Algorithm Design Brief: Quantum-Enhanced PDE Solver for 1D Burgers' Equation via HSE

Nikolaos Makris

August 6, 2025

1 Chosen Framework: Hydrodynamic Schrödinger Equation (HSE)

Our quantum-enhanced PDE solver for the 1D viscous Burgers' Equation will leverage the **Hydrodynamic Schrödinger Equation (HSE)** framework. This approach is chosen for its ability to transform a challenging nonlinear classical PDE into a linear, quantum-simulable form, thereby potentially unlocking exponential state-space compression and polynomial speed-ups. The HSE framework, as described by Meng & Yang (2023), recasts incompressible flow dynamics into a quantum wave-function evolution, making it amenable to universal quantum processors.

2 Mapping of 1D Burgers' Equation to HSE

The 1D viscous Burgers' Equation is given by:

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

This equation is nonlinear due to the $u\frac{\partial u}{\partial x}$ term, which poses a significant challenge for direct quantum simulation, as quantum mechanics is inherently linear. The HSE framework addresses this by employing the **Cole-Hopf transformation**:

$$u(x,t) = -2\nu \frac{\partial}{\partial x} \ln \phi(x,t)$$

Substituting this transformation into the Burgers' equation linearizes it, yielding the **linear diffusion equation** for the auxiliary field $\phi(x,t)$:

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

This diffusion equation is mathematically analogous to the free-particle Schrödinger equation in imaginary time, making its time evolution directly implementable on a quantum computer.

Discretization and Encoding:

1. **Spatial Discretization:** The domain $x \in [0,1]$ is discretized into N_x spatial grid points. For this challenge, we start with a coarse grid of $N_x = 16$ points. The spatial step size is $\Delta x = \frac{1}{N_x - 1}$.

2. **Initial Condition (IC) for ϕ:** The initial Riemann step for $u(x,0)$ ($u(x,0)=1$ for $x \leq 0.5$, 0 otherwise) is transformed to $\phi(x,0)$ using the integral form of the Cole-Hopf transformation:

$$\phi(x,0) = A \exp\left(-\frac{1}{2\nu}\int_{x_0}^{x} u(\xi,0)d\xi\right)$$

where A is an arbitrary normalization constant (typically $A=1$). This ensures $\phi(x,0)$ is continuous across the jump in $u(x,0)$.

3. **Boundary Conditions (BC) for ϕ:** The Dirichlet boundary conditions for $u(x,t)$ at $x=0$ and $x=L$ translate to Neumann-like boundary conditions for $\phi(x,t)$ (i.e., conditions on $\frac{\partial\phi}{\partial x}$). These are implicitly handled by the time evolution operator and the initial state preparation.

4. **Quantum State Encoding:** The discretized values of $\phi(x,t)$ at the N_x grid points are encoded into the amplitudes of a quantum state. For $N_x = 16$, we require $\log_2(16) = 4$ qubits. The state is represented as $|\Psi(t)\rangle = \sum_{j=0}^{N_x-1} \phi(x_j,t)|j\rangle$, where $|j\rangle$ is a computational basis state representing the grid point x_j. This is achieved using **amplitude encoding** via the `StatePreparation` circuit.

3 Gate Decomposition

The quantum algorithm consists of three main parts: initial state preparation, time evolution, and measurement/post-processing.

3.1 Initial State Preparation

The initial state $|\Psi(0)\rangle$ representing $\phi(x,0)$ is prepared using Qiskit's `StatePreparation` circuit. This circuit takes the normalized classical vector $\phi(x,0)$ as input and constructs a quantum circuit that prepares the corresponding state. The decomposition of this circuit depends on the specific state vector, but Qiskit internally optimizes it using various techniques (e.g., using Aharonov-Landau-Lo-Popescu (ALPS) decomposition or other synthesis methods).

3.2 Time Evolution of $\phi(x,t)$

The time evolution of $\phi(x,t)$ is governed by the diffusion equation $\frac{\partial\phi}{\partial t} = \nu\nabla^2\phi$. This can be viewed as a quantum Hamiltonian simulation problem where the Hamiltonian $H = -i\nu\nabla^2$. We need to implement the unitary operator $U(t) = e^{t\nu\nabla^2}$.

1. **Discretized Laplacian as Hamiltonian:** The 1D discrete Laplacian operator for N_x points with spacing Δx can be represented as an $N_x \times N_x$ matrix. For central differences, it has $-2/(\Delta x)^2$ on the diagonal and $1/(\Delta x)^2$ on the off-diagonals (nearest neighbors). This matrix can be mapped to a sum of Pauli operators (Pauli strings) acting on the qubits. For a 4-qubit system, this would involve terms like $I \otimes I \otimes Z \otimes Z$, $X \otimes X \otimes I \otimes I$, etc., representing interactions between qubits.

2. **Trotter-Suzuki Decomposition:** Since the Hamiltonian H is a sum of non-commuting terms, we use the first-order Trotter-Suzuki decomposition:

$$e^{Ht} \approx \left(\prod_k e^{H_k t/N_{\text{steps}}}\right)^{N_{\text{steps}}}$$

where $H = \sum_k H_k$, and H_k are terms that are easier to exponentiate (e.g., individual Pauli strings).

3. **Gate Implementation of Pauli Exponentials:** Each term $e^{H_k \Delta t_{\text{trotter}}}$ (where $\Delta t_{\text{trotter}} = t/N_{\text{steps}}$) is implemented using a sequence of single-qubit rotations and CNOT gates. For example, $e^{-i\theta P}$ where P is a Pauli string (e.g., $Z_0 Z_1$) can be implemented by:
   - Transforming the qubits into a basis where P is diagonal (e.g., using Hadamard gates for X or Y terms).
   - Applying a controlled-phase gate (e.g., `Rz` for Z terms, or a sequence of `CNOT` and `Rz` for multi-qubit Pauli strings).
   - Transforming back to the original basis.

   Illustrative Example (Placeholder for a specific Laplacian decomposition): For a 4-qubit system representing 16 spatial points, a simplified Trotter step might involve terms representing nearest-neighbor interactions. For instance, a term like $Z_i Z_{i+1}$ (representing a part of the discretized Laplacian) would be implemented as: `qc.cx(i, i+1)`

   ```
   qc.rz(angle, i+1)
   qc.cx(i, i+1)
   ```

   This sequence implements a controlled-Z rotation. The `angle` would be proportional to $\nu \Delta t_{\text{trotter}}$ and the coefficient of the corresponding Pauli term in the Hamiltonian. Our prototype uses `rx` and `cx` gates as a generic representation of such local interactions.

3.3 Measurement and Post-processing

After time evolution, the quantum state $|\Psi(t)\rangle$ holds the amplitudes $\phi(x_j, t)$.

1. **Statevector Extraction:** For simulation, we can directly extract the statevector using `qc.save_statevector()`. On real hardware, one would perform quantum state tomography (which is resource-intensive) or more likely, measure in the computational basis multiple times and reconstruct the amplitudes from the probabilities.

2. **Inverse Cole-Hopf Transformation:** The extracted $\phi(x, t)$ vector is then used to compute $u(x, t)$ classically using the inverse Cole-Hopf transformation:

$$u(x,t) = -2\nu \frac{1}{\phi(x,t)} \frac{\partial \phi(x,t)}{\partial x}$$

   The spatial derivative $\frac{\partial \phi}{\partial x}$ is computed using a classical finite difference approximation (e.g., central difference).

4 Resource Estimates

For the 1D Burgers' equation on a 16-point grid ($N_x = 16$):

- **Qubit Footprint:**
  - Number of qubits: $\log_2(N_x) = \log_2(16) =$ **4 qubits**. This is a significant compression compared to classical methods that store N_x values.
- **Circuit Depth:**

  - **State Preparation:** The depth depends on the complexity of the initial $\phi(x,0)$ vector. Qiskit's `StatePreparation` can have a depth that scales polynomially with N_x (or exponentially with number of qubits in worst case), but for specific states, it can be more efficient.
  - **Time Evolution:** The depth scales with the number of Trotter steps (N_{steps}) and the number of Pauli terms in the Hamiltonian decomposition. If the Hamiltonian has M Pauli terms, and each term requires D_{term} gates, the depth is approximately $N_{\text{steps}} \times M \times D_{\text{term}}$. For a 1D nearest-neighbor Laplacian, M scales linearly with N_x, and D_{term} is constant. Thus, the total depth scales as $O(N_{\text{steps}} \cdot N_x)$.
  - **Total Depth (Prototype):** For our prototype's illustrative Trotter step (which involves 4 RX gates and 3 CNOTs per step), with $N_{\text{steps}} = 10$, the depth is approximately $10 \times$ (depth of one step).

- **Two-qubit Gate Count:** Scales similarly to depth.
- **T-count:** Depends on the specific rotations and decompositions used. If arbitrary single-qubit rotations are used, they can be decomposed into a few T-gates and Clifford gates. The T-count can be significant for high-fidelity rotations.
- **Classical Pre/Post-processing Overhead:**
  - **Pre-processing:** Calculating $\phi(x,0)$ from $u(x,0)$ involves a numerical integration and normalization, which is $O(N_x)$.
  - **Post-processing:** Numerical differentiation and inverse Cole-Hopf transformation is $O(N_x)$.
  - This overhead is minimal compared to the potential quantum speed-up for larger N_x and higher dimensions.

Scalability Implications:

The logarithmic scaling of qubits ($O(\log N_x)$) is the primary advantage. The linear scaling of depth with N_x (for 1D) and number of Trotter steps means that for very large N_x, depth could still be a challenge for near-term devices. However, this is a significant improvement over classical methods which scale polynomially with N_x in memory and computation time.

References

[1] Meng, Y., Yang, G. (2023). Quantum Simulation of Incompressible Fluid Dynamics. Phys. Rev. Research 5, 033182.

[2] Peddinti et al. (2024). Quantum Tensor Network CFD. Commun. Phys. 7, 135.

[3] Qiskit Team. (2024). Qiskit SDK. `https://qiskit.org`

[4] Mitiq: Error Mitigation Toolkit. `https://github.com/unitaryfund/mitiq`