Quantum-Enhanced Advection Equation Solver

Ibtissam Chaibi

July 4, 2025

Abstract

This document details the numerical method and quantum algorithm used to solve the one-dimensional advection equation. The approach combines classical finite difference discretization with a quantum linear solver (VQLS) to evolve the solution in time.

1 Problem Formulation

The advection equation is given by:

$$\frac{\partial u}{\partial t} + C\frac{\partial u}{\partial x} = 0 \tag{1}$$

where $u(x,t)$ is the quantity being advected and C is the constant advection speed.

2 Classical Discretization

We discretize the equation using:

- **Spatial discretization**: Second-order central difference

$$\frac{\partial u}{\partial x} \approx \frac{u_{i+1}^{j+1} - u_{i-1}^{j+1}}{2\Delta x}$$

- **Temporal discretization**: First-order implicit Euler method

$$\frac{\partial u}{\partial t} \approx \frac{u_i^{j+1} - u_i^j}{\Delta t}$$

Substituting into Eq. (1):

$$\frac{u_i^{j+1} - u_i^j}{\Delta t} + C\frac{u_{i+1}^{j+1} - u_{i-1}^{j+1}}{2\Delta x} = 0$$

Rearranging terms:

$$u_i^{j+1} - u_i^j + \frac{C\Delta t}{2\Delta x}(u_{i+1}^{j+1} - u_{i-1}^{j+1}) = 0$$

Define $r = \frac{C\Delta t}{2\Delta x}$:

$$-ru_{i-1}^{j+1} + u_i^{j+1} + ru_{i+1}^{j+1} = u_i^j$$

This leads to a linear system at each time step:

$$\mathbf{A}\mathbf{u}^{j+1} = \mathbf{u}^j \tag{2}$$

where $\mathbf{A}$ is a circulant tridiagonal matrix:

$$\mathbf{A} = \begin{bmatrix} 1 & r & 0 & \cdots & -r \\ -r & 1 & r & \cdots & 0 \\ 0 & -r & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r & 0 & 0 & \cdots & 1 \end{bmatrix}$$

3 Quantum Algorithm

We solve the linear system (2) at each time step using the Variational Quantum Linear Solver (VQLS). The algorithm consists of:

3.1 Problem Encoding

- Matrix $\mathbf{A}$ is encoded as a linear combination of unitaries (implicitly by its structure)
- Vector $\mathbf{u}^j$ is normalized to form the state $|b\rangle = \frac{\mathbf{u}^j}{\|\mathbf{u}^j\|}$

3.2 Variational Circuit

We use a hardware-efficient ansatz with parameterized rotations and entangling gates:

```python
def ansatz(theta, wires):
    n_qubits = len(wires)
    layers = len(theta) // (2 * n_qubits)
    idx = 0
    for _ in range(layers):
        for i in wires:
            qml.RY(theta[idx], wires=i)
            idx += 1
        for i in range(n_qubits-1):
            qml.CNOT(wires=[i, i+1])
        for i in wires:
            qml.RY(theta[idx], wires=i)
            idx += 1
        for i in reversed(range(n_qubits-1)):
            qml.CNOT(wires=[i, i+1])
```

3.3 Cost Function

The cost function minimizes the residual:

$$C(\theta) = \|\mathbf{A}|\psi(\theta)\rangle - |b\rangle\|^2$$

3.4 Optimization

We use the Adam optimizer to update the parameters θ to minimize $C(\theta)$.

4 Hybrid Workflow

1. Discretize the spatial domain and set initial condition u^0
2. For each time step j:
    (a) Form the right-hand side vector $\mathbf{u}^j$
    (b) Normalize to get $|b\rangle$
    (c) Run VQLS to solve $\mathbf{A}\mathbf{u}^{j+1} = \mathbf{u}^j$ and get $|\psi(\theta)\rangle$
    (d) Rescale the quantum solution to get $\mathbf{u}^{j+1}$