Project 3: Quantum Algorithm as a PDE Solver for Computational Fluid Dynamics (CFD)

Task

Solve the **1-D Burgers' Equation with Shock Tube**:

$$\frac{\partial u}{\partial t} + \frac{u \partial u}{\partial x} = \frac{\nu \partial^2 u}{\partial x}$$

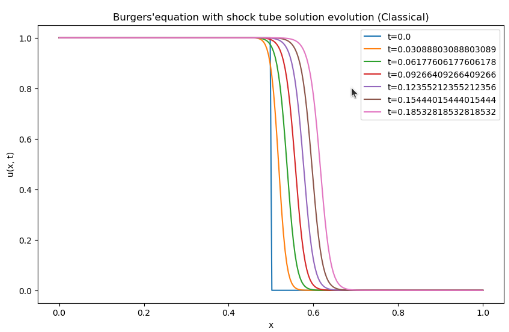
Domain: $x \in [0,1]$
IC: Riemann step $u(x, 0) = 1$ for $x \leq 0.5$, 0 otherwise
BC (Dirichlet): $u(0, t) = u_L$, $u(L, t) = u_R$ for all $t > 0$

Instruction:

This open challenge tasks participants with designing and prototyping resource-lean quantum-enhanced PDE solvers based on either **Quantum Tensor-Network (QTN)** or **Hydrodynamic Shrödinger Equation (HSE)**; **hybrid QTN-HSE** approaches are also welcome.

Validation & Benchmark

In order to assess my quantum-inspired algorithm for solving 1-D Burgers' equation, I designed another algorithm that solve the equation classically, using explicit upwind Scheme with Central Diffusion (Finite Difference Method). Below are the numerical solutions in both cases Figure 1:



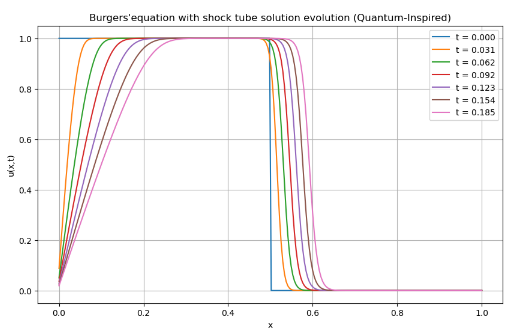


Figure 1: Numerical solution for Burgers' equation: Classical (left) vs Quantum (right)

where I defined exactly the same parameters for both algorithms (number of space grid points = 256, number of time steps = 260, viscosity = 0.005).

Below is a table summarizing some aspects between the two approaches Table 1:

Table 1: Comparison between Classical and Quantum-Inspired solving of the 1-D Burgers' equation with shock tube.

Aspect	**Classical Solving**	**Quantum-Inspired Solving**
Representation of the wavefunction	Represented as a large array of floating-point values on a classical grid.	Encoded into Matrix Product States.
Time evolution	Performed by numerically integrating the PDE using finite-difference methods.	Implemented through Matrix Product state evolution via explicit Euler scheme.
Numerical errors	Floating-point truncation and discretization errors.	Quantum gate errors, decoherence, and measurement noise.
Hardware requirement	CPU/GPU clusters with large RAM for high resolution.	Fault-tolerant quantum processors with sufficient qubits and low error rates.
Execution time	a few second.	about 12 min.

Remark: The huge amount of execution time for the quantum-Inspired algorithm is due to the reconversion of matrix product states to dense vectors. Unfortunately, due to very busy schedules during project period, I did not have time to implement a process of getting the solution directly from the matrix product state (probably via coarse-grained evaluation or pixel sampling (Peddinti et al., Commun. Phys. 7, 135, 2024))