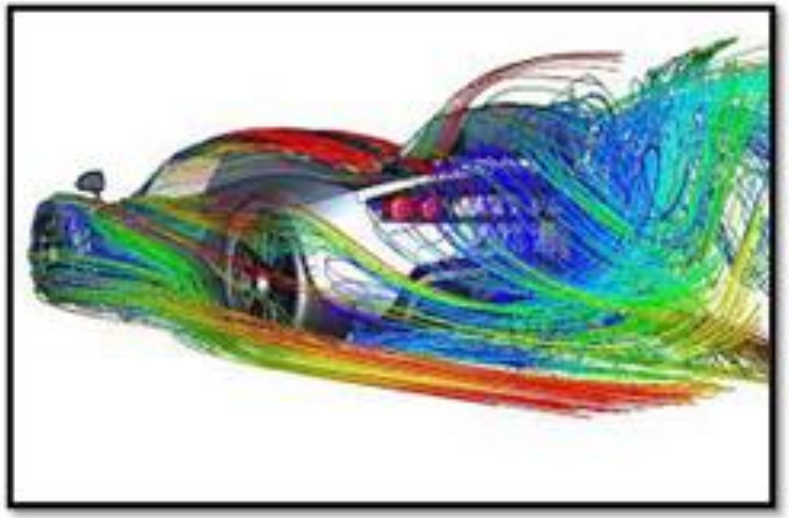
- Can Quantum Computer speed up Fluid Dynamic simulations?



Mr. Tejas Shinde

- Looking for PhD position
- Currently : Working student at TH Deggendorf building a Teaching Data center (Lehrrechenzentrum)
  Master thesis – Lattice Boltzmann method using the Intel Quantum SDK.
- 2021 - 2023 : Master of Science in High Performance Computing/Quantum Computing ,
  Technische Hochschule Deggendorf
- 2020 : Intern, Computer Aided Engineering at Simulation Lab Pvt Ltd, India
  Multi-Feature on wing
- 2015 - 2019 : Bachelor's in Mechanical Engineering , Pune University , India

TECHNISCHE HOCHSCHULE DEGGENDORF THD

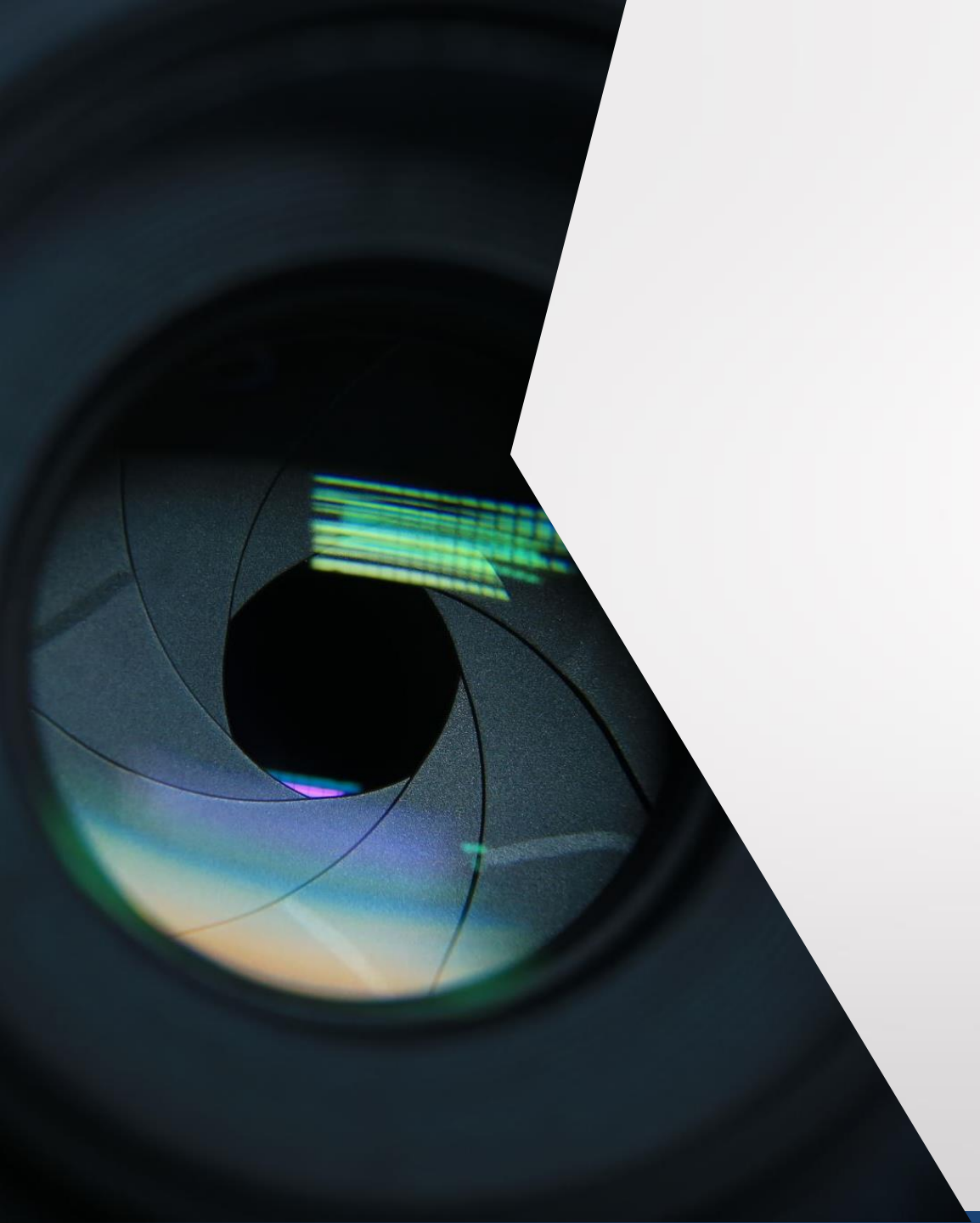
SAVITRIBAI PHULE PUNE UNIVERSITY
॥ यः क्रियावान् स पण्डितः॥

3

21st International Workshop on Advanced Computing and Analysis Techniques in Physics Research

DEGGENDORF INSTITUTE OF TECHNOLOGY

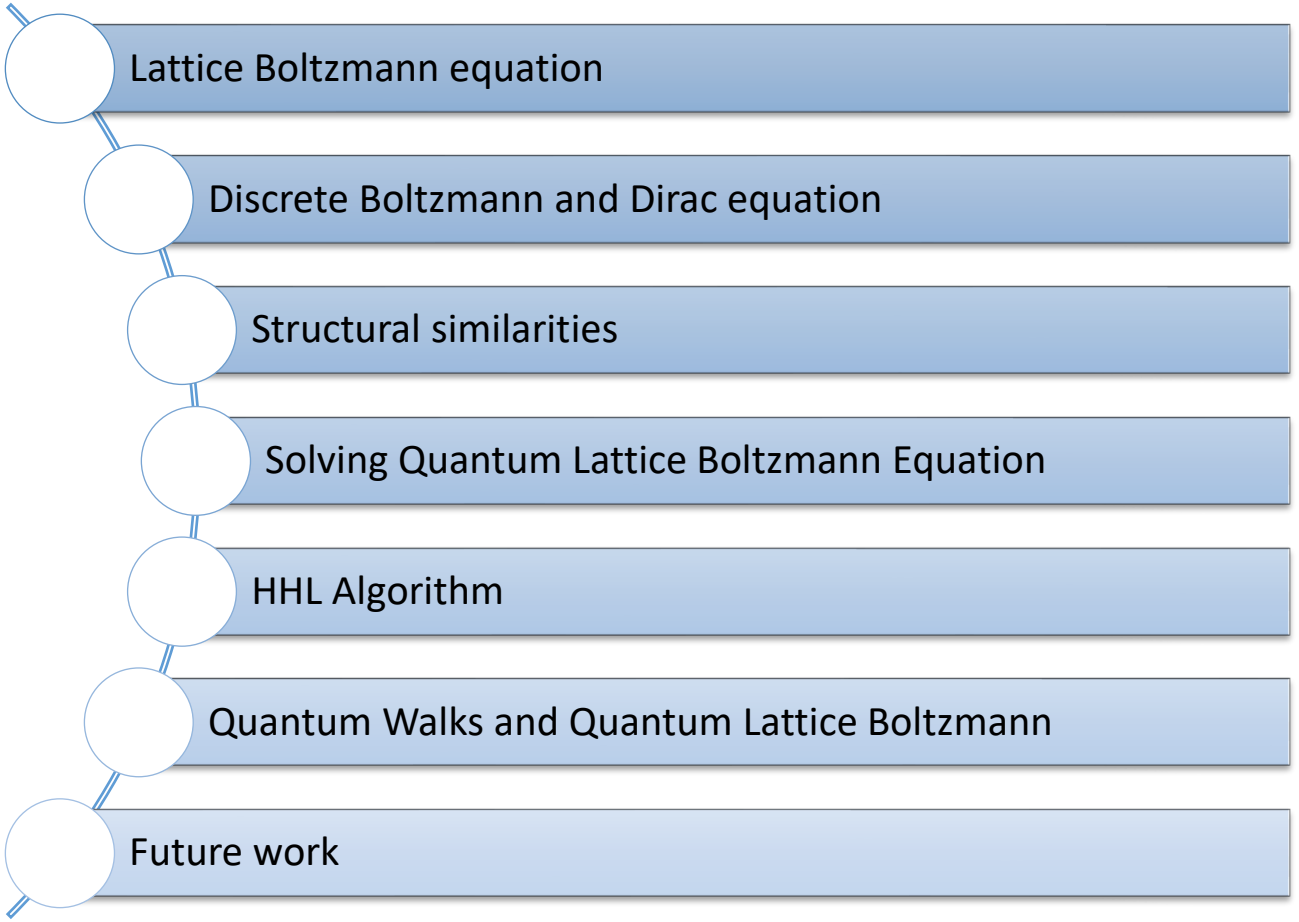
PRELIMINARY LATTICE BOLTZMANN EQUATION USING INTEL QUANTUM SDK



Objective

Solve a simple basic Fluid Dynamics problem using Lattice Boltzmann method on the Intel Quantum SDK

Agenda


Lattice Boltzmann equation
Discrete Boltzmann and Dirac equation
Structural similarities
Solving Quantum Lattice Boltzmann Equation
HHL Algorithm
Quantum Walks and Quantum Lattice Boltzmann
Future work

LATTICE BOLTZMANN METHOD

- Widely used in range of Complex fluid problems
- Basic quantities are the Discrete velocity distribution functions f_i
- f_i -> calculate density and velocity through moments

Boltzmann Equation (force-free)

$$\frac{\partial f}{\partial t} + \xi_i \frac{\partial f}{\partial x_i} = -\frac{(f - f^{eq})}{\tau}$$

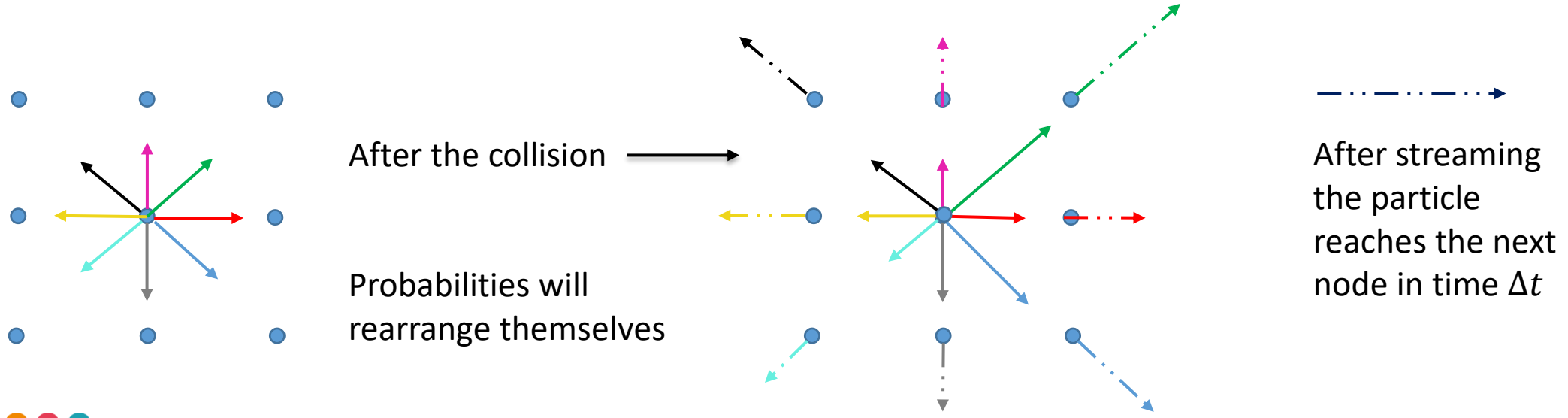
DISCRETIZED EQUATION SOLVED IN 2 STEPS

1- Collision Step – post collision distribution function

$$f_\alpha^*(\vec{x},t) = f_\alpha(\vec{x},t) - \frac{\Delta t}{\tau}(f_\alpha(\vec{x},t) - f_\alpha^{eq}(\vec{x},t))$$

2 – Streaming (Propagation) – Post collision function is transferred into post streaming function

$$f_\alpha^*(\vec{x} + \vec{c}_\alpha \Delta t, t + \Delta t) = f_\alpha^*(\vec{x},t)$$



DISCRETE BOLTZMANN AND DIRAC EQUATION

- Interesting Formal Analogy between fluid and quantum mechanics
- Exporting assets from the LBE to Quantum world
- Quantum Lattice Boltzmann equation is derived from this analogy

STRUCTURAL SIMILARITIES

The Lattice Boltzmann equation is given as -

$$(\partial_t + v_i{}^x \partial_x + v_i{}^y \partial_y + v_i{}^z \partial_z) f_i = \sum_{j=0}^{n} \Omega_{ij} (f_j - f_j{}^0)$$

The Dirac equation is given as

$$(\partial_t + c\alpha^x \partial_x + c\alpha^y \partial_y + c\alpha^z \partial_z)\psi = (-i\, m\, \beta + i\, g\, I)\psi$$

- Similarities

1 – Both equations contain first order derivatives in Space and time

2 – $f_i \rightarrow \psi_i$

3 – $\vec{v}_i = (v_i^x, v_i^y, v_i^z) \rightarrow \alpha^k = (\alpha^x, \alpha^y, \alpha^z)$

4 - $\Omega_{ij} \rightarrow \beta$

In one dimension – analogy exact

In multi dimension – operator splitting to be used

Where :-

c -> speed of light

$\hbar$ - > reduced planck's constant

ω_c -> $m c^2 / \hbar$ - Compton frequency of particle
m -> mass of particle

g -> $q * V / \hbar$ - couples the wave function with the potential V and q being electric charge

$\psi = \psi(x, t)$ - > wave function that describe the complex four-dimensional spinor (4-spinor)

$\alpha^x, \alpha^y, \alpha^z, \beta$ are the 4 * 4 Hermitian matrices and commonly referred as Dirac matrices

1D QUANTUM LATTICE BOLTZMANN SCHEME

The one dimensional Dirac equation can be written as –

$$(\partial_t + \alpha^x \, \partial_x)\psi = (-i \, \omega_c \, \beta + i \, g \, I)\psi$$

- $X^{-1} \, \alpha^x \, X = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 1 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} = \beta$

- $X^{-1} \, \alpha^x \, X = \beta$

- Therefore , multiplying the one-dimensional equation from left by X^{-1} and inserting identity matrix - $I = XX^{-1}$

$$X^{-1} \, (\partial_t + \alpha^x \, \partial_x) XX^{-1}\psi = X^{-1}\big(-i \, m \, \beta + i \, g \, I\big)XX^{-1}\psi$$

$$(\partial_t + \underbrace{X^{-1}\alpha^x X}_{\beta}\partial_x) \, X^{-1}\psi = \underbrace{X^{-1}(-i \, m \, \beta + i \, g \, I)X}_{Q} \, X^{-1}\psi$$

$$(\boldsymbol{\partial_t + \beta \, \partial_x}) \, \boldsymbol{X^{-1}\psi = Q \, X^{-1}\psi} \quad \text{- > } Q = \begin{pmatrix} ig & 0 & 0 & -im \\ 0 & ig & im & 0 \\ 0 & im & ig & 0 \\ -im & 0 & 0 & ig \end{pmatrix}$$

1D QUANTUM LATTICE BOLTZMANN SCHEME – CONT'D

$$(\boldsymbol{\partial_t} + \boldsymbol{\beta}\,\boldsymbol{\partial_x})\,\boldsymbol{X^{-1}\psi} = \boldsymbol{Q}\,\boldsymbol{X^{-1}\psi}$$

To write this matrices and equation in the form of partial differential equations , following is done -

Define rotated wave function - $\boldsymbol{X^{-1}\psi} = (u_1, u_2, d_1, d_2)^T$

u and d propagating up and down the x-axis and subscripts 1,2 indicate spin states

Therefore, the partial differential equations we get are as follows : -

$$\partial_t u_1 + \partial_x u_1 = igu_1 - imd_2$$

$$\partial_t u_2 + \partial_x u_2 = igu_2 + imd_2$$

$$\partial_t d_1 - \partial_x d_1 = igd_1 + imu_2$$

$$\partial_t d_2 - \partial_x d_2 = igd_2 - imu_1$$

SOLVING QUANTUM LATTICE BOLTZMANN EQUATION

- Few algorithms exist which solve some problems such as

❖ HHL

❖ Quantum walks

❖ Variational Quantum Eigensolver

❖ Hamiltonian Simulation Techniques

DISCRETIZED 1D-QLB AND MAPPING INTO HHL

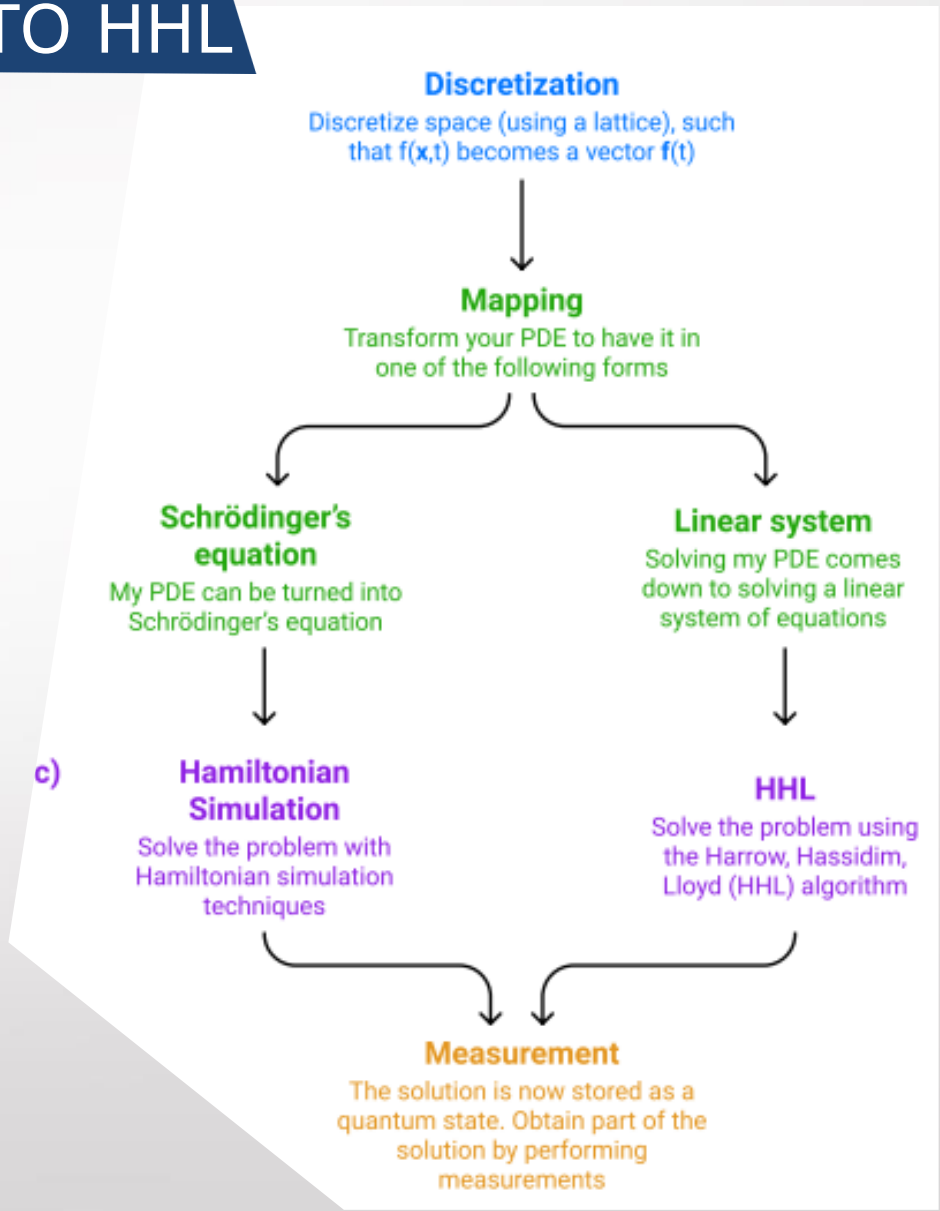
- Solving the Discretized qLB equation with HHL.
- Discretized equation

$$\widehat{u_1} - u_1 = \frac{1}{2} i\tilde{g} (\widehat{u_1} + u_1) + \frac{1}{2} i\tilde{m} (\widehat{d_2} + d_2)$$
$$\widehat{u_2} - u_2 = \frac{1}{2} i\tilde{g} (\widehat{u_2} + u_2) + \frac{1}{2} i\tilde{m} (\widehat{d_1} + d_1)$$
$$\widehat{d_1} - d_1 = \frac{1}{2} i\tilde{g} (\widehat{d_1} + d_1) - \frac{1}{2} i\tilde{m} (\widehat{u_2} + u_2)$$
$$\widehat{d_2} - d_2 = \frac{1}{2} i\tilde{g} (\widehat{d_2} + d_2) - \frac{1}{2} i\tilde{m} (\widehat{u_1} + u_1)$$

HHL algorithms that solves System of Linear equations

Ax = b - form

A – Hermitian matrix



- Isotropy of three-dimensional quantum lattice Boltzmann schemes - P. J. Dellar* and D. Lapitski S. ` S. Succi
- Quantum Algorithms for Solving Partial Differential Equations Arthur Pesah
- Step-by-Step HHL Algorithm Walkthrough to Enhance the Understanding of Critical Quantum Computing Concepts

HHL IN INTEL QUANTUM SDK

- Currently, solving a group of linear equation in Intel Quantum SDK using HHL algorithm.

- Intel Quantum SDK follow a Kernel based programming similar to other hardware accelerator environments

Developing Software for the Quantum Era

Intel Labs Releases Beta Version of the Intel Quantum Software Development Kit

Sept. 28, 2022 — Quantum computing promises to dramatically speed up complex problem-solving and has the potential to enable significant breakthroughs in materials, chemicals and drug design, financial and climate modeling, and cryptography. Advances in quantum bits, or qubits, are one step toward achieving quantum practicality, but significant breakthroughs are needed across the full hardware and software stack to realize its full potential.

To advance this journey, Intel Labs has developed a full-stack Software Development Kit (SDK), called the Intel® Quantum SDK, that interfaces with Intel's quantum computing stack. The kit allows developers to program new quantum algorithms for executing qubits in simulation and on real quantum hardware in the future. Beta users are already exploring chemistry, materials and fluid dynamics simulations, as well as algorithms for solving linear systems of equations, which could be used in real-life situations such as financial modeling.

```
#include <clang/Quantum/quintrinsics.h>

//your quantum data types
//define qubit registers

quantum_kernel void kernel_1(){
     ...
}


quantum_kernel void kernel_2(){
     ...
}

quantum_kernel void kernel_3(){
     kernel_1();
     kernel_2();
}

int main(){
     ...
     kernel_3();
     ...
     return 0;
}
```


- Khalate, P., Wu, X., Premaratne, S., Hogaboam, J., Holmes, A., Schmitz, A., Guerreschi, G. G., Zou, X. & Matsuura, A. Y., https://arxiv.org/abs/2202.11142
- https://download.intel.com/newsroom/2022/2022innovation/quantum-sdk-backgrounder.pdf

QUANTUM WALKS

- The discretization techniques of qLBE fall into the category of Quantum Walks
- Transport phenomena problems could be solved

- Evolution operator –

$$\boldsymbol{\psi}_{j+1} = \widehat{W}\,\boldsymbol{\psi}_j$$

$$\widehat{W} = \widehat{S}\,(\hat{C})$$

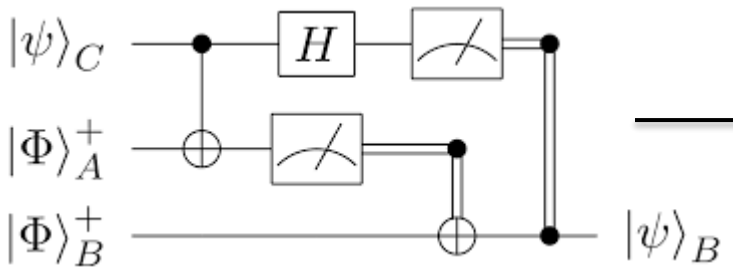
So – S – conditional shift operator

C – Coin operator

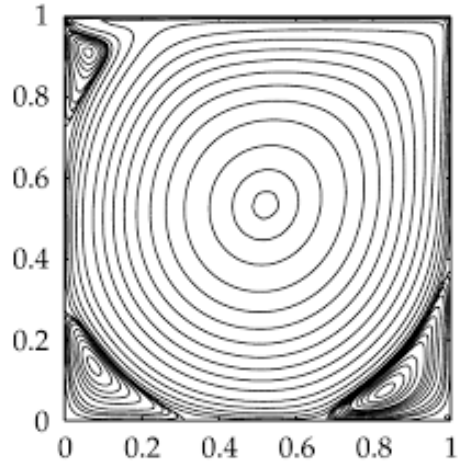
- Quantum Lattice Boltzmann is a quantum walk - Sauro Succi1* , Fran¸cois Fillion-Gourdeau3 and Silvia Palpacelli2
- Towards quantum lattice-Boltzmann methods - Giuseppe Di Molfetta

NEXT STEPS

- Build Quantum Circuit for the Collision and Streaming of qLBM
- Implement the Quantum Circuit in the Intel Quantum SDK
- Finally, Solve a simple Fluid Dynamics problem using this Circuit.
- Validate the results.




```
const int N = 5;

/* global array of qubits*/
qbit qreg[N];
cbit creg[N];

quantum_kernel void cphase(qbit ctr, qbit tgt, double phase) {
  RZ(ctr, phase / 2);
  CNOT(ctr, tgt);
  RZ(tgt, -1 * phase / 2);
  CNOT(ctr, tgt);
  RZ(tgt, phase / 2);
}

quantum_kernel void prepare() {
  for (int i = 0; i < N; i++) {
    PrepZ(qreg[i]);
  }
}

quantum_kernel void measure() {
  for (int i = 0; i < N; i++) {
    MeasZ(qreg[i], creg[i]);
  }
}

quantum_kernel void QFT() {

  int n = N - 1;
```



ACKNOWLEDGEMENT

I would like to express my special thanks and gratitude to Intel Labs for supporting me in this wonderful project

THANK YOU

Backup

- The Dirac matrices are given in terms of 2 * 2 pauli matrices

$$\alpha^k = \begin{pmatrix} 0 & \sigma^k \\ \sigma^k & 0 \end{pmatrix} \; and \; \beta = \begin{pmatrix} I & 0 \\ 0 & -I \end{pmatrix}$$

σ^k -> Pauli matrices , k = x, y ,z

$$\sigma^x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad , \sigma^y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad , \sigma^z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Therefore the dirac matrices can be written as follows –

$$\alpha^x = \begin{pmatrix} 0 & \sigma^x \\ \sigma^x & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Similarly we can write the α^y , α^z , β Dirac Matrices

Backup slide

moments :-

Density - $\rho(\vec{x},t) = \sum_\alpha f_\alpha(\vec{x},t)$

Momentum Density $-$ $\rho(\vec{x},t)\vec{u}(\vec{x},t) = \sum_\alpha f_\alpha(\vec{x},t)\, c_\alpha$