Quantum Simulation of Hawking Radiation Using VQE Algorithm on IBM Quantum Computer

Saqlain Afroz

20MS230

Contents

[1]

[1]This project work is an attempt to reproduce the results of following paper

1 Introduction

Hawking radiation is a quantum phenomena and it was proposed by Stephen Hawking. Hawking predicted that there is creation of particle and anti-particle pair near the Event Horizon of Black Hole, due to intense gravity and when these particles gets separated, one of them goes into the vast Universe, while the other one is absorbed by the Black Hole. The energy lost during this process is released in the form of radiation known as **Hawking Radiation**. We will be using Qiskit to create a model of Hawking radiation, by finding the eigenvalues of a rotating, non-charged black hole, for which we will use VQE (Variational Quantum Eigensolver algorithm).

2 Black Hole Metric

We will be working with a spherically symmetric, non-rotating black hole. We know that Einstein's field equations are given as

$$R_{\mu\nu} - \frac{1}{2} R g_{\mu\nu} = \kappa T_{\mu\nu}$$

In a matter free space the term $T_{\mu\nu}$ goes to zero and the solution of above equation in a matter free space, is given by Birkhoff's theorem, which is $\nabla_\mu G^{\mu\nu} = 0$, and is known as Schwarzschild solution. The Schwarzschild metric is then given as:

$$ds^2 = -\left(1 - \frac{2M}{R}\right) dt^2 + \left(1 - \frac{2M}{R}\right)^{-1} dR^2 + R^2 d\Phi^2$$

where $d\Phi^2 = d\theta^2 + \sin^2\theta d\phi^2$. We note that, we are working in Planck's units, so G = c = 1.

The complete derivation is quite lengthy but I have done some if it and is available on my website, and the link for the same is here.

Now let's we consider an isotropic radius, r, such that

$$R = r\left(1 + \frac{M}{2r}\right)^2$$

And before transforming our metric, we calculate some useful relations

$$\begin{aligned} dR &= \left[\left(1 + \frac{M}{2r}\right)^2 + 2r\left(1 + \frac{M}{2r}\right)\left(-\frac{M}{2r^2}\right)\right] dr \\ &= \left[\left(1 + \frac{M}{2r}\right)\left(1 + \frac{M}{2r} - \frac{M}{r}\right)\right] dr \\ &= \left(1 + \frac{M}{2r}\right)\left(1 - \frac{M}{2r}\right) dr \end{aligned}$$

which then implies that

$$dR^2 = \left(1+\frac{M}{2r}\right)^2\left(1-\frac{M}{2r}\right)^2 dr^2$$

$$\begin{aligned}\left(1-\frac{2M}{R}\right) &= 1-\frac{2M}{r\left(1+\frac{M}{2r}\right)^2}\\ &= \frac{r\left(1+\frac{M}{2r}\right)^2-2M}{r\left(1+\frac{M}{2r}\right)^2}\\ &= \frac{\left(1+\frac{M}{2r}\right)^2-2M/r}{\left(1+\frac{M}{2r}\right)^2}\\ &= \frac{(1-M/2r)^2}{(1+M/2r)^2}\end{aligned}$$

$$\begin{aligned}\left(1-\frac{2M}{R}\right)^{-1} dR^2 &= \frac{(1+M/2r)^2}{(1-M/2r)^2}\left(1+\frac{M}{2r}\right)^2\left(1-\frac{M}{2r}\right)^2 dr^2\\ &= \left(1+\frac{M}{2r}\right)^4 dr^2\end{aligned}$$

Now we can write the Schwarzschild metric as following:

$$\begin{aligned}ds^2 &= -\frac{(1-M/2r)^2}{(1+M/2r)^2}dt^2+\left(1+\frac{M}{2r}\right)^4 dr^2+\left(1+\frac{M}{2r}\right)^4 d\Phi^2\\ &= -\frac{(1-M/2r)^2}{(1+M/2r)^2}dt^2+\left(1+\frac{M}{2r}\right)^4 (dr^2+d\Phi^2)\end{aligned}$$

We will be needing to transform this metric into rectangular coordinate system, such that

$$\begin{aligned}x &= r\sin\theta\cos\phi\\ y &= r\sin\theta\sin\phi\\ z &= r\cos\theta\\ r^2 &= x^2+y^2+z^2\end{aligned}$$

and in doing so we can see that

$$\begin{aligned} dr^2 &= dx^2 + dy^2 + dz^2 \\ r^2 d\theta^2 &= dx^2 + dy^2 + dz^2 \\ r^2 \sin^2\theta d\phi^2 &= dx^2 + dy^2 + dz^2 \end{aligned}$$

Thus our Schwarzschild metric can be written in following form:

$$ds^2 = -\frac{(1 - M/2r)^2}{(1 + M/2r)^2} dt^2 + \left(1 + \frac{M}{2r}\right)^4 (dx^2 + dy^2 + dz^2)$$

Now to form a Hamiltonian we need to do 3+1 Decomposition of this metric. For our case we can take the time slicing as a function of coordinate variable t, and thus we get the spatial metric γ_{ij} as the coefficients of $(dx^2 + dy^2 + dz^2)$:

$$\gamma_{ij} = \left(1 + \frac{M}{2r}\right)^4 \eta_{ij} \tag{1}$$

where, $i, j = 1, 2, 3$.

or we can say that proper distance in our each time slice, Σ can be written as

$$ds^2 = \left(1 + \frac{M}{2r}\right)^4 (dx^2 + dy^2 + dz^2) \tag{2}$$

We can obtain Lagrangian from this equation by dividing throughout with ds^2, as we can see that, then the RHS becomes like a Kinetic energy terms, and $\mathcal{L} = T - U$.

Due to our choice of time-slicing function as t, we can see that our spatial metric is independent of time, which means that our metric tensor remains constant throughout the time evolution. Thus we can say that the Hamiltonian is constant wrt. time. Now with the help of Jacobi's formulation of the principle of least action and inverse of Eq. (2), we can define the Hamiltonian for a non-rotating black hole as follows:

$$H^d = \frac{1}{2}\left(1 + \frac{M}{2r}\right)^{1/4} \left[\frac{(p_x^d)^2}{2} + \frac{(p_y^d)^2}{2} + \frac{(p_z^d)^2}{2}\right]$$

3 Discretization of Hamiltonian

To solve this Hamiltonian in a quantum computer we need to discretize it. So let's take a space of $x, y \in [-L, L]$ such that x, y each has N eigenvalues. We finally obtain a mesh with N^2 spatial elements, where each element corresponds to an eigenvalue specific to the x and y value for that element. Let the mesh be centered at $[0, 0]$ then we have an $N \times N$ matrix for the position operator, with position eigenvalues lying along the diagonal. The position operator is given as (for N = 4):

$$x^d = \sqrt{\frac{\pi}{8}} \cdot \begin{bmatrix} -2 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Now we find p^d using the following relation:

$$p^d = (F^d)^{-1} x^d F^d = (F^d)^\dagger x^d F^d$$

where F^d is the discrete Fourier transform (DFT) matrix, $(F^d)^{-1}$ is its inverse (which equals its conjugate transpose $(F^d)^\dagger$ due to unitarity). For a general N, we have

$$[F^d]_{j,k} = \frac{\exp i2\pi jk/N}{N^{1/2}}$$

First Row ((j = 0)):

$$[F^d]_{0,0} = \frac{\exp\left(i2\pi \cdot \frac{0\cdot 0}{4}\right)}{2} = \frac{1}{2}; [F^d]_{0,1} = \frac{\exp\left(i2\pi \cdot \frac{0\cdot 1}{4}\right)}{2} = \frac{1}{2}$$
$$[F^d]_{0,2} = \frac{\exp\left(i2\pi \cdot \frac{0\cdot 2}{4}\right)}{2} = \frac{1}{2}; [F^d]_{0,4} = \frac{\exp\left(i2\pi \cdot \frac{0\cdot 4}{4}\right)}{2} = \frac{1}{2}$$

Second Row ((j = 1)):

$$[F^d]_{1,0} = \frac{\exp(i2\pi \cdot 1 \cdot 0/4)}{2} = \frac{1}{2}$$
$$[F^d]_{1,1} = \frac{\exp(i2\pi \cdot 1 \cdot 1/4)}{2} = \frac{\exp(i\pi/2)}{2} = \frac{i}{2}$$
$$[F^d]_{1,2} = \frac{\exp(i2\pi \cdot 1 \cdot 2/4)}{2} = \frac{\exp(i\pi)}{2} = \frac{-1}{2}$$
$$[F^d]_{1,3} = \frac{\exp(i2\pi \cdot 1 \cdot 3/4)}{2} = \frac{\exp(i3\pi/2)}{2} = \frac{-i}{2}$$

Third Row ((j = 2)):

$$[F^d]_{2,0} = \frac{\exp(i2\pi \cdot 2 \cdot 0/4)}{2} = \frac{1}{2}$$
$$[F^d]_{2,1} = \frac{\exp(i2\pi \cdot 2 \cdot 1/4)}{2} = \frac{\exp(i\pi)}{2} = \frac{-1}{2}$$
$$[F^d]_{2,2} = \frac{\exp(i2\pi \cdot 2 \cdot 2/4)}{2} = \frac{\exp(i2\pi)}{2} = \frac{1}{2}$$
$$[F^d]_{2,3} = \frac{\exp(i2\pi \cdot 2 \cdot 3/4)}{2} = \frac{\exp(i3\pi)}{2} = \frac{-1}{2}$$

Fourth Row ((j = 3)):

$$
\begin{aligned}
[F^d]_{3,0} &= \frac{\exp(i2\pi \cdot 3 \cdot 0/4)}{2} = \frac{1}{2} \\
[F^d]_{3,1} &= \frac{\exp(i2\pi \cdot 3 \cdot 1/4)}{2} = \frac{\exp(i3\pi/2)}{2} = \frac{-i}{2} \\
[F^d]_{3,2} &= \frac{\exp(i2\pi \cdot 3 \cdot 2/4)}{2} = \frac{\exp(i3\pi)}{2} = \frac{-1}{2} \\
[F^d]_{3,3} &= \frac{\exp(i2\pi \cdot 3 \cdot 3/4)}{2} = \frac{\exp(i9\pi/2)}{2} = \frac{i}{2}
\end{aligned}
$$

Thus the DFT matrix F_4 for $N = 4$ is:

$$F_4 = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix}$$

Since the DFT matrix is unitary, its inverse is its conjugate transpose:

$$F_4^\dagger = F_4^{-1} = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}$$

To compute $p^d = F_4^\dagger x^d F_4$, we follow the given steps:

1. First, we multiply x^d by F_4:

$$x^d \cdot F_4 = \sqrt{\frac{\pi}{8}} \cdot \begin{bmatrix} -2 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix}$$

This results in:

$$x^d \cdot F_4 = \frac{\sqrt{\pi}}{\sqrt{8 \cdot 4}} \cdot \begin{bmatrix} -2 & -2 & -2 & -2 \\ -1 & -i & 1 & i \\ 0 & 0 & 0 & 0 \\ 1 & -i & -1 & i \end{bmatrix}$$

2. Next, we multiply by $F_4^\dagger$:

$$F_4^\dagger \cdot (x^d \cdot F_4) = \frac{\sqrt{\pi}}{8\sqrt{2}} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix} \begin{bmatrix} -2 & -2 & -2 & -2 \\ -1 & -i & 1 & i \\ 0 & 0 & 0 & 0 \\ 1 & -i & -1 & i \end{bmatrix}$$

Performing the matrix multiplication gives:

$$p^d = \frac{\sqrt{\pi}}{8\sqrt{2}} \begin{bmatrix} -2 & -2-2i & -2 & -2+2i \\ -2+2i & -2 & -2+2i & -2 \\ -2 & -2+2i & -2 & -2-2i \\ -2-2i & -2 & -2-2i & -2 \end{bmatrix}$$

Now in order to make measurements, we need to account for higher dimension. For this we take tensor product of $(p^d)^2$ with Identity matrix, this is because measurement made in one momentum space is independent of the presence of other other momnetum spaces. So, we can write our Hamiltonian as:

$$H^d = \frac{1}{2}\left(1+\frac{M}{2r}\right)^{\frac{1}{4}}\left[(p^d)^2 \otimes I \otimes I + I \otimes (p^d)^2 \otimes I + I \otimes I \otimes (p^d)^2\right] \tag{3}$$

4 Variational Quantum Eigensolver

We follow the given algorithm:

1. **Ansatz Preparation**: Start by preparing an ansatz, which is used to search for the ground state wave function, $\psi(\vec{\theta})$, for the given Hamiltonian.
2. **Initial Parameters**:The initial parameters $\vec{\theta}$ of the ansatz are chosen randomly.
3. **Apply the Hamiltonian**: Once the wave function is prepared, apply the Hamiltonian operator and perform measurements to obtain the energy eigenvalue E_n.
4. **Classical Optimization**: We use a classical optimizer to update the parameters $\vec{\theta}$ in order to minimize the energy eigenvalue and approach the ground state energy, E_g.
5. **Iterative Process**: The updated parameters $\vec{\theta}$ are used as new ansatz parameters, and the process is repeated.
6. **Stopping Criterion**: The iteration continues for a maxiter.
7. **Optimizer Choice**: In this case, the Simultaneous Perturbation Stochastic Approximation (SPSA) optimizer is used to find the upper bound of the energy eigenvalue for different values of M (mass) and r (radial distance) in the Hamiltonian.

The code that I wrote to perform the above tasks is given below:

Some important packages to import

```
import numpy as np
from qiskit import Aer
from qiskit.circuit import Parameter
from qiskit.algorithms import VQE
from qiskit.algorithms.optimizers import SPSA
from qiskit.opflow import X, I, PauliSumOp
from qiskit import QuantumCircuit
from qiskit.utils import QuantumInstance
import matplotlib.pyplot as plt
from scipy.linalg import eigh
```

Defining Hamiltonian for my system

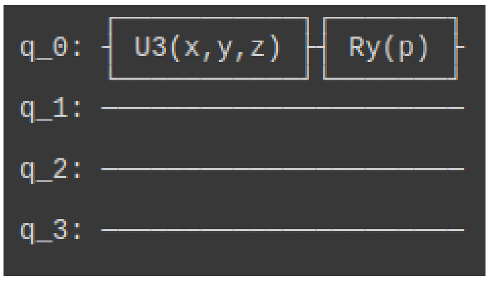
```
def create_hamiltonian(M, r):
    coefficient = 0.5 * (1 + M / (2 * r)) ** 0.25

    terms = [
        ('XXII', coefficient * (np.pi / 8)),
        ('XIII', coefficient * (np.pi / 16)),
        ('IXII', coefficient * (np.pi / 8)),
        ('IIII', coefficient * (3 * np.pi / 16)),
        ('IXXI', coefficient * (np.pi / 8)),
        ('IXII', coefficient * (np.pi / 16)),
        ('IIXI', coefficient * (np.pi / 8)),
        ('IIII', coefficient * (3 * np.pi / 16)),
        ('IIXX', coefficient * (np.pi / 8)),
        ('IIXI', coefficient * (np.pi / 16)),
        ('IIIX', coefficient * (np.pi / 8)),
        ('IIII', coefficient * (3 * np.pi / 16))
    ]

    hamiltonian = PauliSumOp.from_list(terms)

    return hamiltonian
```

The PauliSumOp() function is used to define quantum operators in terms of Pauli matrices.

Here we are preparing an ansatz for my quantum circuit, where we have U3 gate on qubit register 0 and Ry gate on the same qubit register. Thus we have a parameterized circuit.

```
def parametrized_ansatz():
    theta_u3 = Parameter('x')
    phi_u3 = Parameter('y')
    lam_u3 = Parameter('z')

    theta_ry = Parameter('p')

    qc = QuantumCircuit(4)

    qc.u3(theta_u3, phi_u3, lam_u3, 0)

    qc.ry(theta_ry, 0)

    return qc, [theta_u3, phi_u3, lam_u3, theta_ry]
```

The pictorial representation of the circuit is given below.



We are defining optimizer, namely perturbative stochastic approximation optimizer, which finds and updates the parameters so that we get an upper bound for energy eigenvalues for different values of M and r.

Also quantum_instance specifies the quantum backend, here we are using the simulator (statevector_simulator) from Aer to perform simulations.

```
optimizer = SPSA(maxiter=100)

quantum_instance = QuantumInstance(Aer.get_backend('
    ↪ statevector_simulator'))
```

This is a function to run VQE for a given mass, which returns the computed minimum eigenvalue using VQE algorithm (and optimal parameters).

```
def run_vqe_for_mass(M, r):
    hamiltonian = create_hamiltonian(M, r)

    ansatz, parameters = parametrized_ansatz()

    vqe = VQE(ansatz, optimizer, quantum_instance=quantum_instance)

    result = vqe.compute_minimum_eigenvalue(hamiltonian)
    return result.eigenvalue.real, result.optimal_parameters
```

Now we are using the following function to compute exact eigenvalues using matrix diagonalization from scipy.linalg.eigh module and finally return the minimum eigenvalue.

```
def compute_exact_eigenvalue(hamiltonian):
    matrix = hamiltonian.to_matrix().real

    eigenvalues, _ = eigh(matrix)

    return np.min(eigenvalues)
```

Now we vary mass from 1 to 10, keeping radius fixed and store results for minimum eigenvalue computed using VQE as well as the exact eigenvalue using scipy module.

```python
r = 1.0
results = []
for M in range(1, 17):
    vqe_eigenvalue, optimal_parameters = run_vqe_for_mass(M, r)

    hamiltonian = create_hamiltonian(M, r)
    exact_eigenvalue = compute_exact_eigenvalue(hamiltonian)

    results.append((M, vqe_eigenvalue, exact_eigenvalue))

    print(f"Mass: {M}, VQE Minimum Eigenvalue: {vqe_eigenvalue}, Exact
    ↪ Minimum Eigenvalue: {exact_eigenvalue}")
```

Plotting the results

```python
masses = [result[0] for result in results]
vqe_eigenvalues = [result[1] for result in results]
exact_eigenvalues = [result[2] for result in results]

plt.plot(masses, vqe_eigenvalues, marker='o', label='VQE')
plt.plot(masses, exact_eigenvalues, marker='x', label='Exact',
  ↪ linestyle='--')
plt.xlabel('Mass (M)')
plt.ylabel('Minimum Eigenvalue')
plt.title('Minimum Eigenvalue vs Mass (VQE vs Exact)')
plt.legend()
plt.grid(True)
plt.show()
```

The final output of thee above code is:

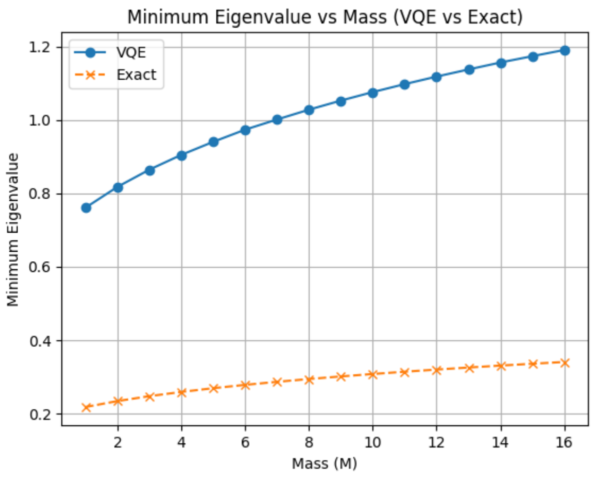


Figure 1:

Now for Radius versus Energy Eigenvalue we modify the above code's last parts as:

```
M_fixed = 5.0
results = []
r_values = np.linspace(1.0, 10.0, 16)

for r in r_values:
    vqe_eigenvalue, optimal_parameters = run_vqe_for_radius(M_fixed, r)

    hamiltonian = create_hamiltonian(M_fixed, r)
    exact_eigenvalue = compute_exact_eigenvalue(hamiltonian)

    results.append((r, vqe_eigenvalue, exact_eigenvalue))

    print(f"Radius: {r:.2f}, VQE Minimum Eigenvalue: {vqe_eigenvalue:.4
        ↪ f}, Exact Minimum Eigenvalue: {exact_eigenvalue:.4f}")

radii = [result[0] for result in results]
vqe_eigenvalues = [result[1] for result in results]
exact_eigenvalues = [result[2] for result in results]

plt.plot(radii, vqe_eigenvalues, marker='o', label='VQE')
plt.plot(radii, exact_eigenvalues, marker='x', label='Exact',
    ↪ linestyle='--')
plt.xlabel('Radius (r)')
plt.ylabel('Minimum Eigenvalue')
plt.title('Minimum Eigenvalue vs Radius (VQE vs Exact)')
plt.legend()
plt.grid(True)
plt.show()
```

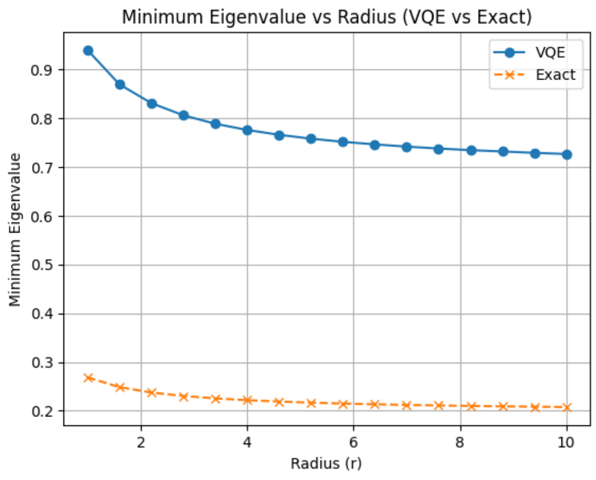


Figure 2:

5 Conclusion

From the two plots that we got we infer following two properties for a Black Hole:

1. A black hole that has more mass has more energy than a black hole that has less mass. (Fig. 1)
2. Also we notice that the energy of the black hole decreases at a faster rate, for small radial distances (most likely inside the event horizon) while it decreases at a slower pace outside the event horizon. (Fig. 2)