition unitary matrices could be considered, but would then require additional work for decomposing in elementary quantum gates. The quantum algorithm is applied from time 0 to 10^{-2}, with $\Delta x = 10^{-2}$, corresponding to $n_T = 30$ iterations. In addition, $\mathcal{I}^{n_T}$ and $\mathcal{S}^{n_T}$ are such that:

$$\begin{aligned}
\mathcal{I}^{n_T} = &((3,3),(4,3),(1,3),(2,3),(3,2),(3,3),(4,2),(4,3),(1,2),(1,3),\\
&(2,2),(2,3),(3,3),(4,3),(3,1),(3,2),(3,3),(4,1),(4,2),(4,3),\\
&(1,3),(2,3),(3,3),(4,3),(1,1),(1,2),(1,3),(2,1),(2,2),(2,3))\\
\mathcal{S}^{n_T} = &(+,+,-,-,+,+,+,+,-,-,\\
&-,-,+,+,+,+,+,+,+,+,\\
&-,-,+,+,-,-,-,-,-,-)
\end{aligned}$$

The corresponding time steps take the value $0.0177, 0.0073, 0.0104, 0.0030, 0.0177, 0.0043, \cdots$. We then encode $\mathcal{I}^{n_T}$ in the quantum algorithm in order to implement the quantum reservoir method. This list provides the directional characteristic field to update. Say for respectively $n_x = n_y = n_z = 2$, $n_x = n_y = n_z = 4$, and $n_x = n_y = n_z = 8$, the circuit depth computed by `Quipper` is 672, 3042, 14262, and the circuit width is respectively 8, 16, 32. For instance for $n_x = n_y = n_z = 4$ that is $N_x = N_y = N_z = 16$, the respective numbers of Hadamard, Clifford, Toffoli, CNOT gates, are found to be 400, 1037, 675, 840. We report in Fig. 12 the quantum circuit for the first iteration with $N_x = N_y = N_z = 4$, and a circuit width equal to 8. The quantum circuit which is generated by `Quipper` is much more complex (Toffoli, Hadamard, Clifford quantum gates, etc), as it also includes the changes of basis (rotations), and the translations.

The same conclusion as in the 1-D case can be reached from these results, i.e. a proof-of-principle calculation could possibly be performed with the smaller systems, but relevant calculations would require improvements in quantum technologies.