Algorithm Design Brief: A Quantum-Inspired PDE Solver for CFD

Jerome B

August 10, 2025
WISER X Womanium Quantum Program Project with BQP

1 Chosen Framework: Quantum Tensor Network (QTN)

This project utilizes the Quantum Tensor Network (QTN) framework, specifically leveraging **Matrix Product States (MPS)**, to solve the 1D viscous Burgers' Equation. This approach is "quantum-inspired," meaning it uses mathematical techniques from quantum many-body physics to efficiently represent and manipulate large classical data structures on a classical computer.

The core motivation is to overcome the "curse of dimensionality" inherent in classical CFD solvers. By representing the state vector of the fluid (the velocity at each grid point) as a compressed MPS, the memory required scales *linearly* with the number of qubits (N) used to define the grid, rather than *exponentially* with N (2^N) as in a classical approach.

Our prototype is a hybrid QTN solver. It maintains the fluid state as a compressed MPS at all times but uses a "classical shortcut" for the numerical calculations. This involves decompressing the MPS to a vector, applying stable classical numerical schemes, and immediately re-compressing the result. This strategy allows us to validate the physics and the MPS data structure while avoiding the complexities of pure MPO-MPS arithmetic, making it an ideal approach for a near-term prototype.

2 Mapping of the PDE to the QTN Framework

The 1D viscous Burgers' Equation is the central focus of our simulation. Understanding its components is key to understanding the fluid's behavior.

$$\underbrace{\frac{\partial u}{\partial t}}_{\text{Acceleration}} = - \underbrace{u\frac{\partial u}{\partial x}}_{\text{Convection}} + \nu\underbrace{\frac{\partial^2 u}{\partial x^2}}_{\text{Diffusion}}$$

- **Acceleration ($\frac{\partial u}{\partial t}$):** This term represents the rate of change of the fluid's velocity at a fixed point in space. It's the quantity we aim to solve for at each time step.
- **Convection ($u\frac{\partial u}{\partial x}$):** This non-linear term describes how the fluid's own momentum carries it from one place to another. It is the dominant term responsible for the formation and movement of shock waves.

- **Diffusion ($\nu\frac{\partial^2 u}{\partial x^2}$):** This term, scaled by the viscosity ν, describes the tendency of the fluid's velocity to "smooth out" or "spread." It is the dominant term responsible for the blurring and stabilization of sharp features like shock waves.

The mapping to our framework involves several key steps:

2.1 Discretization

The continuous spatial domain $x \in [0, 1]$ is discretized into 2^N points, where N is the number of qubits. For our prototype, we use $N = 4$, resulting in 16 grid points. The velocity field $u(x, t)$ is thus represented by a state vector of size 2^N.

2.2 State Representation as an MPS

The state vector $\mathbf{u}(t)$ is converted into an MPS. This is the central feature of our solver. A vector of size 2^N is reshaped into a rank-N tensor, which is then decomposed via Singular Value Decomposition (SVD) into a chain of N smaller tensors. This conversion is handled by the `quimb` library's `MatrixProductState.from_dense` method. At each time step, the state of our system is stored in this compressed format.

2.3 Time Evolution

To simulate the flow of time on a computer, we must convert the continuous PDE into a discrete, step-by-step formula. We start by approximating the time derivative:

$$\frac{\partial u}{\partial t} \approx \frac{u(t + \Delta t) - u(t)}{\Delta t}$$

Substituting this into the Burgers' equation and solving for the velocity at the next time step, $u(t + \Delta t)$, gives us the **Forward Euler method**:

$$\mathbf{u}(t + \Delta t) \approx \mathbf{u}(t) + \Delta t \left(-\mathbf{u}(t) \circ \frac{\partial \mathbf{u}(t)}{\partial x} + \nu \frac{\partial^2 \mathbf{u}(t)}{\partial x^2} \right)$$

In our prototype, this update is performed using the classical shortcut:

1. The MPS $\mathbf{u}_{mps}(t)$ is decompressed to the dense vector $\mathbf{u}(t)$.
2. The derivative terms are calculated using stable classical methods (an upwind scheme for the first derivative and central differences for the second).
3. The right-hand-side (RHS) is calculated using NumPy.
4. The new vector $\mathbf{u}(t + \Delta t)$ is calculated.
5. This new vector is immediately re-compressed into a new MPS, $\mathbf{u}_{mps}(t + \Delta t)$.

3 Gate Decomposition and Quantum Implementation

The quantum part of our prototype focuses on the most fundamental subroutine: **state preparation**. This corresponds to **Deliverable #2** and demonstrates that the data structures used in our QTN solver can be faithfully represented on a quantum computer.

3.1 The Quantum Kernel

We have implemented a quantum circuit using **Qiskit** that prepares the initial state of the fluid on $N = 4$ qubits. The target state is the normalized vector corresponding to the initial shock tube condition.

3.2 Gate Decomposition

We use Qiskit's high-level `QuantumCircuit.initialize()` function. This is a powerful instruction that, when transpiled, is automatically decomposed into a sequence of single-qubit rotations (U3 gates) and two-qubit entangling gates (CNOTs). The exact decomposition is hardware-specific and is handled by the transpiler to optimize for a given backend. This circuit serves as our runnable prototype for a noiseless simulator and a real QPU.

4 Preliminary Resource Estimates

This section provides an initial estimate of the resources required to implement the quantum components of this solver. A more detailed analysis will be provided in the final report.

Resource	**Estimate**	**Rationale**
Qubits	N	The number of qubits scales logarithmically with the number of grid points (2^N).
Two-Qubit Gates	$O(2^N)$	For the `initialize` method, the number of CNOT gates required scales exponentially in the worst case.
Circuit Depth	$O(2^N)$	The depth is also expected to scale exponentially for an arbitrary state preparation.
Mitigation Strategy	ZNE / PEC	For near-term hardware, Zero-Noise Extrapolation (ZNE) or Probabilistic Error Cancellation (PEC) would be suitable for mitigating errors in expectation value calculations.

Table 1: Preliminary resource estimates for the quantum state preparation kernel.

This brief outlines our successful implementation of a quantum-inspired solver. The prototype is stable, physically correct, and demonstrates the core principles of the QTN framework.