Quantum Algorithm as a PDE Solver for Computational Fluid Dynamics

Nahid Binandeh Dehaghani, Ban Tran

August 2025

1 Introduction

The viscous Burgers' equation is a fundamental nonlinear partial differential equation (PDE) used to model phenomena such as shock formation, turbulent transport, and nonlinear wave propagation. Traditional numerical solvers, including finite difference and spectral methods, often become computationally expensive when applied to fine spatial grids or extended time evolutions.

Quantum-inspired tensor network methods—particularly *Matrix Product States* (MPS) and *Matrix Product Operators* (MPOs)—offer compact representations of discretized scalar fields and differential operators. Originally developed for simulating quantum many-body systems, these structures exploit spatial correlations to achieve exponential compression in high-dimensional settings.

This report adapts the quantum-inspired computational fluid dynamics (CFD) framework proposed in [1] to the one-dimensional viscous Burgers' equation. We outline the discretization strategy, MPS/MPO mappings, and algorithmic workflow, aiming to enable efficient simulation of nonlinear PDEs on large spatial domains using tensor network techniques.

2 Problem Statement

We study the one-dimensional viscous Burgers' equation, a nonlinear PDE modeling advection–diffusion processes and serving as a simplified prototype for turbulence and shock dynamics:

$$\frac{\partial u}{\partial t} + u\,\frac{\partial u}{\partial x} = \nu\,\frac{\partial^2 u}{\partial x^2}, \quad x \in [0,1],\ t > 0, \tag{1}$$

Here, $u(x,t)$ is the scalar velocity field and $\nu > 0$ is the viscosity coefficient. The nonlinear convection term appears on the left-hand side, while the right-hand side models viscous diffusion.

The problem setup is:

- **Initial condition:** A smoothed Riemann step profile:

$$u(x,0) = \frac{1}{2}\left[1 - \tanh\!\left(\frac{x-0.5}{\varepsilon}\right)\right], \tag{2}$$

  where $\varepsilon > 0$ controls the transition steepness.

- **Boundary conditions:** Dirichlet boundaries:

$$u(0,t) = u_L, \quad u(1,t) = u_R, \tag{3}$$

  with fixed constants u_L, u_R.

This configuration models the viscous evolution of an initial discontinuity. Resolving steep gradients or shock-like features typically requires fine spatial grids, which can make dense-vector simulations memory-intensive and computationally demanding.

To address this, we encode $u(x,t)$ as a compressed *Matrix Product State* (MPS) and apply differential operators as *Matrix Product Operators* (MPOs). This enables efficient time evolution with reduced computational and memory cost, particularly well-suited for one-dimensional domains.

3 Tensor Network Framework Overview

The tensor network framework is a classical simulation method inspired by quantum many-body physics. It represents high-dimensional data using low-rank tensor factorizations, enabling efficient storage and manipulation. In PDE solvers, spatial fields are stored as *Matrix Product States* (MPS), while differential operators are implemented as *Matrix Product Operators* (MPOs), allowing scalable and memory-efficient time evolution.

3.1 Matrix Product State (MPS) Encoding

We discretize the scalar field $u(x,t)$ on a uniform grid of 2^N points, indexed by binary strings $(i_1, i_2, \ldots, i_N) \in \{0,1\}^N$. Instead of storing the full 2^N-dimensional vector, we approximate it with a chain of low-rank tensors:

$$u_{i_1 i_2 \cdots i_N}(t) \approx A_1^{(i_1)}(t) A_2^{(i_2)}(t) \cdots A_N^{(i_N)}(t), \tag{4}$$

where each site tensor $A_k^{(i_k)} \in \mathbb{R}^{\chi_{k-1} \times \chi_k}$, and χ_k is the *bond dimension* controlling accuracy. This reduces memory usage from $\mathcal{O}(2^N)$ to $\mathcal{O}(N\chi^2)$.

The initial condition $u(x,0)$ is encoded directly as an MPS using the TT-cross algorithm, which constructs a low-rank approximation by sampling the function at selected points, avoiding dense vector formation.

3.2 Matrix Product Operators (MPO)

Spatial derivatives are discretized using finite-difference stencils and encoded as MPOs—chains of four-index tensors $W_k^{(i_k, j_k)}$ that act on an MPS via site-wise tensor contractions:

$$\left(\frac{\partial u}{\partial x}\right)_i \approx \frac{u_{i+1} - u_{i-1}}{2\Delta x}, \tag{5}$$

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_i \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2}. \tag{6}$$

These MPOs typically have small bond dimension ($\chi \leq 5$), enabling efficient operator application without assembling full matrices.

3.3 Advantages

The tensor network approach provides:

- **Memory efficiency:** Only $\mathcal{O}(N\chi^2)$ parameters are stored.
- **Modularity:** Fields (MPS) and operators (MPO) are independent, allowing flexible composition.
- **Nonlinearity handling:** Products of MPSs can represent nonlinear terms, though bond dimension may grow and require truncation to maintain efficiency.

This compression and modularity make tensor networks a practical framework for simulating nonlinear PDEs such as the viscous Burgers' equation on large grids.

4 Algorithmic Pipeline

The following steps outline the computational procedure for solving the one-dimensional viscous Burgers' equation using the tensor network framework, following the methodology described in [1].

1. **Spatial Discretization:** The domain $x \in [0,1]$ is discretized into 2^N grid points. Each grid index is represented as a binary string $(i_1, i_2, \ldots, i_N) \in \{0,1\}^N$, allowing the scalar field to be encoded as a Matrix Product State (MPS).

2. **Initialization:** The initial profile

$$u(x,0) = \frac{1}{2}\left(1 - \tanh\left(\frac{x - 0.5}{\varepsilon}\right)\right)$$

   is encoded directly into MPS form using the TT-cross algorithm, which constructs a low-rank approximation by sampling the function at selected grid points.

3. **Operator Encoding:** Discrete derivative operators ∂_x and ∂_{xx} are implemented as Matrix Product Operators (MPOs) using centered finite-difference stencils. The MPO format enables efficient application to the MPS without explicit matrix assembly.

4. **Nonlinear Term Evaluation:** The advection term $u\,\partial_x u$ is computed by first applying the ∂_x MPO to u, yielding $\partial_x u$ in MPS form. The product $u \cdot \partial_x u$ is then approximated via tensor contraction, increasing the bond dimension. Periodic truncation is applied to control this growth.

5. **Time Integration:** Time evolution is performed using an explicit time-stepping scheme (e.g., forward Euler), where each term in the Burgers' equation is evaluated in MPS form and combined to update the solution.
6. **Boundary Conditions:** Dirichlet boundary conditions are enforced by modifying the edge tensors or by projecting the MPS onto a constrained subspace after each time step.
7. **Periodic Truncation:** To maintain computational efficiency, the MPS is periodically truncated to a maximum bond dimension χ. This ensures that the representation remains compact while preserving essential features of the solution.

This procedure enables efficient simulation of the Burgers' equation in a compressed tensor network format, significantly reducing memory usage while maintaining numerical accuracy.

5 Algorithm Summary

The following procedure summarizes the quantum-inspired tensor network method for solving the one-dimensional viscous Burgers' equation, following the implementation described in [1]. The algorithm encodes the solution as a Matrix Product State (MPS) and the spatial derivatives as Matrix Product Operators (MPOs). Nonlinear terms are computed directly in MPS form, with periodic truncation to control bond dimension growth.

Algorithm 1 Tensor Network Solver for 1D Viscous Burgers' Equation

```
1: Input:
     • Number of spatial points: $N = 2^n$
     • Time step: $\Delta t$, final time: $T$
     • Viscosity: $\nu$, maximum bond dimension: $\chi_{\max}$
     • Initial condition: $u(x,0) = \frac{1}{2}\left(1 - \tanh\left(\frac{x-0.5}{\varepsilon}\right)\right)$
     • Boundary values: $u(0,t) = u_L$, $u(1,t) = u_R$
2: Output: MPS representation $U_T \approx u(x,T)$
3: 1. Discretize the spatial domain $x \in [0,1]$ into $N$ points with spacing $\Delta x = \frac{1}{N-1}$.
4: 2. Encode the initial condition $u(x,0)$ as an MPS using the TT-cross algorithm.
5: 3. Construct MPOs for the first and second derivatives ($\partial_x$ and $\partial_{xx}$) using centered finite-difference stencils.
6: for each time step $t = 0$ to $T$ with step size $\Delta t$ do
7:     Apply $\partial_x$ MPO to $U$ to obtain $V = \partial_x U$.
8:     Compute the nonlinear term $F_{\text{adv}} = U \odot V$ (approximated pointwise product in MPS form).
9:     Apply $\partial_{xx}$ MPO to $U$ to obtain $F_{\text{diff}}$.
10:    Update the solution:
           $U_{\text{new}} \leftarrow U - \Delta t \cdot F_{\text{adv}} + \Delta t \cdot \nu \cdot F_{\text{diff}}.$
11:    Truncate $U_{\text{new}}$ to bond dimension $\chi_{\max}$ to control growth.
12:    Enforce Dirichlet boundary conditions by modifying edge tensors or projecting boundary values.
13:    Set $U \leftarrow U_{\text{new}}$.
14: return $U_T = U$ as the final MPS approximation.
```

6 Resource Estimates and Scalability

The QTN-based method for solving the 1D viscous Burgers' equation reduces computational and memory costs by representing both the solution and the operators in compressed tensor formats. The key parameters influencing performance are:

- $N = 2^n$: number of spatial grid points
- χ: bond dimension of the MPS/MPO tensors
- $T/\Delta t$: number of time steps

6.1 Memory Requirements

In a standard grid-based solver, storing the field $u(x,t)$ over N points requires $\mathcal{O}(N)$ memory. In contrast, an MPS representation stores u as a chain of $n = \log_2 N$ tensors, each of size $\mathcal{O}(\chi^2)$, giving:

$$\text{Memory}_{\text{MPS}} = \mathcal{O}(n \cdot \chi^2) = \mathcal{O}(\log N \cdot \chi^2)$$

This leads to substantial compression when χ remains small.

6.2 Computational Complexity

The dominant cost per time step comes from:

- Applying MPOs for ∂_x and ∂_x^2 to the MPS: $\mathcal{O}(n \cdot \chi^3)$
- Computing nonlinear terms and performing periodic truncation: $\mathcal{O}(n \cdot \chi^3)$

For $M = T/\Delta t$ time steps, the total complexity is:

$$\mathcal{O}(M \cdot \log N \cdot \chi^3)$$

This scaling allows simulations on very large grids (e.g., $N = 2^{20}$) with moderate bond dimensions ($\chi \lesssim 32$).

6.3 Accuracy vs. Efficiency

The bond dimension χ controls how well the MPS captures fine features of the solution:

- Small χ values are sufficient for smooth or diffusive flows.
- Larger χ values are needed for sharp fronts or strong nonlinear effects.

Periodic truncation keeps χ bounded while limiting the approximation error.

6.4 Scalability and Parallelism

MPS–MPO contractions are local operations that can be parallelized over tensor sites, and truncation steps can use optimized parallel linear algebra routines. This locality also makes the approach adaptable to high-performance computing environments and potentially to quantum-hardware-inspired implementations.

7 Algorithm Comparison - Discuss trade-offs between QTN and HSE

In this section, we do a high-level comparison of QTN and HSE approaches. In addition, we compare the computational complexity, and also the strenghts and limitations of both methods. The conceptual differences are the following:

- **MPS Approach**: Solves the Navier-Stokes equations directly using tensor network compression. Fields like velocity and pressure are encoded as matrix-product states.
- **HSE Approach**: Reformulates fluid dynamics into a unitary evolution of a two-component wavefunction. Nonlinearity and vorticity are embedded in the potential term.

In short, both approaches offer innovative pathways to simulate fluid dynamics using quantum principles. The MPS method is well-suited for near-term classical acceleration, while the HSE framework holds promise for future quantum-native CFD simulations. Their complementary strengths suggest potential for hybrid strategies as quantum computing matures.

Table 1: High-level comparison of MPS and HSE approaches

Feature	**Matrix-Product States (MPS)**	**Hydrodynamic Schrödinger Equation (HSE)**
Origin	Tensor networks from quantum many-body physics	Generalized Madelung transform from quantum mechanics
Equation Solved	Incompressible Navier-Stokes (NSE)	Schrödinger-like equation for fluid flow (HSE/IHSE)
Computational Platform	Classical simulation with compressed representations	Quantum computing
Compression Strategy	Matrix-product representation of fields	Wavefunction encoding with spinor/quaternion structure
Boundary Conditions	Embedded via matrix-product differential operators	Implicit via wavefunction evolution and gauge projection
Sampling/Output	Pixel sampling and coarse-grained evaluation	Measurement of wavefunction post-evolution
Scalability	Polynomial in bond dimension, logarithmic in mesh size	Potential exponential speedup via quantum circuits
Validation	Compared with Ansys Fluent	Compared with classical DNS of NSE

Table 2: Computational complexity comparison

Task	**MPS Complexity**	**HSE Complexity**
Mask Generation	$O(Nx^3)$	N/A
Divergence-Free Projection	$O(Nx^6)$	Gauge transformation (bottleneck)
Time Stepping	$O(x^6)$	$O(n^2)$ quantum gates
Sampling	$O(x^3)$	Measurement via tomography/statistics
Overall Speedup	Classical compression	Potential exponential quantum speedup

Table 3: Strengths and limitations

Aspect	**MPS**	**HSE**
Strengths	Efficient classical compression; integrates well with existing CFD solvers	Quantum-native formulation; potentially scalable with future quantum hardware
Limitations	Constrained by classical memory and may lose fine-scale turbulence features	Depends on quantum hardware; normalization and gauge steps remain algorithmic bottlenecks

8 Validation & Benchmark

We compare the solutions obtained from three methods: the classical RK45 method, the Quantum Tensor Network (QTN) method and Quantum Physics Informed Neural Networks (QPINN). The results shown in Figure 1 indicate that the three solutions are almost the same. The first method is the classical RK45 solver, the second method is the quantum-inspired QTN, and the final method is QPINN, which uses a quantum circuit with a classical optimizer.

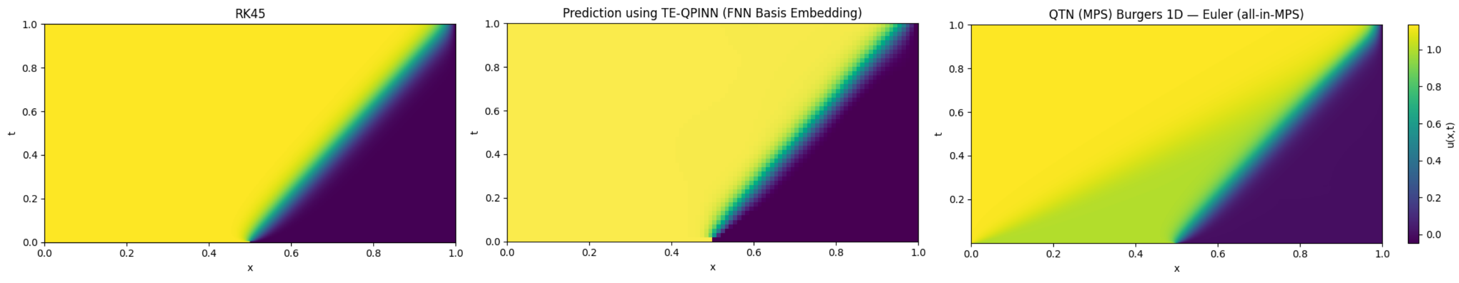

Figure 1: Burgers' Equation Shock Tube u(x,t) comparison between RK45, QPINN and QTN.

Figure 2 presents the comparison of the L_2 -error between the RK45 and QTN methods in solving the 1D Shock Tube Burgers' equation. The error starts at around 2% at t=0 and increases steadily, reaching approximately 14% at t=1. This trend indicates that while QTN initially matches RK45 closely, discrepancies accumulate as time progresses, likely due to differences in numerical discretization (Euler vs. RK45) and truncation/compression in the MPS representation. The monotonic growth suggests a consistent propagation of numerical error without significant oscillations.

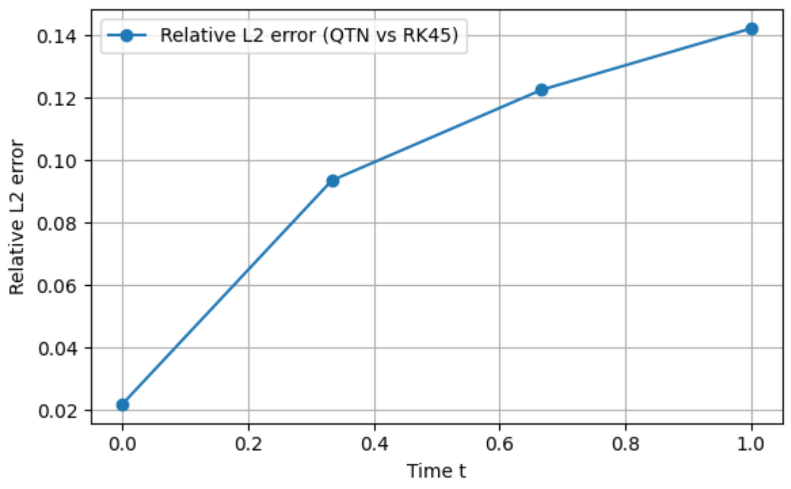

Figure 2: Compare L2-error between RK45 and QTN.

References

[1] R. D. Peddinti, S. Pisoni, A. Marini, P. Lott, H. Argentieri, E. Tiunov, and L. Aolita, ”Quantum-inspired framework for computational fluid dynamics,” *Commun. Phys.*, vol. 7, no. 1, p. 135, 2024.

[2] Z. Meng and Y. Yang, ”Quantum computing of fluid dynamics using the hydrodynamic Schrödinger equation,” *Phys. Rev. Res.*, vol. 5, no. 3, p. 033182, 2023.