aerospace

MDPI

Article

An Intuitive Implementation of a Quantum Linear Solver for Electrostatic Propulsion for Spacecraft

Aeriana Venuta Narbonne [1,2], Cecile Malardier-Jugroot [1] and Manish Jugroot [2,*]

[1] Department of Chemistry and Chemical Engineering, Royal Military College of Canada, Kingston, ON K7K 7B4, Canada
[2] RMC Advanced Propulsion and Plasma Exploration Laboratory (RAPPEL Lab), Department of Mechanical and Aerospace Engineering, Royal Military College of Canada, Kingston, ON K7K 7B4, Canada
* Correspondence: jugroot-m@rmc.ca

Abstract

The need for a resilient and enhanced satellite constellation is crucial for the scientific and societal challenges of the future. Resilient satellites for Low Earth Orbit (LEO) can also be achieved through advanced propulsion systems, namely Electric Propulsion Systems (EPs). High-precision design tools would be extremely useful for understanding complex phenomena such as particle kinetics, transport, and acceleration in advanced propulsion systems for spacecraft. Quantum methods have the potential to increase the speed of simulation and, more interestingly, allow for further physical phenomena to be implemented in the numerical experiments. The purpose of this paper is to provide a step-by-step guide to applying quantum algorithms to practical engineering problems. Quantum computing addresses time complexity issues in these simulations to look at small-scale, higher-resolution problems.In our case, it is advanced satellite propulsion systems, and the current results are promising. EP systems that accelerate the propellant by an electric field fall under the electrostatic group; these thrusters include Hall Effect Thrusters (HETs) and Electrospray Thrusters (ESPs). The calculation of the electric potential, central to electrospray and Hall Effect Thrusters, is a stepping stone in the implementation of quantum computing and is the focus of the current paper. The expectation is to also provide guidance for other applications.

Keywords: quantum computing; electric propulsion for spacecraft; plasma modeling

1. Introduction

Resilient satellites for Low Earth Orbit (LEO) can be achieved through advanced propulsion systems, namely Electric Propulsion Systems (EPs). EP systems may be used for altitude and attitude control, essential for constellations, Deep Space exploration [1], and potentially for debris avoidance [2] and de-orbiting required for satellites within the LEO region 5 years after launch [3].

Since these high-efficiency propulsion systems are sometimes built at the small micro- and nano-scale, they require high-precision understanding to allow future development. Consequently, high-precision simulation design tools to model particle kinetics, transport, collisions and acceleration are necessary for the detailed understanding of underlying complex physics and design of new systems. Quantum methods have the potential of increasing the speed of simulation to allow for further physical phenomena to be implemented in numerical experiments. For example, the computational cost of 3D simulations, essential in capturing anomalous transport within Hall Effect Thrusters or droplet and ion

Check for updates

Academic Editors: Jian Liu and Yiheng Tong

Received: 3 September 2026
Revised: 14 October 2026
Accepted: 14 October 2026
Published: 22 October 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

formation in electrospray propulsion, could be substantially enhanced by implementing Quantum methods.

We present a simple hybrid linear solver and implementation of an HHL circuit using classical to speed up in an attempt to speed up the otherwise long process of encoding the operator and vector onto a quantum computer, and then a quantum circuit to solve for $x = A^{-1}b$ (looking specifically at how to encode the gradient to solve for the voltage in a quantum computing-readable format). We follow existing procedures for 1D [4], provide more documentation, and extend to 2D, and discuss how to include it into particle simulations of propulsion. Future work will add complexities to better simulate electric propulsion—and will include mixed boundary conditions, scaling to higher dimensions and resolution, a more accurate implementation of the charge density, addition of particles, and implentation on real quantum hardware. This paper uses the IBM quantum computing SDK, Qiskit, for easy implementation on a classical computer. This engineering step-by-step approach to quantum algorithms for modeling problems and numerical example of interest to electric propulsion systems for spacecraft should give a good start for researchers interested in advancing their simulations with quantum computing. The results are promising, even within the bounds of the Noisy Intermediate-Scale Quantum (NISQ) Era and working in a hybrid environment.

This is a hybrid algorithm, where the eigenvalues and eigenvectors are computed classically following [4]. Previous work [5] attempted to estimate the eigenvalues and eigenvectors with Quantum Phase Estimation (QPE) by mapping the mesh to a linear combination of Pauli matrices; however, the time complexity for this is 4^n, making higher level or multidimensional meshes unrealistic and failing to show a true quantum advantage. In this paper, we may compute the eigenvalues λ and and eiegenvectors v to speed up this bottleneck and then construct a quantum circuit to solve the LSP for the electric potential, $\phi \rightarrow |x\rangle = A^{-1}|b\rangle = \sum_{j=1}^{N} \lambda_j^{-1}|\mu_j\rangle\langle\mu_j||b\rangle$, which is detailed in Section 3.2.1.

1.1. Advanced Propulsion for Spacecraft: Electric Propulsion Fundamentals

Conventional rockets fall under chemical propulsion and create a high thrust through combustion, whereas electric propulsion has a low thrust but a very high exhaust velocity as illustrated in the rocket equation.

$$\Delta v = v_e \ln(\frac{m_0}{m_f}) \tag{1}$$

where Δv is the total change in velocity of the spacecraft relative to the exhaust velocity relative to the rocket (v_e), and the initial mass of the fuel (m_0), the final mass of the fuel (m_f). The rocket equation emphasizes that the exhaust velocity is no longer constrained by enthalpy of the propellant (chemical propulsion) but is theoretically unlimited and ions can reach extremely high values dependent only on the electric field (and power).

The efficiency of a rocket is defined as the fraction of power that is converted into work, i.e., propulsion,

$$\eta = \frac{\frac{1}{2}\dot{m}v_e^2}{P} \tag{2}$$

which may be rearranged and by the Rocket Equation,

$$\frac{2\eta}{v_e} = \frac{T}{P} \tag{3}$$

Observe that there is a trade-off between the achievable thrust and the available power which is governed by the exhaust velocity (specific impulse).

$$I_{sp} = \frac{v_e}{g_0} \tag{4}$$

EP accelerates a small amount of mass very quickly; hence, the v_e is high. We measure this as a specific impulse (I_{sp}), where g_0 is the gravitational constant. The high specific impulse is the main advantage of electric propulsion systems over more classical propulsion systems (such as chemical thrusters) and has the potential of unlocking enhanced space missions.

1.1.1. Operation of Electric Propulsion Systems

We will focus on two types of Electric Propulsion Systems (EPs) in the electrostatic category: Hall Effect Thrusters (HETs) and Electrospray Thrusters (ESPs). When classifying and simulating EP, we focus on the critical subsystems, how the propellant or charge particles are generated, and how the propellant is accelerated, since we need to know how to create a high specific impulse we need to know how to sustain emission over a long period of time. In the case of HET, how to maintain a large electric field in the bulk of the plasma so that positive ions can be extracted efficiently [6]; in the case of Electrosprays, how to stay in ion mode to achieve a high specific impulse [7].

In HETs, the subsystems are and an electric field and a magnetic field, where the magnetic field is either electromagnetic or permanent magnets [8]. Plasma cannot be accelerated by the magnetic field but can control the direction of the motion, while electric fields are responsible for particle acceleration [9], as shown in Figure 1. In ESP, the propellant is pulled out of the emitter via capillary forces. Once the electrostatic force exceeds the surface tension of the fluid, a Taylor cone is formed. The electric potential is created by charging the extractor and grounding the emitter. Taylor cones are produced when conductive liquids are placed under an external electric field [10]. We assume that the cone is formed by a balance between the electric field in the liquid E_{nl} and surface tension γ to estimate the shape of the cone, where $1/R_c$ represents the curvature at the apex. Once the Taylor cone forms, the Schottky barrier-lowering relation we may deduce the required local electric field E required to overcome the energy barrier (ΔG) and achieve ion emission [11] as depicted in Figure 2.

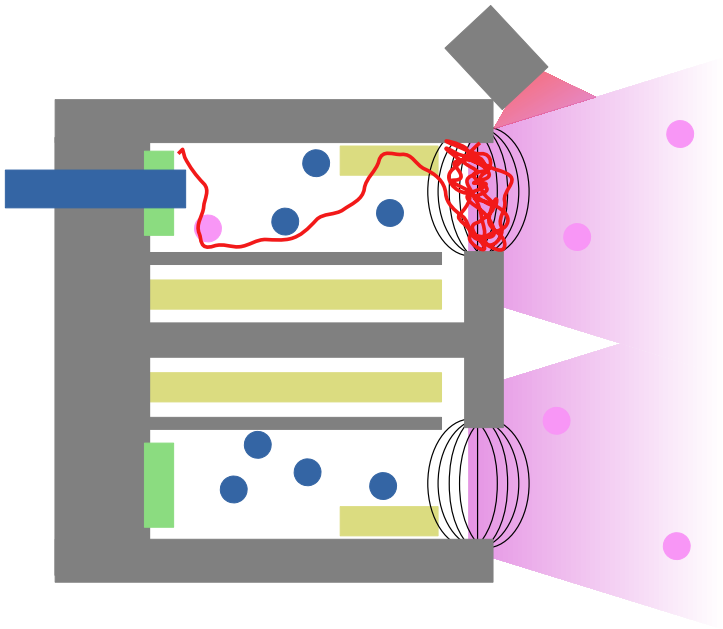


Figure 1. High-level schematic of particle motion in a Hall Effect Thruster (HET) showing neutral particles (blue), charged ions (purple) and electron trajectories (red). Magnetic field lines are represented at the channel exit to show the trapped electrons.

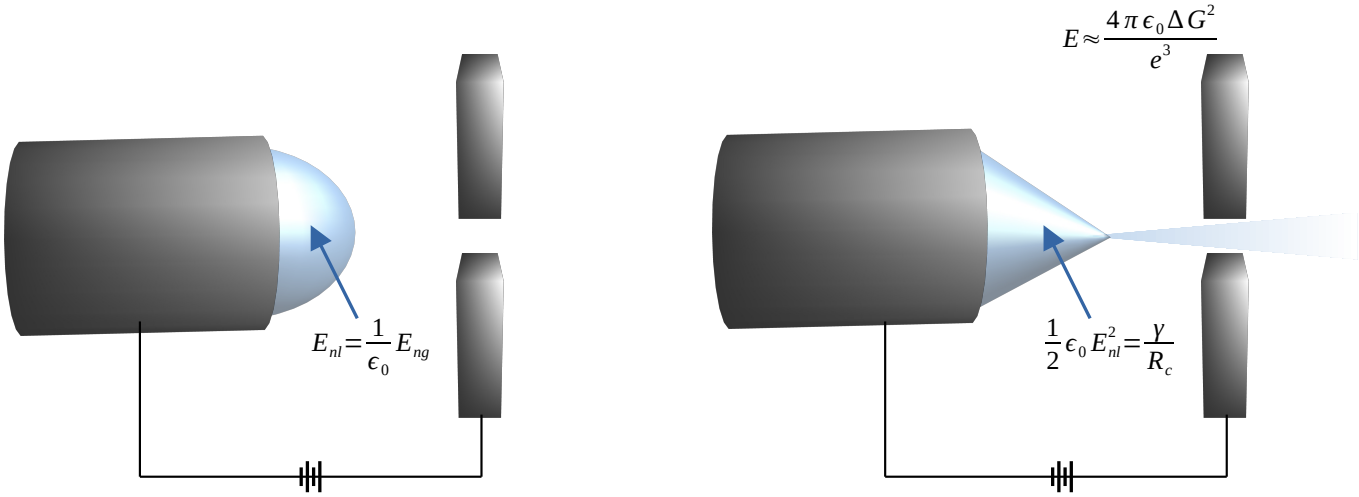


Figure 2. Schematic of Taylor formation—showing a small meniscus forming (**left**) until a Taylor cone is formed (**right**). Where electric field in the liquid is E_{nl}, surface tension γ, the permittivity ϵ, the charge e, $1/R_c$ represents the curvature at the apex, and the required local electric field E required to overcome the energy barrier ΔG.

This is an illustration of what will be paramount in further developing electric propulsion for spacecraft, and the future of simulations could be accelerated by implementing quantum computing. For instance, successful missions require solution of many challenging computational problems; and interestingly, quantum algorithms outperform classical algorithms [12].

1.1.2. Development Issues in Hall Effect Thrusters

The geometry of a Hall thruster can be cylindrical, for smaller thrusters, or coaxial for more powerful HETs, with an axial electric field applied from the anode to the cathode and a radial magnetic field [6] shown in Figure 1. It should be noted that the coaxial configuration offers higher ionization and efficiency compared to the cylindrical geometry. The basic principle of HETs is to generate an electric field locally in the plasma by a transverse magnetic field. This magnetic field reduces electron conductivity and these trapped electrons move azimuthally in the chamber, efficiently ionizing the plasma. The positive ions in the plasma are extracted by an applied electric field, and accelerated out of the chamber. The expected behaviour is that electrons are perfectly trapping. However perfect trapping is not realized, with more than expected electrons moving to the anode, leading to instabilities and imperfect ionization. As such, classical, linear simulations do not accurately model this anomalous electron movement. The Particle-in-Cell (PIC) method is a kinetic simulation deployed to model this particle motion by solving field equations and pushing the particles at each time step [13]. Poisson solvers are often 1D in the azimuthal direction [14]. Two dimensional Poisson solvers have been developed using Julia's 2D Poisson solver function based on the Gaussian elimination algorithm [15].

1.1.3. Development Issues in Electrospray Thrusters

The limiting lifetime factor for ESP is overspray onto the extractor electrode [16], which leads to emitter shortening. After the propellant accumulates on the grids from overspray, sufficiently strong electric fields lead to emission of the opposite polarity back to the emitter, called backspray [17]. An understanding of the propellant under the electric field is critical to simulate this behaviour and find design flaws to extend thruster lifetime.

1.1.4. Motivation: Simulation for Electric Propulsion Systems

Simulations are deployed in the design stage to build functional rockets by capturing the complex physics at the small scale, test out innovative ideas and to predict how spacecraft will evolve over time. There are still many challenges in the development of advanced propulsion systems and the design stage is critical. Furthermore, troubleshooting post-deployment is impossible: reinforcing the importance of bimodal and multimodal propulsion [7,18], where some design parameters could still be modified after launch by controlling thrust and specific impulse.

1.1.5. Motivation for Quantum Computing for Electric Propulsion

The idea proposed in the current paper is timely—as it is suggested to expand the global idea to advance the resilience of space missions. This could ultimately be achieved by implementing quantum computing for the advancement of the EP field for the acceleration of the expansion of humans in the Universe through future Deep-Space missions. To contribute, it is crucial that we must do both simultaneously: we must have simulations of physical phenomena at the local and global time and length scales on a quantum computer—before testing prototypes and improving predictions iteratively. The calculation of the electric potential, central to electrospray and Hall Effect Thrusters is a stepping stone in the implementation of quantum computing and is the focus of the current paper—where the calculation of the electric field is at the cornerstone of the EP simulation. The fundamentals of quantum computing are presented in the next sections.

2. Quantum Computing: Definitions

Before constructing the quantum algorithms, we must define elementary quantum computing concepts. This will form the building blocks of the solver as discussed in the subsequent subsections.

2.1. Definitions: Qubits, Quantum States, Gates, and State Evolution

A quantum computer uses quantum bits, *qubits*, which, when a gate is applied, evolve through a *quantum state*. Through a combination of superposition and entanglement, we achieve quantum speed up. The basis states are as follows.

Definition 1. *A quantum state $|\phi\rangle$ is a complex linear combination of basis states of the form*

$$|\phi\rangle = \alpha_1|00\rangle + \alpha_2|01\rangle + \alpha_3|10\rangle + \alpha_4|11\rangle$$

where:

$$|\alpha_1|^2 + |\alpha_2|^2 + |\alpha_3|^2 + |\alpha_4|^2 = 1$$

In other words, they are unit vectors. A non-basis state is said to be in a superposition.

U is written for a unitary operator, $U^\dagger$ for its conjugate transpose, and I as the usual identity.

Definition 2. *A quantum gate is a unitary matrix.*

$$UU^t = U^tU = I \tag{5}$$

A quantum state, initialized by convention at 0, is updated through a quantum gate. A collection of quantum gates forms a quantum circuit. A quantum state evolves on each qubit, which is updated by these quantum gates. A quantum circuit can hold more information as it scales by the tensor product, 2^n, doing calculations in parallel, and finding

problems that can be written in periodicity faster than traditional classical algorithms. This distinction between classical and quantum hardware and software is shown in Figure 3.

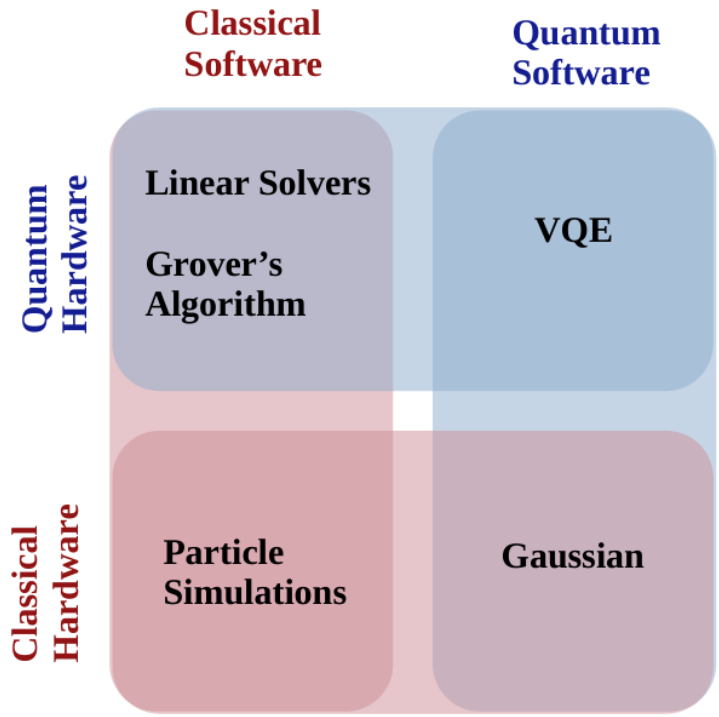


Figure 3. Quantum algorithms include anything run on a quantum circuit to simulate quantum systems. This paper focuses on linear solvers.

2.2. Qubit Counting

After measurement, the state collapses to 0 or 1, which outputs a binary string on a classical register. This subroutine, *qubit counting*, is the mapping from binary fractions to the quotients. The number of qubits is proportional to the resolution.

$$\begin{aligned} \hat{\phi} &= \sum_{m} \frac{\phi_k}{2^k} + \epsilon \\ &= 0.\phi_0\phi_k \dots \phi_m \end{aligned} \tag{6}$$

In Figure 4 we give an example of a binary mapping for eigenvalues ($\lambda_1 = 1, \lambda_2 = 2$) by selecting a time evolution ($t = \frac{\pi}{2}$) where ratios ($\frac{1}{4} : \frac{2}{4} = 1 : 2$) are preserved with minimum number of (2) qubits ($|\psi\rangle = \alpha_0|00\rangle + \alpha_1|01\rangle + \alpha_2|10\rangle + \alpha_3|11\rangle$) which may be written as binary fractions ($\lambda_1 \to \hat{\phi}_1 = 01$ and $\lambda_2 \to \hat{\phi}_2 = 10$).

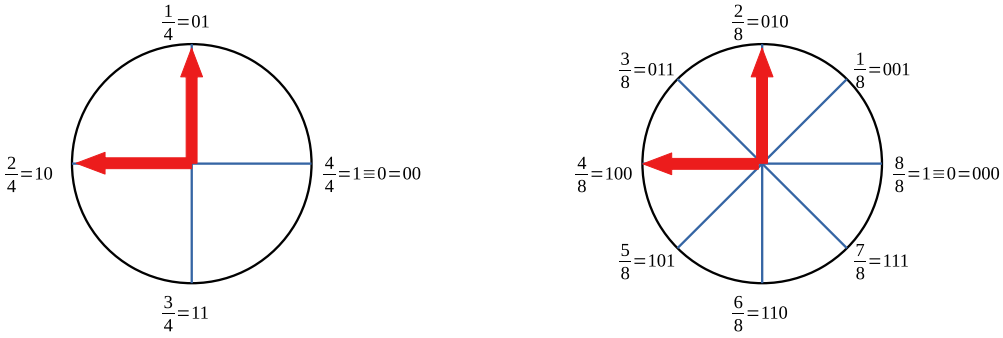


Figure 4. Example of a binary map on 2 qubits (**left**) and 3 qubits (**right**). Number of qubits increases the accuracy.

2.3. Quantum Algorithms

A quantum algorithm is often implemented using hybrid classical-quantum hardware with some subset of the calculations preformed on a quantum circuit [12]. NASA has been looking into the use of quantum algorithms [19]. A Variational Quantum Eigensolver may be used to simulate a quantum system [20], whereas quantum annealing provides an alternative approach to approximate optimization [21].

2.4. Quantum Fourier Transforms

Recall the Discrete Fourier Transform (DFT) [22] is an $N \times N$ matrix which takes an input vector x and outputs vector y,

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{\frac{2\pi i j k}{N}} \tag{7}$$

where y_k are the coefficients for each i, j in the matrix. The DFT allows for sampling of the wave function at evenly spaced intervals; hence, it works best with periodic problems. Similarly, the Quantum Fourier Transform preforms a basis change from the computational basis to the frequency domain. It is, essentially, a Discrete Fourier Transform on qubits,

$$\mathrm{QFT_N}|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{\frac{2\pi i \phi \psi}{N}} |\phi\rangle \tag{8}$$

where N is the dimension of the Hilbert space and can be simulated on $\log_2(N)$ qubits. Observe that we have written this in terms of roots of unity, which is the computation the circuit applies to the quantum state.

$$\mathrm{QFT_N}|\psi\rangle = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^1 & \omega^2 & \omega^3 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{4(N-1)} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega^{N-1} & \omega^2 & \omega^3 & \cdots & \omega^{(N-1)(N-1)} \end{bmatrix} \begin{bmatrix} \phi_0 \\ \phi_1 \\ \phi_2 \\ \phi_3 \\ \vdots \\ \phi_{N-1} \end{bmatrix} \tag{9}$$

Conversely, the inverse Quantum Fourier Transform, iQFT, transforms from the frequency domain to the spatial domain,

$$\mathrm{iQFT_N}|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{\frac{-2\pi i \phi \psi}{N}} |\phi\rangle \tag{10}$$

Observe that the higher N, the more divisions in the fraction. Thus, the more qubits allotted for qubit counting, the more divisions of the binary fraction, the more precise the end result, the accuracy of which scales at 2^n. So, QFT on two qubits is represented as a 4×4 unitary operator [23], QFT on three qubits is an 8×8 unitary operator, and we may calculate for four qubits the 16×16, and so on, as shown in Figure 5. These computations are calculated by a quantum circuit. QFT is expressed as a circuit by decomposing it into 1-qubit gates.

Where each j^{th} phase gate R_y may be calculated as follows,

$$R_Y = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{2\pi i}{2^k}} \end{bmatrix} \tag{11}$$

where $R_2 = S$ and $R_2 = T$. This allows us to evenly slice the complex unit circle, i.e., about the Z axis. Observe that the computation of the circuit and the matrix output the same result.

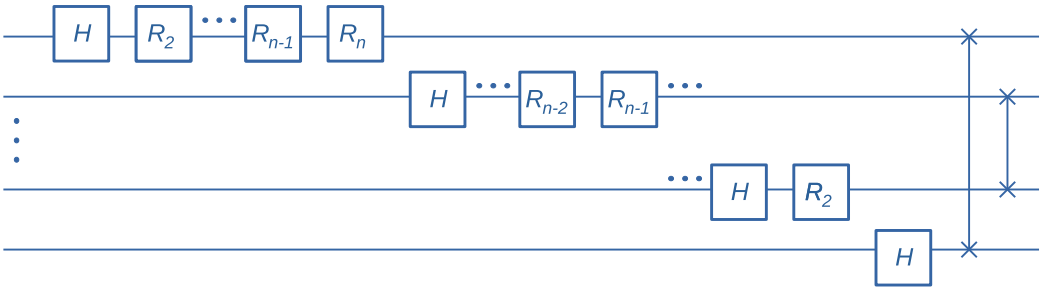


Figure 5. Generalized Quantum Fourier Transform circuit decomposed into single qubit gates.

The QFT and iQFT subroutines both provide the foundations for higher-level quantum algorithms. Since a quantum operates in the frequency domain that collapses upon measurement, we may speed up computation in the frequency domain; however, we require a way to input and output readable, classical data.

2.5. Quantum Phase Estimation

Given the Unitary Operator, U, and an eigenstate of U, $|\psi\rangle$, Quantum Phase Estimation (QPE) kicksback an eigenphase of U.

$$U|\psi\rangle = e^{i\theta}|\psi\rangle \tag{12}$$

By using QFT, we are able to more quickly calculate the eigenvalues.

3. Theory: Numerical Formalisms

Now that we have reviewed the required background on electric propulsion and quantum computing, we discuss more specifically where in both fields we intend to link the two. Specifically, we are focusing on simulation of electric propulsion by building a particle simulation, for now focusing on the electric potential solver. On the quantum side, we focus on how to solve this equation on a quantum computer.

3.1. Modeling Electric Propulsion: Constraints

Simulations of electric propulsion focus on how the propellant evolves over time under the influence of an electric field. These simulations are generally classified into two groups: Fluid [24] (propellant as a continuum fluid) and Kinetic [13] (propellant as individual particles). In vacuum space conditions the distance between particles is relatively large, implying that a kinetic model on the ions would give the most accurate estimation. If the electron temperature is high enough, we may still assume that the electrons behave like a fluid.

The cells give a grid in which we model particles as outlined in Figure 6, which may be represented as a mesh, i.e., as a matrix. This is the link from continuous modeling to a discrete space, a well defined process from Finite Element Methods (FEMs).

The Monte Carlo Particle-in-Cell Approach (MC-PIC) was first introduced by Birdsall [13]. Several calculations are performed in a single time step, Δt, in Figure 7: the force on the particles from the Lorentz Force, the particle charges, and the electric field by the Poisson Equation.

In this paper, we will focus on the Poisson Equation to first build a quantum linear solver that can interact with a more complex simulation. In future work, we will apply the PIC methods to determine how a hybrid particle simulation compares to existing classical simulations.

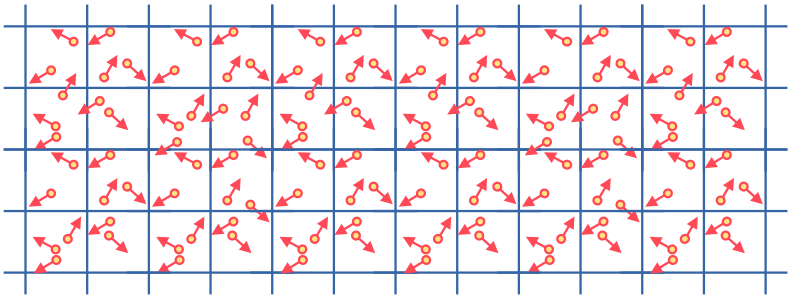


Figure 6. Particle-in-Cell (PIC) model uses a grid structure to compute fields $(\vec{E}, \vec{B})$ on fixed points in space.

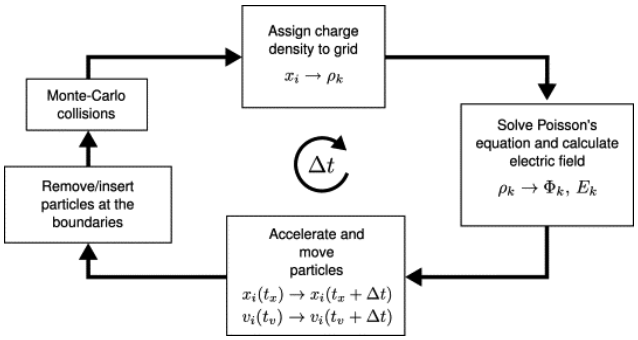


Figure 7. Particle-in-Cell timestep calculations for the plasma (from [13]).

A discretized thruster can be assumed to be a rectangular domain for preliminary validation of the concepts. In this paper, we focus on 1D (a line) and 2D (a rectangle). In 2D a more accurate representation is the mixed Neumann–Dirichlet boundary conditions [25]. In 1D, we may use Dirichlet boundary conditions (Figure 8) with 1 side grounded and the other end applied with a voltage. For this paper, we will assume only Dirichlet conditions in order to preserve the Hermitian property; more work on alternative boundary conditions will be carried out in future work.

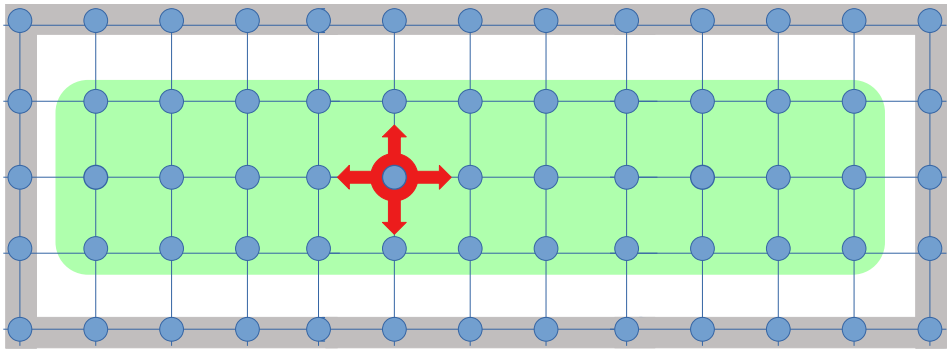


Figure 8. 2D electric field schematic with Dirichlet boundary conditions. Interior nodes determined by the neighbouring points.

3.2. Quantum Linear Solver for Electric Field

To construct an algorithms which models electric propulsion on a quantum computer, we must find an area for improvement (time complexity for high dimension high resolution), we may then find an existing classical algorithm (MC-PIC) and an existing bottle neck (electric field solver). From here, we need to provide a way of mapping this problem to a quantum computer. In this section, we review an existing linear solver and how we may link the existing classical electric propulsion simulation problem to a quantum computing solution.

3.2.1. Harrow–Hassidim–Lloyd (HHL) Algorithm

The Harrow–Hassidim–Lloyd (HHL) Algorithm is a linear solver [26]. Given a Linear System Problem ($Ax = b$), x is found by inverting A and applying it to b ($x = A^{-1}b$). We select an A such that A is Hermitian, and hence diagonalize (i.e., it may be written as a sum of an inner product of its eigenvectors μ_j, V weighted by its eigenvalues λ_j, Λ).

$$\begin{aligned} A &= V\Lambda V^{\dagger} \\ &= \sum_{j=1}^{N} \lambda_j |\mu_j\rangle\langle\mu_j| \end{aligned} \tag{13}$$

As a consequence of the Spectral Theorem, the inverse of A is determined solely by inverting the eigenvalues.

$$A^{-1} = \sum_{j=1}^{N} \lambda_j^{-1} |\mu_j\rangle\langle\mu_j| \tag{14}$$

Therefore to solve a linear system problem, we apply b to A^{-1}.

$$x = \sum_{j=1}^{N} \lambda_j^{-1} |\mu_j\rangle\langle\mu_j|b\rangle \tag{15}$$

This equation is then solved on a quantum computer by building quantum gates which are proportional to the RHS. These may be viewed as the four subroutines outlined in Figure 9. The HHL circuit depicts four subroutines, state preparation on b, QPE (Hadamards, Unitaries, iQFT), Rotational Y, iQPE (QFT, Inverse Unitaries, Hadamards). We measure the system register to solve for x and the ancilla for a post-selection flag for which outputs to keep.

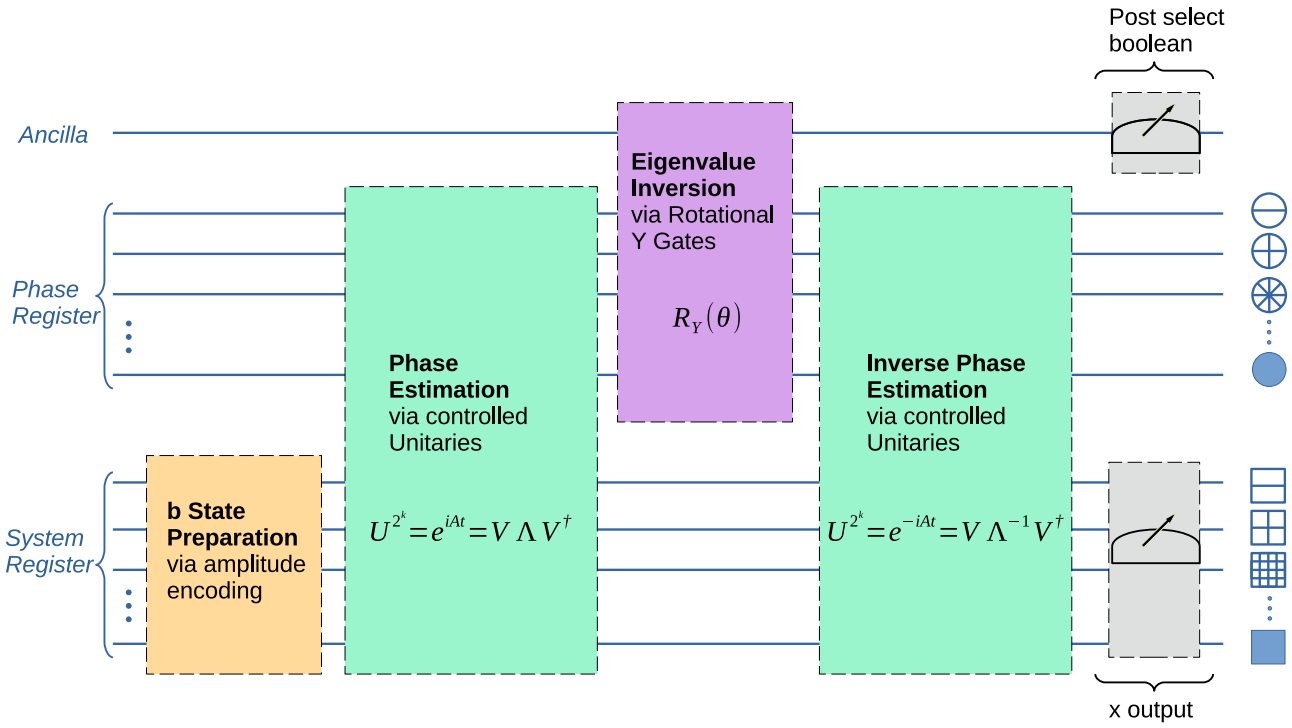


Figure 9. General multiqubit HHL circuit to solve the LSP ($x = A^{-1}b$) showing the two different resolutions of both eigenvalues by qubit counting and mesh by increasing the system register.

3.2.2. Mapping the Poisson Equation to a Quantum Problem: Mesh Derivation for 1D and 2D with Boundary Conditions

The selected equation to model the electric field of these electrostatic thrusters is the Poisson Equation,

$$-\nabla^2\Phi = \frac{\rho}{\epsilon_0} \tag{16}$$

where ∇^2 is the variation over the space, Φ is the electric potential, ρ is the charge density, and ϵ_0 is the permittivity of the vacuum. Since qubits are discrete and we wish to use HHL, first the Poisson equation must be written as a Linear System Problem (LSP).

$$-\nabla^2\Phi = \frac{\rho}{\epsilon_0} \rightarrow Ax = b \tag{17}$$

where $A \in \mathbb{R}^{n\times n}$, and $x, b \in \mathbb{R}^{n\times 1}$. To convert this continuous partial differential equation into a linear equation, we use the finite differences—centered differences approximation:

$$-\Delta = -\nabla^2 = -\frac{d^2v(x)}{d(x)} = \frac{-v_{i-1} + 2v_i - v_{i+1}}{h^2} \tag{18}$$

From this formulation, we may construct the *discrete Laplacian in 1D with Dirichlet boundary conditions*. The minimum size for this matrix is 2×2.

$$L_{1D}^N = \begin{bmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{bmatrix} \tag{19}$$

Recall that we are using the HHL algorithm, so only the eigenspectrum and eigenfunctions have to be determined. These are as follows [27]:

$$\begin{aligned} \{\lambda_j\} &= 2\Big(1 - \cos\Big(\frac{j\pi}{N+1}\Big)\Big) \\ \{\mu_j\} &= \sqrt{\frac{2}{N+1}}\sin\Big(\frac{jk\pi}{N+1}\Big) \end{aligned} \tag{20}$$

By extending to 2D with a 5-point stencil method, the *discrete Laplacian in 2D* of the form is constructed,

$$L_{2D}^N = \begin{bmatrix} T & I & & \\ I & T & I & \\ & \ddots & \ddots & \ddots \\ & & I & T \end{bmatrix} \quad \text{where } T_M = \begin{bmatrix} 4 & -1 & & \\ -1 & 4 & -1 & \\ & \ddots & \ddots & \ddots \\ & & -1 & 4 \end{bmatrix} \tag{21}$$

Definition 3. *To increase the dimension of the Poisson Equation for $n \geq 2$, we increase the number of terms as follows,*

$$L_N^{nD} = \underbrace{T_M \underbrace{\otimes I_M \otimes \cdots \otimes I_M}_{n-1 \text{ times}} + I_M \otimes T_M \otimes \cdots \otimes I + \cdots + I_M \otimes I_M \otimes \cdots \otimes T_M}_{n \text{ times}} \tag{22}$$

Since a Hermitian matrix is required, we construct a square matrix of size $N = 2^n$ represents the resolution. The minimum size for the 2D case is $N = 4$, which may be

simulated on $\log_2(4) = 2$ qubits. In fact, we may scale $N = 8$, which may be simulated on a minimum of $\log_2(8) = 3$ system qubits and results in a 2×4 rectangular domain.

$$L_8^{2D} = \left[\begin{array}{ccc|ccc|cc} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 \\ \hline -1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 \\ 0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 \\ \hline 0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \end{array}\right] \tag{23}$$

The resolution of the space can be increased by increasing the size of the above matrix (i.e., increasing the number of entries N). Observe that we may have a lower dimension with a higher resolution mesh vs a high dimension low resolution mesh on the same number of qubits.

The Discrete Laplacian in 2D of N resolution is

$$L_N^{2D} = T_M \otimes I_M + I_M \otimes T_M \tag{24}$$

Hence the eigenvalues and eigenvectors for 2D are

$$\begin{aligned} \{\lambda_{i,j}\} &= \lambda_i + \lambda_j \\ &= 2\Big(1 - \cos\Big(\frac{i\pi}{N_x + 1}\Big)\Big) + 2\Big(1 - \cos\Big(\frac{j\pi}{N_y + 1}\Big)\Big) \\ \{\mu_{i,j}\} &= \mu_i \otimes \mu_j \\ &= \sin\Big(\frac{ik\pi}{N_x + 1}\Big) \sin\Big(\frac{jk\pi}{N_y + 1}\Big) \end{aligned} \tag{25}$$

As we extend using this matrix with the Kronecker product, we keep the Hermitian property; hence, the eigenvalues are real and the eigenvectors are orthonormal. This will allow us to construct quantum gates.

$$\begin{aligned} L_N^{2D} &= T_M \otimes I_M + I_M \otimes T_M \\ &= (V \otimes V) \cdot (\Lambda \otimes I + I \otimes \Lambda) \cdot (V \otimes V)^\dagger \end{aligned} \tag{26}$$

Theorem 1. *L_N^{nD} is Hermitian $\forall n, N \in \mathbb{N}$.*

Proof. Given that T_M and I_M are Hermitian ($\in \mathbb{R}^{M \times M}$ symmetric with orthonormal eigenvectors and real eigenvalues). Since the Kronecker product is well-defined, i.e.,

$$\begin{aligned} L_N^{nD\dagger} &= \left(T_M \otimes I_M \otimes \cdots \otimes I_M + I_M \otimes T_M \otimes \cdots \otimes I_M + \cdots + I_M \otimes I_M \otimes \cdots \otimes T_M\right)^\dagger \\ &= T_M^\dagger \otimes I_M^\dagger \otimes \cdots \otimes I_M^\dagger + I_M^\dagger \otimes T_M^\dagger \otimes \cdots \otimes I_M^\dagger + \cdots + I_M^\dagger \otimes I_M^\dagger \otimes \cdots \otimes T_M^\dagger \\ &= T_M \otimes I_M \otimes \cdots \otimes I_M + I_M \otimes T_M \otimes \cdots \otimes I_M + \cdots + I_M \otimes I_M \otimes \cdots \otimes T_M \\ &= L_N^{nD} \end{aligned}$$

Therefore the resulting operator is also Hermitian. □

As has been shown the matrix is Hermitian regardless of the dimension [28]; it is now possible to implement the controlled unitary step of HHL, the critical link between this application and the quantum solver. We now apply the HHL algorithm for Electric Propulsion.

4. The Proposed Software

The HHL circuit may be separated into four main subroutines. First, b is encoded using *amplitude encoding* (Equation (27)); next, A is encoded using *Quantum Phase Estimation* and decomposing A in terms of its eigenvectors weighed by the corresponding eigenvalue (Equation (13)), then the matrix is inverted (the eigenvalues are inverted) by a series of *Rotational Y* gates, and finally the solution is returned from the Fourier basis to the computational basis via *Inverse Quantum Phase Estimation*.

We propose a Hybrid Classical-Quantum (Figure 10) wherein classical parameters are defined based on the model, eigenvalues and eigenvectors are prepared classically, and then Poisson's Equation is solved via a Quantum Circuit (HHL), as shown in Figure 9. First, Python is used to compute the classical preparations such as the eigenvalues and eigenvectors. Next, Qiskit is used to construct the quantum circuit by making diagonalized matrices of the eigenvalues and eigenvectors, which are unitary matrices and can be constructed as quantum gates on the circuit. Observe that A and b are encoded on the system register, hence as the resolution increases so does the qubit requirement.

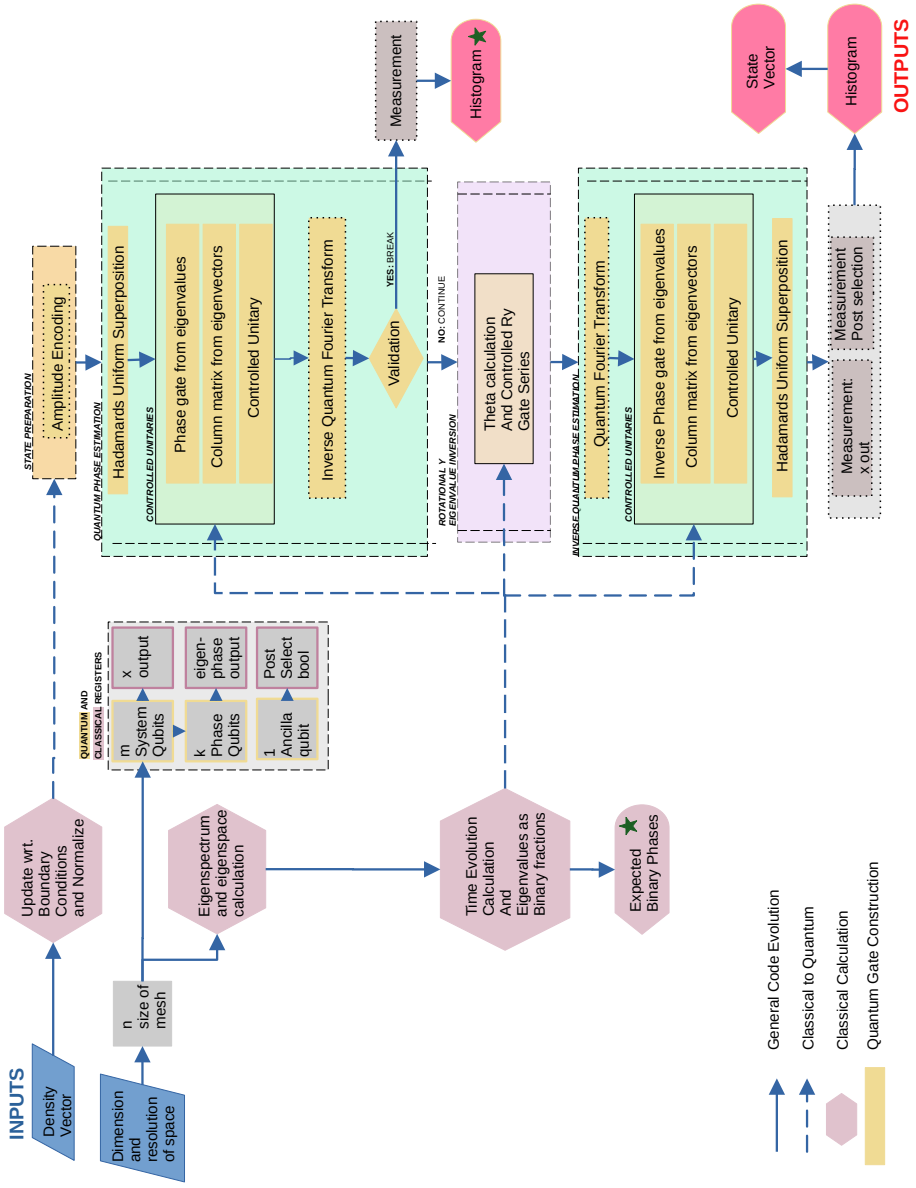


Figure 10. Hybrid classical-quantum HHL systems diagram.

4.1. Amplitude Encoding

The $\frac{\rho}{\epsilon_0}$ term is $b \in \mathbb{R}^N$ in the LSP. Dependent on the boundary conditions (if Neumann), this term may need to be updated on the endpoints. After b is determined by the problem, it is normalized $\hat{b} = \frac{b}{\|b\|}$. This is shown by Algorithm 1. Finally encoded using amplitude encoding,

$$|b\rangle = \sum_{i}^{N} \hat{b}_i |i\rangle \tag{27}$$

where $|i\rangle$ is the computational basis (for example $|0\rangle = |00\rangle, \ldots, |3\rangle = |11\rangle$ and so on) and the circuit representing this equation may be built directly via Qiskit.

Algorithm 1 Amplitude Encoding

Define vector
for $i = 1$ to N **do**
    read in b_i
end for
Calculate norm $\|b\| = \sum_i^N b_i$
Normalize
for $i = 1$ to N **do**
    $\hat{b_i} = \frac{b_i}{\|b\|}$
end for
Computational Basis: Qiskit Amplitude Encoding
    return $b = \sum_i \hat{b_i}|i\rangle$ Quantum Circuit — **State Preparation**

4.2. Hamiltonian Simulation

The critical subroutine of Quantum Phase Estimation is the time evolution operator, e^{iAt}, the Hamiltonian simulation. In the quantum circuit, this is encoded as a series of controlled unitaries via the Suzuki-Trotterization. The base gate is derived from the phase gate,

$$P(\varphi) = R_z(\varphi) = \begin{bmatrix} e^{-i\varphi_1} & 0 \\ 0 & e^{-i\varphi_2} \end{bmatrix} \tag{28}$$

We may scale these gates for higher dimensions or higher resolutions of the mesh by adding additional qubits to the system register and increasing the size of the phase gate applied to the entire system register as follows.

$$P_N(\varphi) = \begin{bmatrix} e^{-i\varphi_1} & 0 & \cdots & 0 \\ 0 & e^{-i\varphi_2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & e^{-i\varphi_N} \end{bmatrix} \tag{29}$$

4.2.1. Time Evolution

The time evolution, t, is selected to ensure a 1-1 mapping which preserves ratios.

$$\begin{aligned} T &= \lceil \log_2(\lambda_{MAX}) \rceil \\ t &= \frac{\pi}{T} \end{aligned} \tag{30}$$

where each $\varphi_j = \lambda_j t$, which may be calculated using $e^{i\varphi_j 2^k} = \cos(\varphi_j 2^k) + i\sin(\varphi_j 2^k)$. Observe that this results in $2T$ roots of unity, i.e., evenly portioned fractions $Z_{2T} = 1 = e^{2ki\pi/2T}$ where $k = 0, 1, 2, \ldots, 2T-1$.

4.2.2. Controlled Unitaries

Following the process outlined in HHL step by step, after building the Λ, we construct the unitary with the eigenvectors.

$$A \rightarrow U^{2^k} = V\Lambda^{2^k}V^{\dagger} \tag{31}$$

This is shown by Algorithm 2.

Algorithm 2 Controlled Unitaries

Calculate eigenvalues and eigenvectors
for $i = 1$ to N **do**
    Calculate the eigenvectors $\{\mu\}$ by Equation (20) or Equation (25)
    Calculate the eigenvalues $\{\lambda\}$ by Equation (20) or Equation (25)
end for
Column Eigenvector Matrix V and $V^{\dagger}$
for $i = 1$ to N **do**
    $V(i) = \hat{\mu}_i$
end for
Time evolution t
if Small Case $(N < 2^4)$ **then**
    choose $(t = \frac{\pi}{M}$ for some $M \in \mathbb{N})$
else if Large Case $(N > 2^4)$ **then**
    $t = \frac{\pi}{T}$ where $T = \lceil \log_2(\lambda_{MAX}) \rceil$ (rounded up to the nearest 2^n)
end if
Eigenvalue Binary Fraction Mapping $\varphi_j = \lambda_j t$
Diagonal Eigenvalues Matrix Λ^{2^k}
for $i = 1$ to N **do**
    $e^{-i\varphi_1} = \cos(\lambda_j t 2^k) + i\sin(\lambda_j t 2^k)$
end for
Unitary Matrix $U^{2^k} = V\Lambda^{2^k}V^{\dagger}$
Quantum Gate
    return k^{th} Quantum Gate

For small cases, we can easily see that the choice of t works nicely by calculating the mapping and making sure the ratios remain and after $2^4 = 16$, these by-hand calculations are unrealistic. For larger cases, we take the $t = \pi/\lambda_{max}$ ceiling rounded to the nearest 2^n. Each of these unitaries is wrapped into a controlled unitary where the k^{th} unitary's control is applied to the k^{th} qubit of the phase register. This is the key subroutine, outlined in (Algorithm 3).

Algorithm 3 Quantum Phase Estimation

Uniform Superposition of basis states on the phase register
for $i = 0$ to $k - 1$ **do**
    Apply Hadamard gate H to each qubit on the phase register
end for
Controlled Unitaries
for $i = 0$ to $k - 1$ **do**
    Algorithm Controlled Unitaries
    Attach that quantum gate to the full system register of size $\log_2 N = m$ and the control the k^{th} qubit of the phase register
end for
Inverse Quantum Fourier
    return Quantum Phase Estimation circuit

For inverse Quantum Phase Estimation, we reverse the order and apply e^{-iAt}, which may be calculated by using the same function as e^{iAt} but reading in $(-1)t$.

4.3. Eigenvalue Inversions

We use a series of R_y gates which are applied to the ancilla qubit and controlled by phase qubits.

$$R_Y(\theta_k) = \begin{bmatrix} \cos(\frac{\theta_k}{2}) & -\sin(\frac{\theta_k}{2}) \\ \sin(\frac{\theta_k}{2}) & \cos(\frac{\theta_k}{2}) \end{bmatrix} \tag{32}$$

Each θ_k is found as a function of individual eigenvalues $\theta_k = 2\arcsin(\frac{C}{\tilde{\lambda}_k})$, C is selected to be in bounds for the arcsin function, shown in Algorithm 4.

Algorithm 4 Eigenvalue Inversions

1: **for** $i = 0$ to $k - 1$ **do**
2: φ eigenvalue's binary fraction mapping
3: Find each angle $\theta_k = 2\arcsin(\frac{C}{\tilde{\lambda}_k})$, set $C = 1$ for small cases, or $C = \frac{1}{\lambda_{MAX}}$
4: Calculate $R_Y(\theta_k)$
5: attach $R_Y(\theta_k)$ to the ancilla qubit and control the th k^{th} qubit of the phase register
6: **end for**
return Quantum Circuit

φ, the eigenvalue's binary fraction mapping, may be reused from the previous algorithm. We set $k = N$.

5. Results and Discussions

Circuits were built using Python 3.13.9 (Jupyter notebook) and Qiskit 2.0.0 (AerSimulator). The circuits and histograms are discussed in this session with an emphasis on mapping binary fractions to a quotient.

5.1. Reading Quantum Results

A quantum computer is a probabilistic machine. Speed up is achieved by entangling many possible outputs which collapse into a single option upon measurement. As such, we will run the circuit for some s number of shots and compile all outputs into a histogram. We will see quantum noise, i.e., incorrect values as outputs. If the circuit is constructed correctly, the most probable outputs will be the correct outputs, so we may discard less probable outputs. Since we are working with a LSP of some dimension n, we take the n most probable outputs as x.

5.2. Quantum Phase Estimation (Validation Checkpoint)

The foundational subroutine in HHL is Quantum Phase Estimation (QPE), where the eigenvalues are estimated and entangled with the corresponding eigenvectors. The intermediary verification step ensures the accuracy of the solver by validating that the eigenphases map to the expected eigenvalues. We may calculate eigenvalues classically and compare with the quantum output of eigenphases. QPE requires a superposition of the eigenvector as an input which, for this example, works nicely to be either $|0\rangle$ or $|1\rangle$. Taking $A = L_2^{1D}$, we encode A by the series of unitaries using the Algorithm 3 outlined above. By convention, qubits are initialized $|0\rangle$, so we may construct $b = |1\rangle$ by applying an X gate to the system register or taking $b = |0\rangle$. To make the code reusable, there is a black box for the b preparation regardless of the value.

5.2.1. QPE with 2 Phase Qubits

The smallest case of $m = 1$ system qubit may be used to build a mesh of dimension $n = 2^1$. The only mesh that can effectively model a system with a 2×2 matrix is L^{1D},

i.e., 2 end points. Though unlikely to have any applications in plasma modeling, this small case of a circuit shown in Figure 11 validates the procedure of construction of e^{iAt} results in a unitary matrix. The circuit is built with one system qubit representing a 2×2 mesh and a binary result of accuracy of two phase qubits, i.e., 4 binary fractions 00, 01, 10, 11. Outputs are compiled into a histogram shown in Figure 12, taken to be the best estimates of eigenvalues.

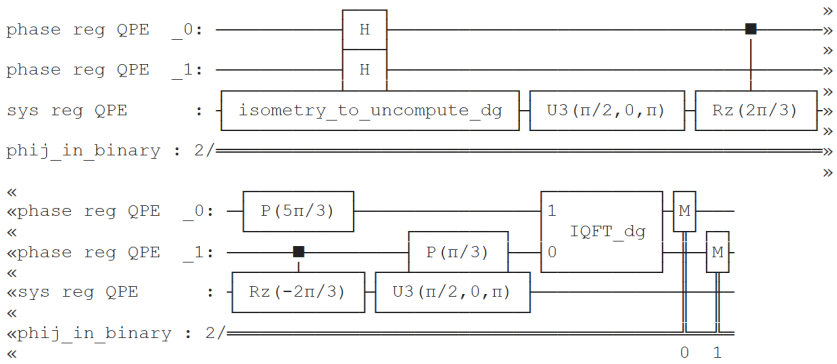


Figure 11. Quantum Phase Estimation circuit built with Qiskit 2.0.

Once the circuit is constructed, it is run for 4000 shots and results are compiled into a histrogram, Figure 12. $\lambda = \{1, 3\}$. Taking $t = \frac{2\pi}{3}$ with $k = 2$ phase qubits hence,

$$\lambda_1 = 1 \rightarrow \hat{\phi} = 1\frac{2\pi}{3 \times 2\pi} = 0.3 \equiv 0.6 \rightarrow 1$$

$$\lambda_2 = 3 \rightarrow \hat{\phi} = 3\frac{2\pi}{3 \times 2\pi} = 1 \equiv 0 \rightarrow 00$$

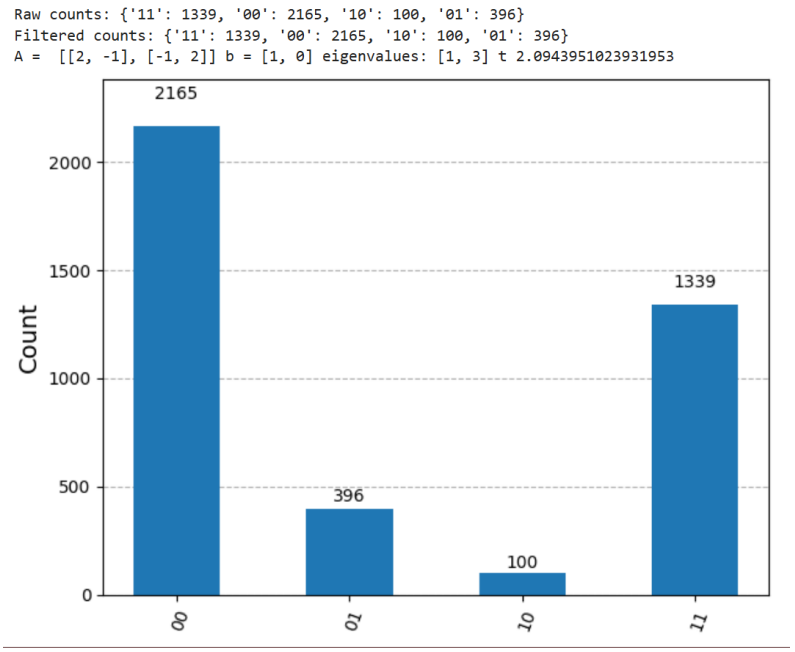


Figure 12. Histogram of the smallest case for the discrete Laplacian. Allows validation of the construction of the unitaries.

5.2.2. QPE with 5 Phase Qubits

Observing that this is not a perfect mapping, to gain a higher accuracy in the binary resolution, the number of phase qubits was increased to 5.

We then run this circuit (Figure 13) for 4000 shots and compile results in a histogram (Figure 14). $n = 1$ system qubit $\implies 2^1$ eigenvalues $\lambda = \{1, 3\}$. Taking $t = \frac{2\pi}{3}$ with $k = 5$ phase qubits hence,

$$\lambda_1 = 1 \rightarrow \hat{\phi} = 1\frac{2\pi}{3 \times 2\pi} = 0.3 \equiv 0.6 \rightarrow 10101$$

$$\lambda_2 = 3 \rightarrow \hat{\phi} = 3\frac{2\pi}{3 \times 2\pi} = 1 \equiv 0 \rightarrow 00000$$

Since $1 \equiv 0 \mod 1$, and Qiskit uses little Endian ordering, so $\frac{1}{3} \rightarrow 1 - \frac{1}{3} = \frac{2}{3}$. We may conclude that the construction of the unitaries is correct and an accuracy is achieved at tw phase qubits. Results with probability less than 1% are filtered out to increase readability without removing critical information.

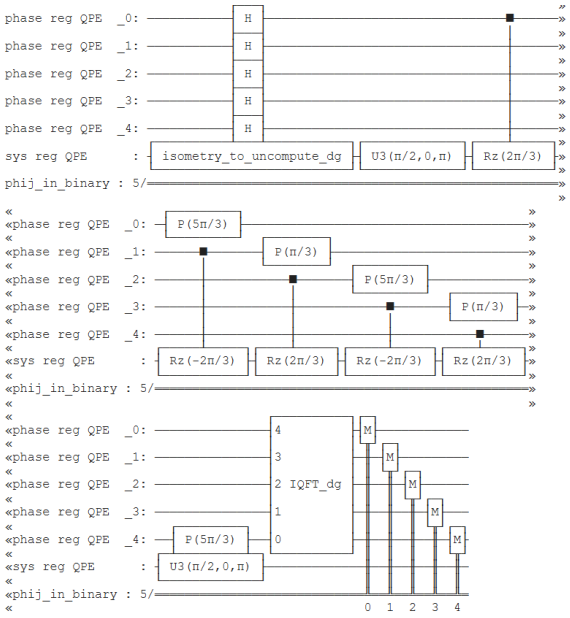


Figure 13. Quantum Phase Estimation circuit for the minimum 1D Poisson solver.

5.3. HHL Circuit—Quantum Poisson Solver

A quantum algorithm is a quantum circuit, so to solve the Poisson equation on a quantum computer, a circuit must be constructed which represents the discretized partial differential equation. The problem-specific gates include the controlled unitaries where the matrix A is encoded and the state preparation wherein the vector b is encoded by amplitude encoding. The dimension (construction of A) and then the resolution of the space is chosen first to define the size of A and b.

Once the circuit is built, it is run for many iterations; these iterations are colloquially known as shots. Results are compiled into a histogram and $\vec{x}$ is found by looking at the ratios. Hence, there is not an exact numerical result with the HHL algorithm but rather a structure of the electric field may be observed.

To achieve readable results using a quantum circuit, the number of k phase qubits must be significantly higher than the number of m system qubits. This is due to the fact that k is the binary resolution of the eigenvalues and m is the size of the matrix, hence the number of eigenvalues.

We will observe two small cases and discuss the opportunities and challenges. Quick explanations are done throughout the text and in depth step-by-step calculations for gates and outputs may be found in Appendix A.

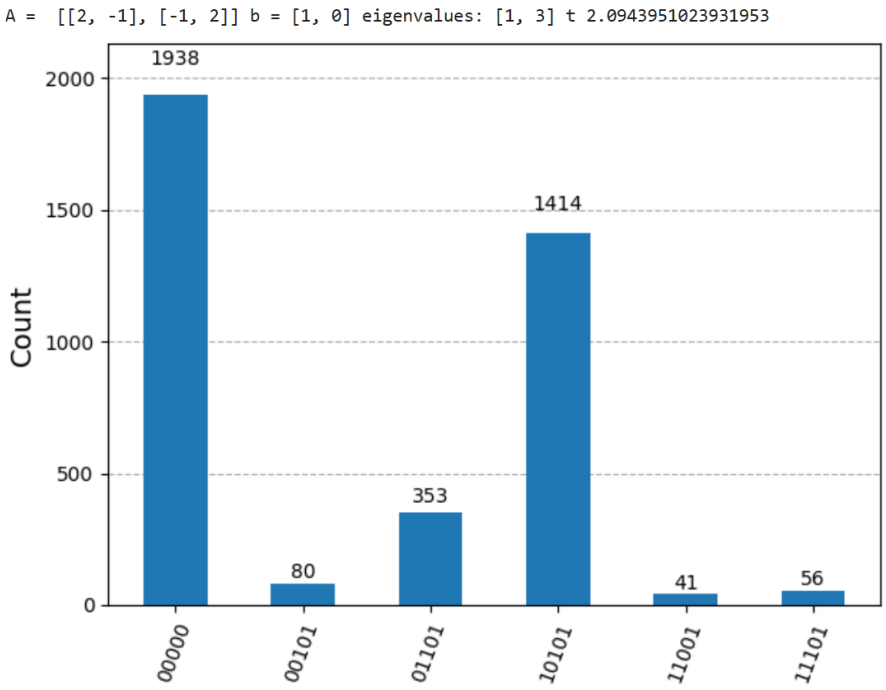


Figure 14. Histogram for 1D Poisson solver just Quantum Phase Estimation section, i.e., estimations of eigenvalues 1 and 3.

5.3.1. 1D Case (Minimum Example)

The minimum case for a discrete domain the represent the Poisson equation is a line defined by two points. So continuing from Section 5.2.1, where $m = 1$ system qubit and $k = 2$ phase qubits was confirmed to have a map, it is now possible to continue to implement the HHL algorithm with a greater awareness of the accuracy of the solver. This small case as a proof of concept may be used to validate the circuit by also following closely examples found in the literature [4,27,29,30] before extending to higher dimensions which will require additional qubits. The problem-relevant gates are the state preparation, unitaries, and rotational Y for eigenvalue inversion (Algorithm 4). Selecting b to be a classical state will allow an easy implementation for the first case. The state b is defined to be the simple classical state $|1\rangle$ which may be constructed by adding an X gate to the system register. Recalling the order of operations outlined in Figure 10, because there is only 1 system qubit to represents $A \in \mathbb{R}^{n\times n}$ where $n = 2^m$. The smallest Hermitian matrix $A = L_2^{1D}$, which has eigenvalues $\lambda = \{1, 3\}$. To construct the circuit, i.e., the gates, we use the HHL algorithm and follow the procedure on gate construction from literature [4]. Figure 10 outlines the Hybrid nature of this procedure, with some preparation calculations being preformed on a classical computer to prepare gates. Both the unitaries and rotational Y gates are built from the binary eigenvalues, so it makes sense to determine those first. Since this is a small case (only 2 eigenvalues, both < 4), we may look at Figure 4 to determine the binary eigenvalues. So, we may take $t = \frac{\pi}{2}$ hence $\hat{\phi} = \{\frac{1}{4}, \frac{3}{4}\} \rightarrow \{|01\rangle, |11\rangle\}$. The unitaries are constructed by first calculating the phase matrix for the eigenvalues followed by the column matrix of eigenvectors. The entries correspond to the binary map. They are then implemented in the Phase Estimations portion of the solver (Figure 15). Once the gates are built, we may construct the circuit to build the HHL algorithm which consists of b state preparation, QPE (Hadamards, controlled-unitaries, iQFT), rotational Y, iQPE (QFT, controlled-unitaries, Hadamards), and then finally measurement. The 1D Quantum Poisson Solver (minimum case of one system qubit for a 2×2 mesh). Similarly, the R_y eigenvalue inversion gates are built using the binary eigenvalues with the ancilla for a flag where

phases that start with 0 are ignored, and the system register is measured to find the solution vector $\vec{x}$. Each eigenvalue is given an R_y gate, hence $k = 2^n$. Since QPE has already been validated at $k = 2$, the completed HHL should have correct outputs even at such a low binary resolution.

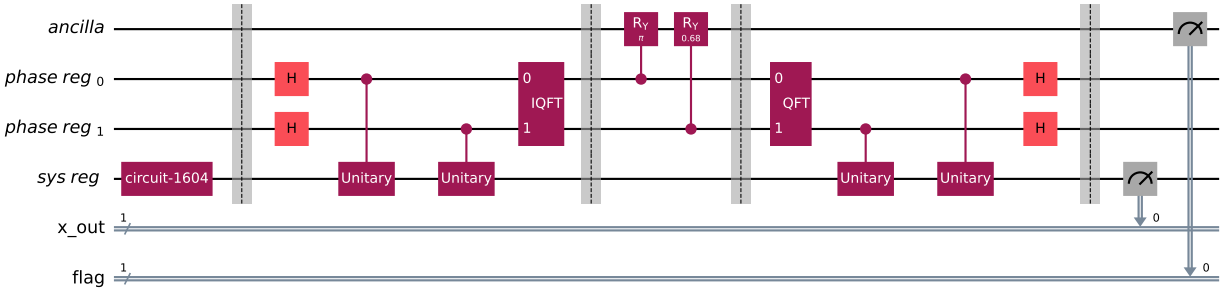


Figure 15. Circuit to build the HHL algorithm.

The circuit is run for 4000 shots and compiled into a histogram (Figure 16). Note: the minimum case does not provide significant research results; however, illustrates a small example to become familiar with quantum algorithms in a step-by-step approach. It is easy to calculate $x = (L_2^{1D})^{-1}|1\rangle = \begin{bmatrix} \frac{1}{3} & \frac{2}{3} \end{bmatrix}^T$, hence the output from the quantum solver is as expected (see Appendix A) and we may move onto high dimension cases.

$$\vec{x} \propto \begin{bmatrix} 1 & 1.9 \end{bmatrix} \tag{33}$$

Though not an exact result, likely due to the fact that there are not enough phase qubits, the result is promising to continue to additional system qubits with $k = 2^n$.

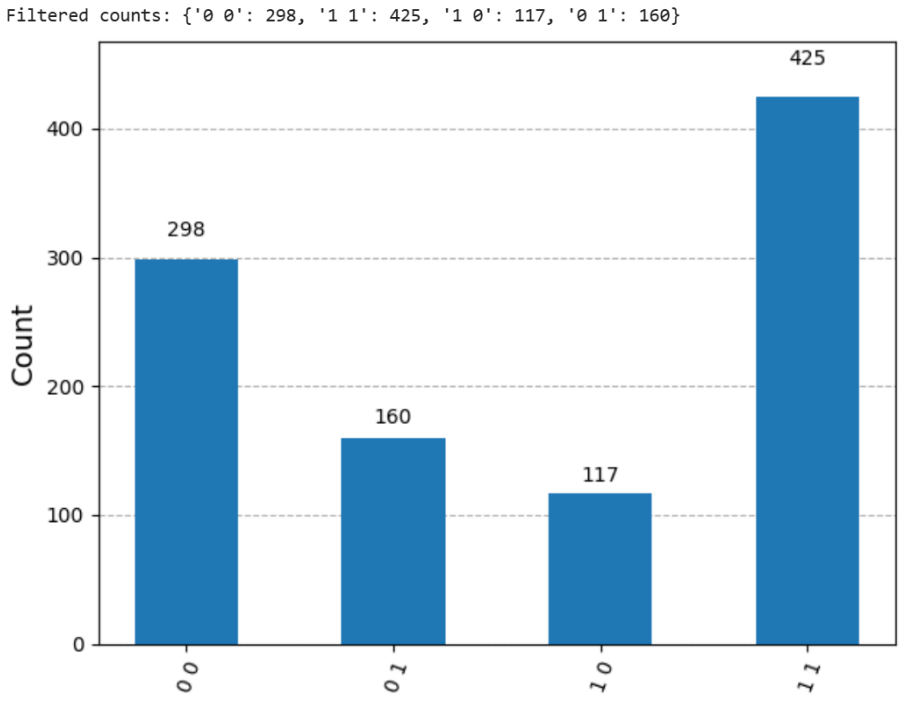


Figure 16. Histogram for 1D Poisson solver using the HHL algorithm.

5.3.2. 2D Case (Low Resolution with Dirichlet Boundary Conditions)

A kinetic model of plasma requires either a 2D or 3D space to observe the motion of the particles. Compilation time due the column matrix and phase matrix decomposition

of A, the number of qubits for Qiskit 2.0 state vector, and the fact that $k = 2^n, m = 4$ system qubits and 16 phase qubits should be used; however, 8 were found to be sufficient. A total of 13 qubits (with the ancilla flag) were required for the HHL circuit to solve a minimum 2D case, i.e., a mesh of size $n = 2^4 = 16$ and a binary resolution of eigenvalues of $2^8 = 256$. So $\Delta\phi_{ij} = 1/2^{256} = 0.00390625$, which allows a good binary map because $\lambda(L_{16}^{2D}) = \{0.7639320225, 1.7639320225, \ldots, 7.2360679775\}$. Mixed boundary conditions offer the best model of EP because side walls are insulated; however, this would require an eigenbasis which is not orthonormal, and therefore matrix A which is not Hermitian. Hence, for 2D, Hermitian Boundary conditions were used to ensure that the eigenvectors are orthonormal and the resulting gates are unitary matrices.

We will run for 4000 shots with outcomes of probability $< 1\%$ filtered out (Figure 17). While not offering an exact numerical answer, these results may be used to find critical points or fluctuations in a plasma. We may now construct the HHL algorithm circuit. We take $\frac{\rho}{\epsilon_0} \rightarrow b = [0.19, \ldots, 0.19]$ as a rough proof of concept. We build this circuit the usual way in Qiskit (Figure 18). b was chosen to be all equal entries; the histogram exhibits the same shape of the chosen A.

Using the method outlined by [4], the solution $\vec{x}$ is found by taking the amplitudes. Just by glancing at this histogram, the results are promising. Results with probability $< 1\%$ have been filtered out, so the ancilla flag of 0 is significantly less probable. There is also not an exact numerical result with this algorithm but rather a vector of ratios. So at a quick glance, while recalling the structure of L_{2D} recalling that b was chosen to be all equal entries, the histogram exhibits the same shape as the chosen A.

$$\begin{aligned} b &= [70 \quad 128 \quad 128 \quad 70 \quad 125 \quad 431 \quad 382 \quad 146 \quad 138 \quad 425 \quad 424 \quad 156 \quad 82 \quad 133 \quad 138 \quad 75]^T \\ \implies \vec{x} &\propto [0.1323 \quad 0.1789 \quad 0.1789 \quad 0.1323 \quad 0.1767 \quad 0.3283 \quad 0.3090 \quad 0.1910 \\ &\quad 0.1857 \quad 0.3259 \quad 0.3255 \quad 0.1975 \quad 0.1432 \quad 0.1824 \quad 0.1857 \quad 0.1369] \end{aligned} \tag{34}$$

The algorithm does not output an exact numerical answer but rather displays a geometry which could be used to find critical points or fluctuations in plasma.

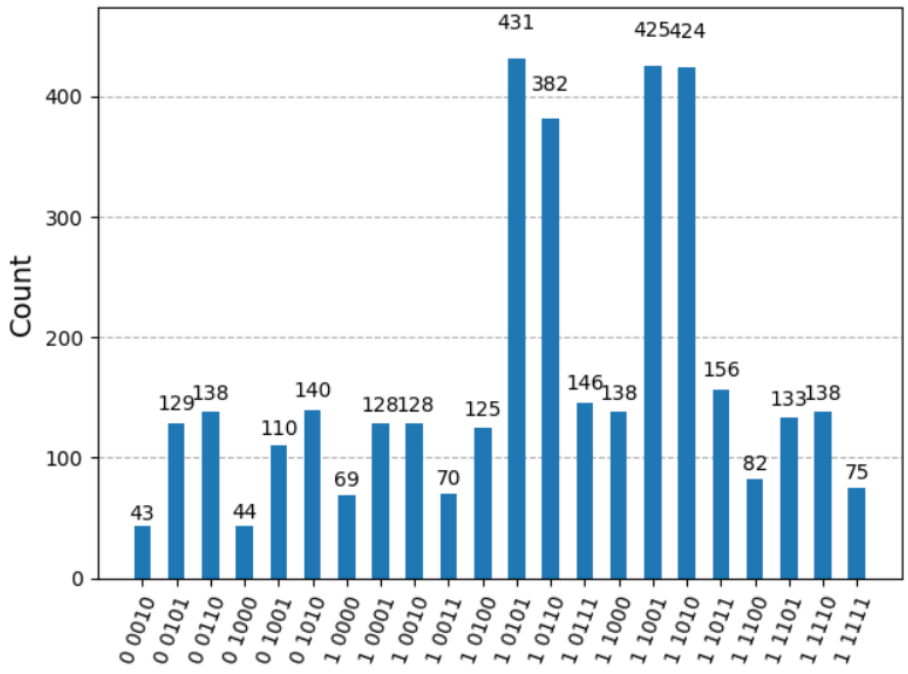


Figure 17. Histogram of the 2D Poisson solver. Outputs with an ancilla flag of 0 are discarded.

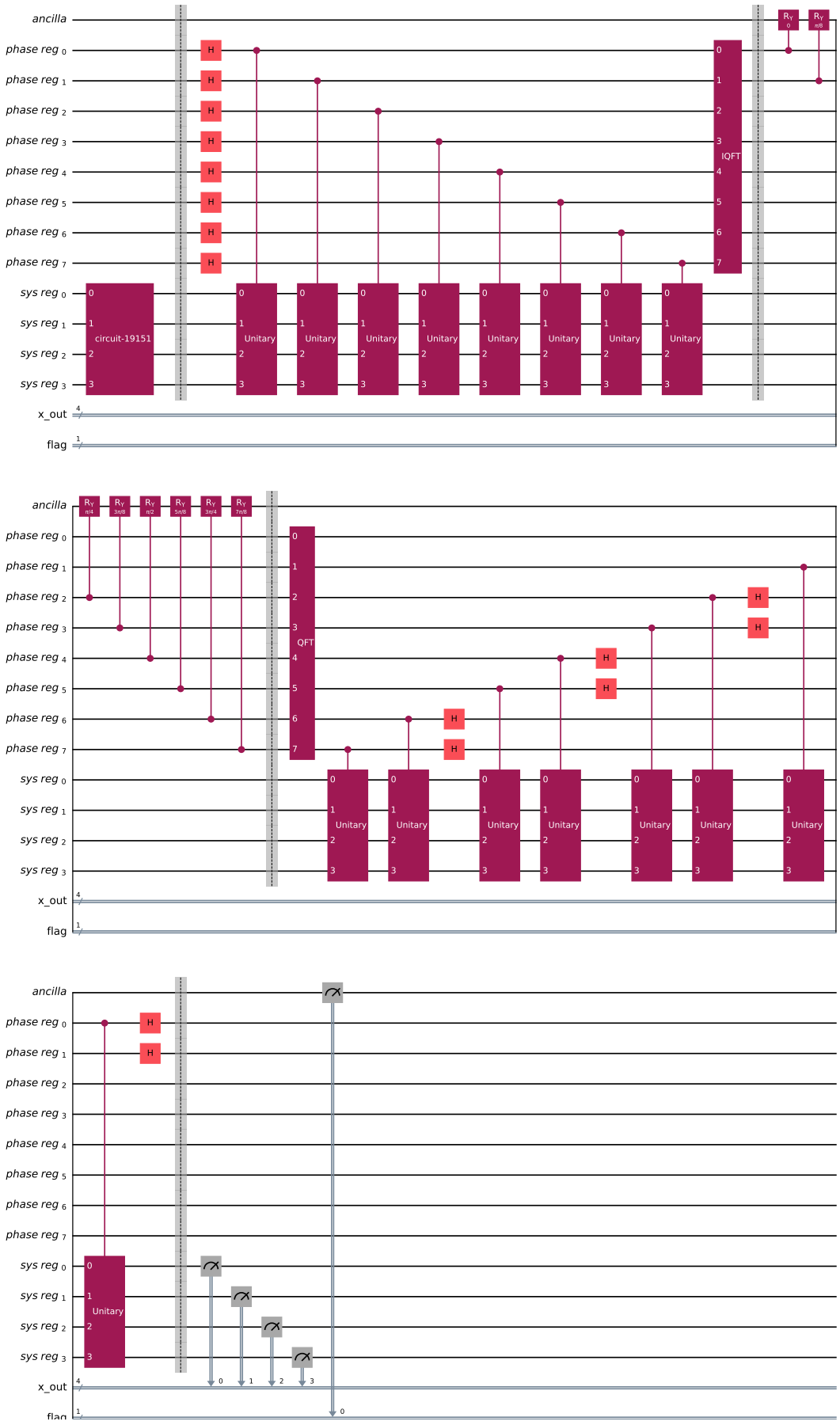


Figure 18. Quantum 2D Poisson solver via HHL circuit.

https://doi.org/10.3390/aerospace13110975

5.4. Concluding Remarks

In this paper, we have presented a framework for future research by presenting a step-by-step mesh to quantum gate level and ultimately a quantum algorithm, while explaining how to scale and providing numerical examples (in Appendix A) for different applications and boundary conditions. We implemented a 2D Poisson solver with Dirichlet boundary conditions and solved using the HHL algorithm. The future of space systems design will potentially use quantum algorithms to achieve optimal results. Known issues such as anomalous electron motion in HETs [6,14], and emitter shorting in ESP [16,17], as well as looking at suggested solutions before lab work (such as coatings on extractors [31]), will be modeled on quantum computers. The current results are already promising and we must develop and test the algorithms to be ready for the arrival of quantum hardware. Quantum Computing will ultimately offer a better understanding of the operation of EP, aiding in the design and development in achieving resilient spacecraft and long-duration missions. Much is still unknown about the behaviour of particles within EP and similarly, algorithms which have the potential of providing the quantum advantage are still under development.

This paper attempts to fix some bottlenecks [5] by speeding up the manner of constructing the unitary by directly finding the eigenvalues and eigenvectors and then solving the LSP. The time complexity was reduced; the presented algorithm for 1D has a time complexity of $O(n)$, for 2D the time complexity of $O(n)$; for previous work the time complexity was $O(n)$ because of permuting over all Pauli matrices to make strings. So, the time complexity was drastically reduced; however, it is limited to only strictly diagonal matrices, which means only Dirichlet boundary conditions could be modeled, i.e., fixed boundaries. For instance, the boundary conditions for ESP at the emitter and extractor [32] may be fixed; and for HETs the anode and the exit plane could be fixed [33].

Quantum computing is still in the Noisy Intermediate-Scale Quantum (NISQ) era, with many quantum algorithms not offering a distinct advantage to existing classical software. [34]. In these results, the presented algorithm is not showing sufficient noise where results are illegible; however, further challenges and opportunities have to be improved.

5.5. Challenges and Opportunities: Future Work

There are many challenges and opportunities [12] present in current builds and opportunities for future development. Challenges include quantum noise and well-known classical bottlenecks present in the HHL algorithm. Future work may be done by extending meshes (while ensuring the Hermitian property remains true) and connecting with various applications. Significant work is being done not only to look at small cases but also to present scalable circuits [20].

Current challenges exist with quantum algorithms generally and areas for improvement in this specific algorithm. In the case of quantum computing, we continue to add additional qubits (width) and the longest sequence of logically dependent gates (depth) we begin to see quantum noise making it more difficult, and in some cases impossible, to make any conclusions on a numerical result. For this algorithm specifically, HHL's well known bottle neck of classical preparation results in an algorithm that does not lead to any speedup in contrast to available classical algorithms.

In this code, there are known issues that may be fixed but must be considered. Linear meshes must be Hermitian for the decomposition of eigenvalues and eigenvectors (orthonormal basis) to build unitary matrices, as has been done in many quantum algorithms [35]. Estimated eigenvalues outputted from QPE must be quite close to expected outputs; otherwise, we will not see reliable outputs of x. Ry eigenvalue inversion gates input a function of the eigenvalue; we must take care in choosing a C such that $\arcsin(\theta)$ is not out of bounds.

There are many opportunities for future development. Quantum computing is still in the early stage and quantum modeling of real-world applications is a rarity. Starting with fixing known issues surrounding the quantum circuits and inclusion of stabilizer circuits for noise reduction in scaling. Additionally, more work is required in the classical preparation stages to ensure a quantum advantage over existing classical methods such as the conjugate gradient method and the Fast Fourier Transform (FFT). Currently, we have implemented an arbitrary vector with real entries. However, this does not show a fully accurate representation of the charge density ρ and permitivity of the space ϵ_0, ρ/ϵ_0 term. To fully show a quantum advantage, we must both show a more accurate simulation that is more quantum than classical, and compiles faster.

$$\rho = \rho_i - \rho_e = c(n_i - n_e) = c(n_0 e^{-e\phi/kT_i} - n_0 e^{e\phi/kT_e}) \tag{35}$$

The charge density must be calculated and updated. The permittivity of free space will be taken for a vacuum; this is simply a constant. We may use the same circuit structure in future work, normalizing the term and then using amplitude encoding and will be explored in future work.

Author Contributions: A.V.N. developed the software, analyzed the results and wrote the paper. C.M.-J. and M.J. reviewed the results and revised the work as thesis supervisors and Professors. All authors have read and agreed to the published version of the manuscript.

Funding: The authors wish to gratefully acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (*NSERC Discovery Grant*), the Canada Research Chair Program (*CRC*) and the Directorate of Space Requirements (*DSR, Department of National Defence*).

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the author on request.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Step-by-Step Calculations

1D Case (Simplest Case)

To provide a small initial example, we will choose the smallest mesh so that we may go into detail on the calculations without the cumbersomeness of redundant calculations. We choose $N = 2$, which can only be simulated in 1D. We choose b to be a classical state (which nicely happens to be in the A basis so that and we can be certain that QPE will work correctly).

PART I : We first perform the classical preparations.

1. DEFINING THE MODEL: By the above, we build the linear system problem taking a mesh of size 2 with Dirichlet boundary conditions and $b = |1\rangle$,

$$\frac{d^2u}{dx} = f(x) \quad \text{where} \quad u(0) = u(L) = 0$$

$$\begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

2. EIGEN-SPECTRUM: By Equation (20),

$$\lambda_1 = 1 \quad \lambda_2 = 3$$

Since we wish to preserve ratios with the minimum number of qubits by Equation (30) we solve for each $\varphi_j \approx \lambda_j t$,

$$\begin{aligned} T &= \lceil \log_2(3) \rceil \\ &= \lceil 1.6 \rceil \\ &= 2 \end{aligned}$$

Hence $t = \frac{\pi}{2}$ (i.e., $\frac{1}{T^2}$ roots of unity) and,

$$\varphi_1 = \frac{1}{4} = |01\rangle \quad \varphi_2 = \frac{3}{4} = |11\rangle$$

We calculate the base Λ, which may be scaled. We use Euler's Formula $e^{i\varphi_j 2^k} = \cos(i\varphi_j 2^k) + i\sin(i\varphi_j 2^k)$ for each λ_j

$$\Lambda^{2^0} = \begin{bmatrix} i & 0 \\ 0 & -i \end{bmatrix} \quad \Lambda^{2^1} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

3. EIGEN-SPACE: By Equation (20),

$$\mu_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \mu_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Which we normalize

$$\hat{\mu}_1 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \quad \hat{\mu}_2 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix}$$

And then find the Column matrix

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

Since $V \in \mathbb{R}^{N \times N}$, $V^\dagger = V^T$ and since we have selected a Hermitian matrix, $V^T = V$

PART II: We now construct the quantum gates.

4. CONTROLLED UNITARIES

$$U^{2^0} = V\Lambda^{2^0}V^\dagger = \begin{bmatrix} 0 & i \\ i & 0 \end{bmatrix} = iX$$

$$U^{2^1} = V\Lambda^{2^1}V^\dagger = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} = -I$$

5. RY Using the above mapping of eigenvalues to binary fractions, we find each $\theta_k = 2\arcsin(\frac{C}{\tilde{\lambda}_k})$. Since we have found an exact mapping, this calculation is straightforward.

$$\theta_1 = 2\arcsin(\frac{1}{1}) = \frac{\pi}{2} \qquad \Longrightarrow \qquad R_Y(\frac{\pi}{2}) = \begin{bmatrix} \cos(\frac{\pi}{4}) & -\sin(\frac{\pi}{4}) \\ \sin(\frac{\pi}{4}) & \cos(\frac{\pi}{4}) \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

$$\theta_2 = 2\arcsin(\frac{1}{3}) = 0.679673819 \implies R_Y(0.679673819) = \begin{bmatrix} 0.942809042 & -0.333333333 \\ 0.333333333 & 0.942809042 \end{bmatrix}$$

6. STATE PREPARATION: Since we have chosen a classical state for $b = |1\rangle$ and states in Qiskit are initialized at $|0\rangle$, we may simply apply an X gate in the state preparation step.

7. RESULTS: Since we do not get exact numerical results, we look at ratios. We may first easily calculate the result classically to get an expected $x = \begin{bmatrix} \frac{1}{3} & \frac{2}{3} \end{bmatrix}^T$. We reference Figure 16

$$S(|10\rangle) = 117$$
$$P(|10\rangle) = \frac{S_j}{S_{tot}} = \frac{117}{1000}$$
$$\alpha(|10\rangle) = \sqrt{P(|10\rangle)}$$
$$S(|11\rangle) = 425$$
$$P(|11\rangle) = \frac{S_j}{S_{tot}} = \frac{425}{1000}$$
$$\alpha(|11\rangle) = \sqrt{P(|11\rangle)}$$

We then find the ratio: $\frac{\sqrt{425/4000}}{\sqrt{117/4000}}$
this is a 1:1.9059 ratio, which matches the 1:2 ratio with noise.

References

1. Rayman, M.D.; Varghese, P.; Lehman, D.H.; Livesay, L.L. Results from the Deep Space 1 technology validation mission. *Acta Astronaut*. **2000**, *47*, 475–487. [CrossRef]
2. Pardini, C.; Anselmo, L. Environmental sustainability of large satellite constellations in low earth orbit. *Acta Astronaut*. **2020**, *170*, 27–36. [CrossRef]
3. Weston, S.V.; Burkhard, C.D.; Stupl, J.M.; Ticknor, R.L.; Yost, B.D.; Austin, R.A.; Galchenko, P.; Newman, L.K.; Soto, L.S. *Small Spacecraft Technology State-of-the-Art Report*; Technical report; NASA: Washington, DC, USA, 2024.
4. Zaman, A.; Jr, H.J.M.; Wong, H.Y. A Step-by-Step HHL Algorithm Walkthrough to Enhance Understanding of Critical Quantum Computing Concepts. *IEEE Access* **2023**, *11*, 77117–77131. [CrossRef]
5. Narbonne, A.; Malardier-Jugroot, C.; Jugroot, M. Quantum Algorithms for the Investigation of Particle Motion in Electric Propulsion: A Step-by-Step Approach. In Proceedings of the 39th International Electric Propulsion Conference, London, UK, 14–19 September 2025.
6. Boeuf, J.P. Tutorial: Physics and modeling of Hall thrusters. *J. Appl. Phys*. **2017**, *121*, 011101. [CrossRef]
7. Mallalieu, P.; Jugroot, M. Multimodal electrospray thruster for small spacecraft: Design and experimental characterization. *J. Electr. Propuls*. **2024**, *3*, 12. [CrossRef] [PubMed]
8. Bapat, A.; Salunkhe, P.B.; Patil, A.V. Hall-Effect Thrusters for Deep-Space Missions: A Review. *IEEE Trans. Plasma Sci*. **2022**, *50*, 189–202. [CrossRef]
9. Ergun, R.E.; Carlson, C.W.; McFadden, J.P.; Mozer, F.S.; Muschietti, L.; Roth, I.; Strangeway, R.J. Debye-Scale Plasma Structures Associated with Magnetic-Field-Aligned Electric Fields. *Phys. Rev. Lett*. **1998**, *81*, 826–829. [CrossRef]
10. Taylor, G.I. Disintegration of Water Drops in an Electric Field. *Proc. R. Soc. A* **1964**, *280*, 383–397. [CrossRef]
11. Takahashi, N. Molecular Dynamics Modeling of Ionic Liquids in Electrospray Propulsion. Ph.D. Thesis, MIT, Cambridge, MA, USA, 2010.
12. Biswas, R.; Jiang, Z.; Kechezhi, K.; Knysh, S.; Mandra, S.; O'Gorman, B.; Perdomo-Ortiz, A.; Petukhov, A.; Realpe-Gómez, J.; Rieffel, E.; et al. A NASA perspective on quantum computing: Opportunities and challenges. *Parallel Comput*. **2017**, *64*, 81–98. [CrossRef]
13. Birdsall, C.; Langdon, A. *Plasma Physics via Computer Simulation*; CRC Press: Boca Raton, FL, USA, 1991. [CrossRef]
14. Lafleur, T.; Baalrud, S.D.; Chabert, P. Theory for the anomalous electron transport in Hall effect thrusters. II. Kinetic model. *Phys. Plasmas* **2016**, *23*, 053503. [CrossRef]
15. Reza, M.; Faraji, F.; Knoll, A. Concept of the generalized reduced-order particle-in-cell scheme and verification in an axial-azimuthal Hall thruster configuration. *J. Phys. D Appl. Phys*. **2023**, *56*, 175201. [CrossRef]
16. Wright, P.; Thuppul, A.; Wirz, R.E. Life-Limiting Emission Modes for Electrospray Thrusters. In Proceedings of the 2018 Joint Propulsion Conference, Cincinnati, OH, USA, 9–11 July 2018. [CrossRef]
17. Thuppul, A.; Wright, P.L.; Collins, A.L.; Ziemer, J.K.; Wirz, R.E. Lifetime Considerations for Electrospray Thrusters. *Aerospace* **2020**, *7*, 108. [CrossRef]
18. Little, B.; Jugroot, M. Bimodal Propulsion System for Small Spacecraft: Design, Fabrication, and Performance Characterization. *J. Spacecr. Rocket*. **2020**, *57*, 707–719. [CrossRef]

19. Eleanor, G.R.; Hadfield, S.; Hogg, T.; Mandrà, S.; Marshall, J.; Mossi, G.; O'Gorman, B.; Plamadeala, E.; Tubman, N.M.; Venturelli, D.; et al. From Ansatze to Z-gates: A NASA View of Quantum Computing. *arXiv* **2019**, arXiv:1905.02860.
20. Gustafson, E.; Sherbert, K.; Florio, A.; Shirali, K.; Chen, Y.; Lamm, H.; Valgushev, S.; Weichselbaum, A.; Economou, S.E.; Pisarski, R.D.; et al. Surrogate-constructed scalable-circuits adaptive variational quantum eigensolver in the Schwinger model. *Phys. Rev. Appl.* **2025**, *23*, 064002. [CrossRef]
21. Munoz-Bauza, H.; Lidar, D.A. Scaling Advantage in Approximate Optimization with Quantum Annealing. *Phys. Rev. Lett.* **2025**, *134*, 160601. [CrossRef] [PubMed]
22. Cochran, W.; Cooley, J.; Favin, D.; Helms, H.; Kaenel, R.; Lang, W.; Maling, G.; Nelson, D.; Rader, C.; Welch, P. What is the fast Fourier transform? *Proc. IEEE* **1967**, *55*, 1664–1674. [CrossRef]
23. Weinstein, Y.S.; Pravia, M.A.; Fortunato, E.M.; Lloyd, S.; Cory, D.G. Implementation of the Quantum Fourier Transform. *Phys. Rev. Lett.* **2001**, *86*, 1889–1891. [CrossRef] [PubMed]
24. Boyd, I.D. Numerical modeling of spacecraft electric propulsion thrusters. *Prog. Aerosp. Sci.* **2005**, *41*, 669–687. [CrossRef]
25. Asher, J.; Huang, Z.; Cui, C.; Wang, J. Multi-scale modeling of ionic electrospray emission. *J. Appl. Phys.* **2022**, *131*, 014902. [CrossRef]
26. Harrow, A.W.; Hassidim, A.; Lloyd, S. Quantum Algorithm for Solving Linear Systems of Equations. *Phys. Rev. Lett.* **2009**, *103*, 150503. [CrossRef] [PubMed]
27. Cao, Y.; Papageorgiou, A.; Petras, I.; Traub, J.; Kais, S. Quantum algorithm and circuit design solving the Poisson equation. *New J. Phys.* **2013**, *15*, 013021. [CrossRef]
28. Strang, G. *3.5 Finite Differences and Fast Poisson Solvers*; Technical Report AM35; Massachusetts Institute of Technology, Department of Mathematics: Cambridge, MA, USA, 2006.
29. Daribayev, B.; Mukhanbet, A.; Imankulov, T. Implementation of the HHL Algorithm for Solving the Poisson Equation on Quantum Simulators. *Appl. Sci.* **2023**, *13*, 11491. [CrossRef]
30. Wang, S.; Wang, Z.; Li, W.; Fan, L.; Wei, Z.; Gu, Y. Quantum Fast Poisson Solver: The Algorithm and Modular Circuit Design. *Quantum Inf. Process.* **2020**, *19*, 170. [CrossRef]
31. Laws, N.; Petro, E. Ab initio simulations of EMI-BF4 neutral-surface interactions in electrospray thrusters. *J. Appl. Phys.* **2026**, *139*, 113302. [CrossRef]
32. Borner, A.; Wang, P.; Levin, D.A. Influence of electrical boundary conditions on molecular dynamics simulations of ionic liquid electrosprays. *Phys. Rev. E* **2014**, *90*, 063303. [CrossRef] [PubMed]
33. Adam, J.C.; Boeuf, J.P.; Dubuit, N.; Dudeck, M.; Garrigues, L.; Gresillon, D.; Heron, A.; Hagelaar, G.J.M.; Kulaev, V.; Lemoine, N.; et al. Physics, simulation and diagnostics of Hall effect thrusters. *Plasma Phys.* **2008**, *50*, 124041. [CrossRef]
34. Bharti, K.; Cervera-Lierta, A.; Kyaw, T.H.; Haug, T.; Alperin-Lea, S.; Anand, A.; Degroote, M.; Heimonen, H.; Kottmann, J.S.; Menke, T.; et al. Noisy intermediate-scale quantum algorithms. *Rev. Mod. Phys.* **2022**, *94*, 015004. [CrossRef]
35. Auckenthaler, T.; Bader, M.; Huckle, T.; Spörl, A.; Waldherr, K. Matrix exponentials and parallel prefix computation in a quantum control problem. *Parallel Comput.* **2010**, *36*, 359–369. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.