TU Delft

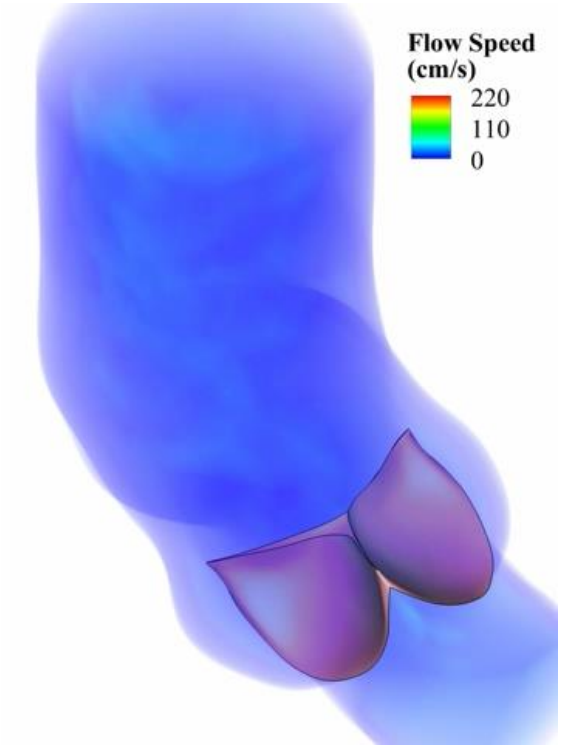
Matthias Möller

Opportunities and Challenges of Computational Fluid Dynamics Applications on Quantum Computers

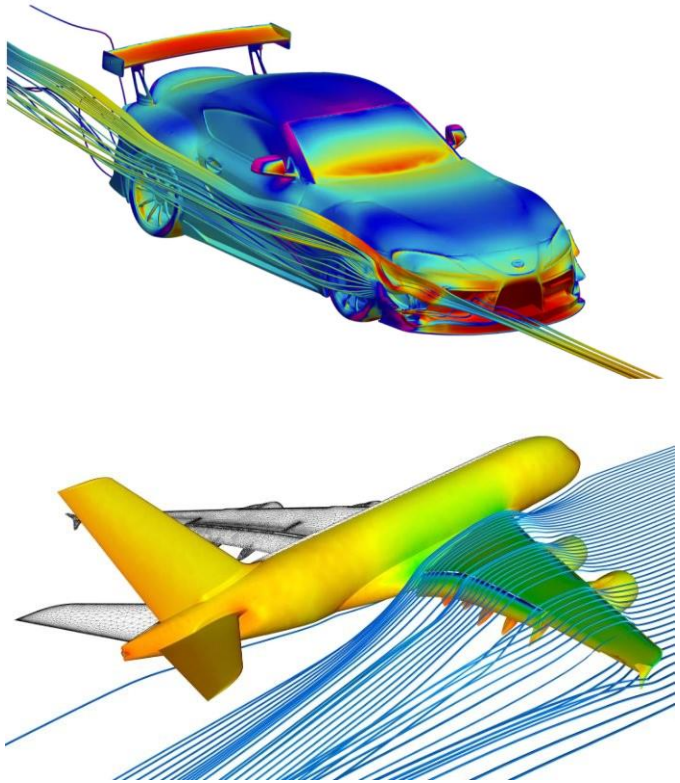
Fujitsu Quantum Day 28-03-2025

V0.1

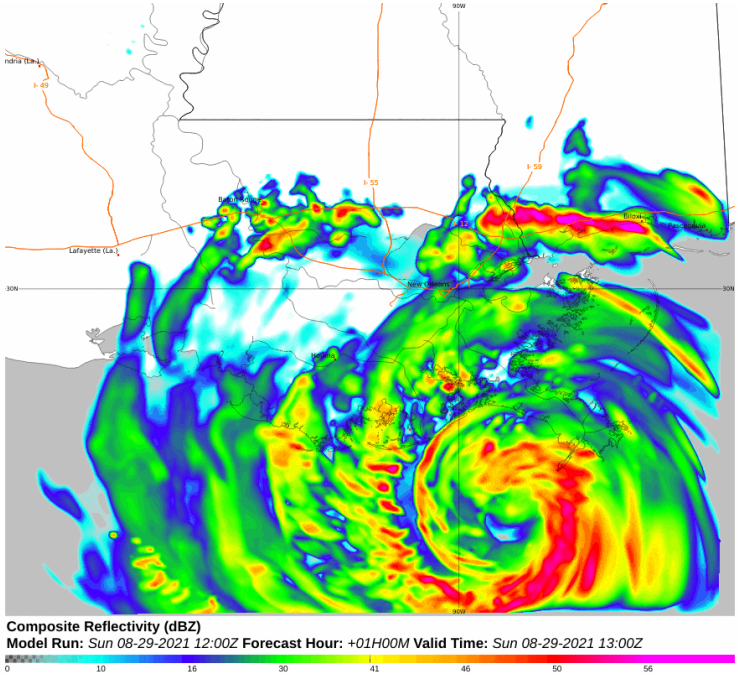
Computational fluid dynamics (CFD)



biomedical flows



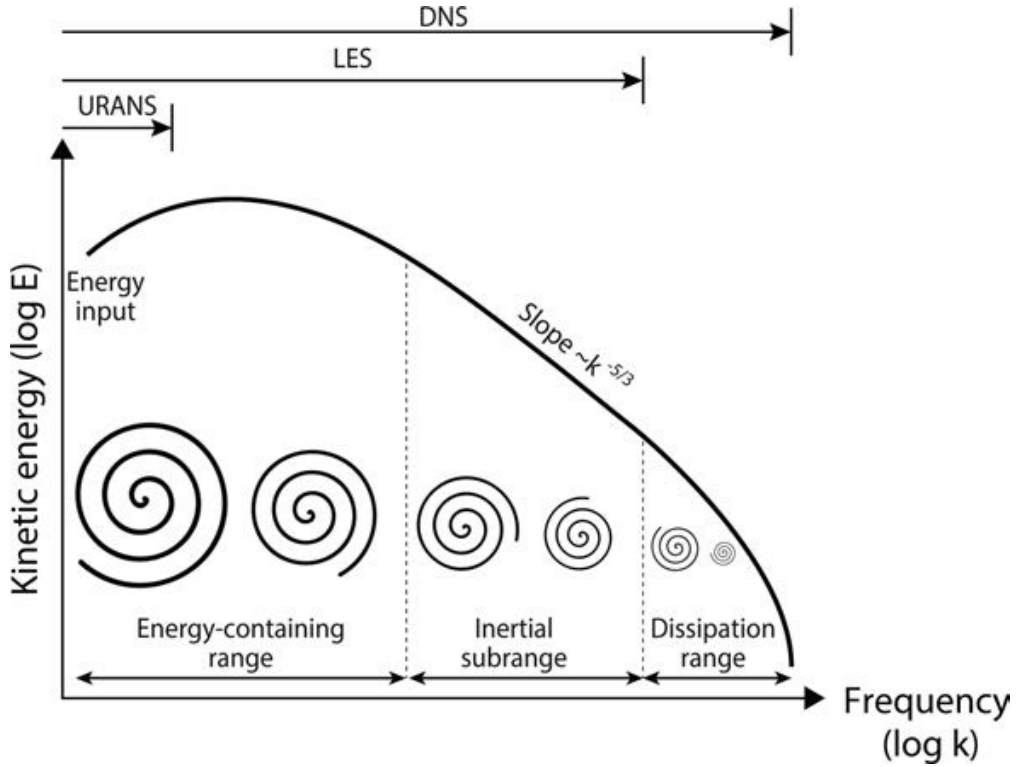
aerodynamics



climate/weather modelling

Left: DOI: 10.1002/cnm.2938, middle: https://berktuning.com, DLR, right: https://tempoquest.com

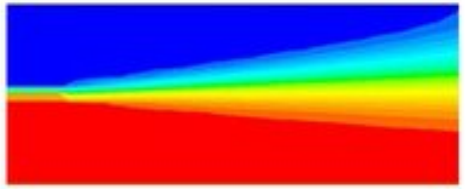
Turbulence

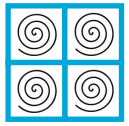


Left: DOI: 10.1007/978-981-33-6660-2_2

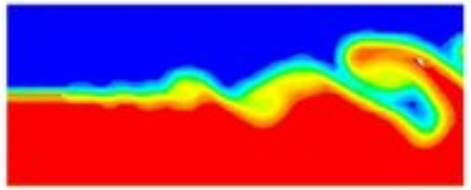
Turbulence

+ RANS =

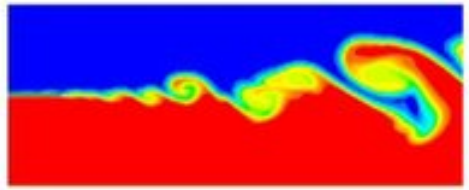




+ LES =

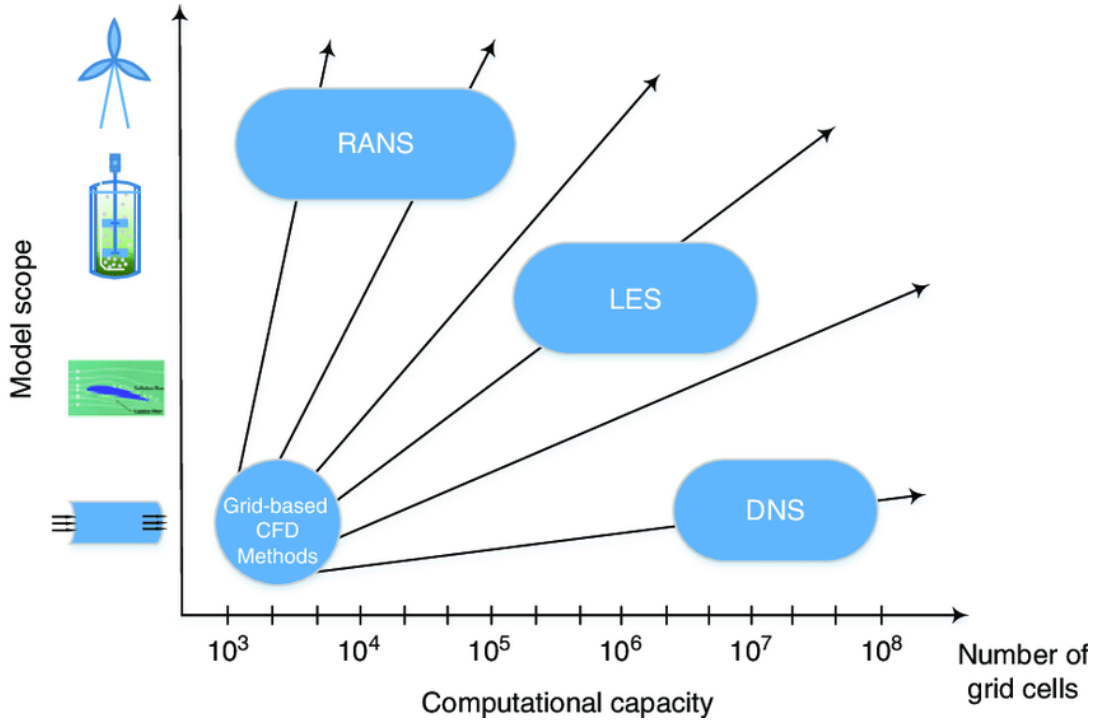


+ DNS =



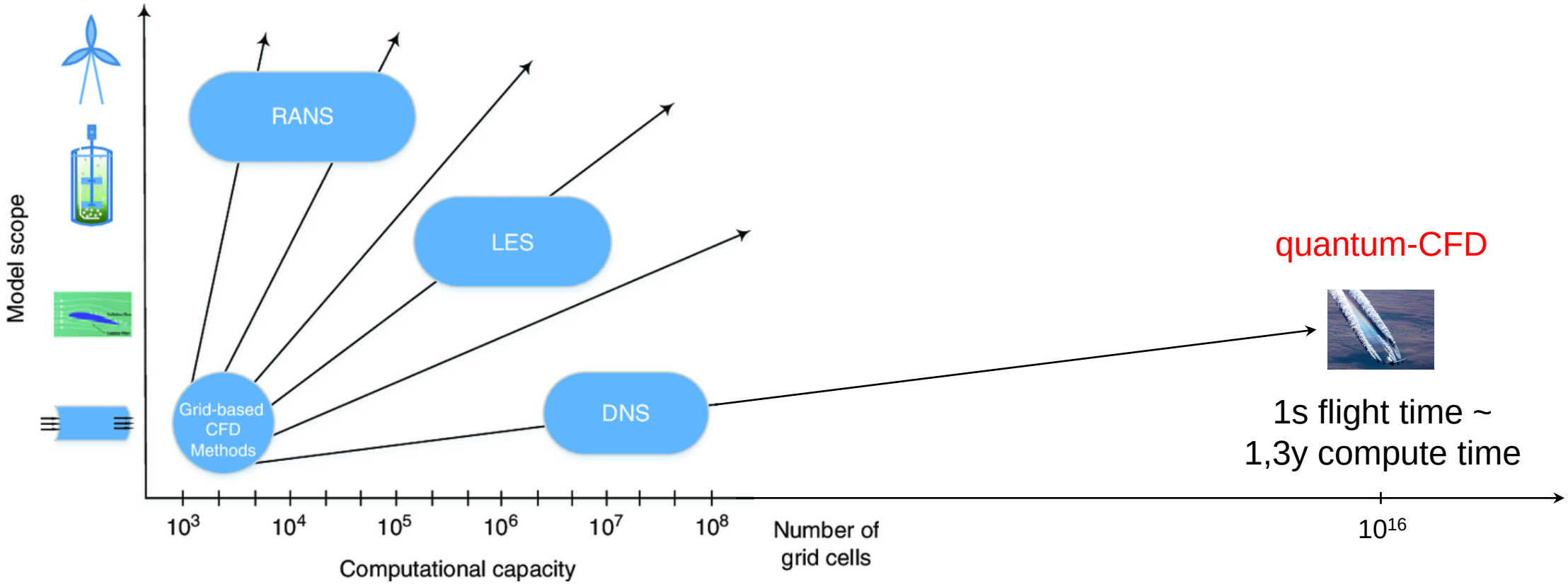
Left: DOI: 10.1007/978-981-33-6660-2_2, right: DOI: 10.1007/978-3-030-10588-4_5

Turbulence



Left: DOI: 10.1007/978-981-33-6660-2_2, right: DOI: 10.1016/j.csbj.2018.06.002

Turbulence



Left: DOI: 10.1016/j.csbj.2018.06.002, right: DOI: 10.1007/978-981-33-6660-2_2

The Quantum-CFD team

FUJITSU


Shinji Kikuchi


Masaya Kibune


Jiading Wang


Yutaka Takita

Mission: Create quantum-CFD applications for fault-tolerant quantum computers

TU Delft


Matthias Möller


Calin Georgescu


Merel Schalkers


Monica Lacatus


Alex Sturges

Quantum-CFD

Opportunities

- Reduction in memory requirements
  $O(N_g)$ grid points $\rightarrow O(\log_2 N_g)$ qubits
- Reduction in compute operations
  $O(N_t N_g)$ flops $\rightarrow O(N_t \log_2 N_g)$ qops

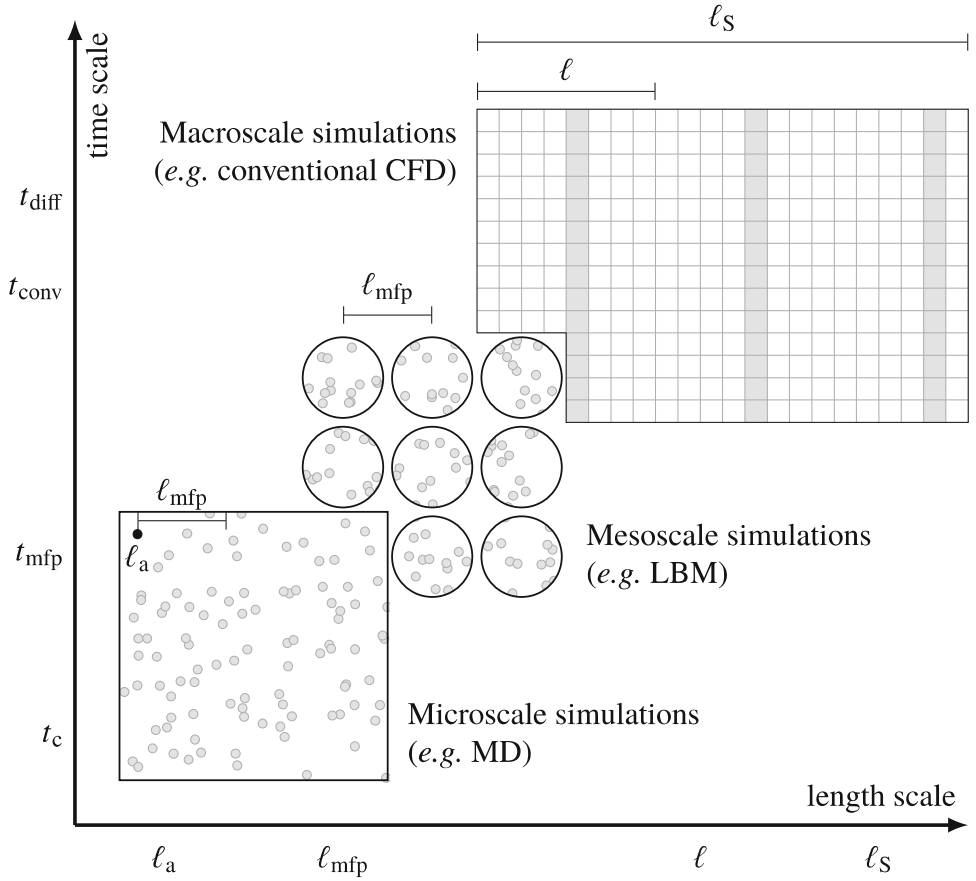
Challenges

- Efficient **encoding** of data into quantum register
- Efficient **read-out** of QoIs from quantum register
- Realization of 'fluid dynamics' as **quantum circuit**

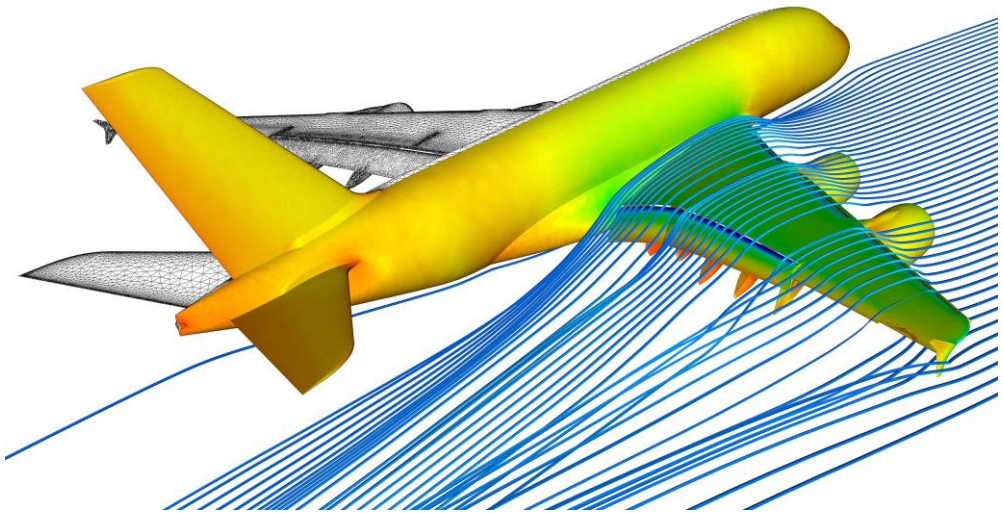
Most 'fluid dynamics' is **non-linear** and **irreversible**

Quantum computing is **linear** and **reversible**

Different fluid models

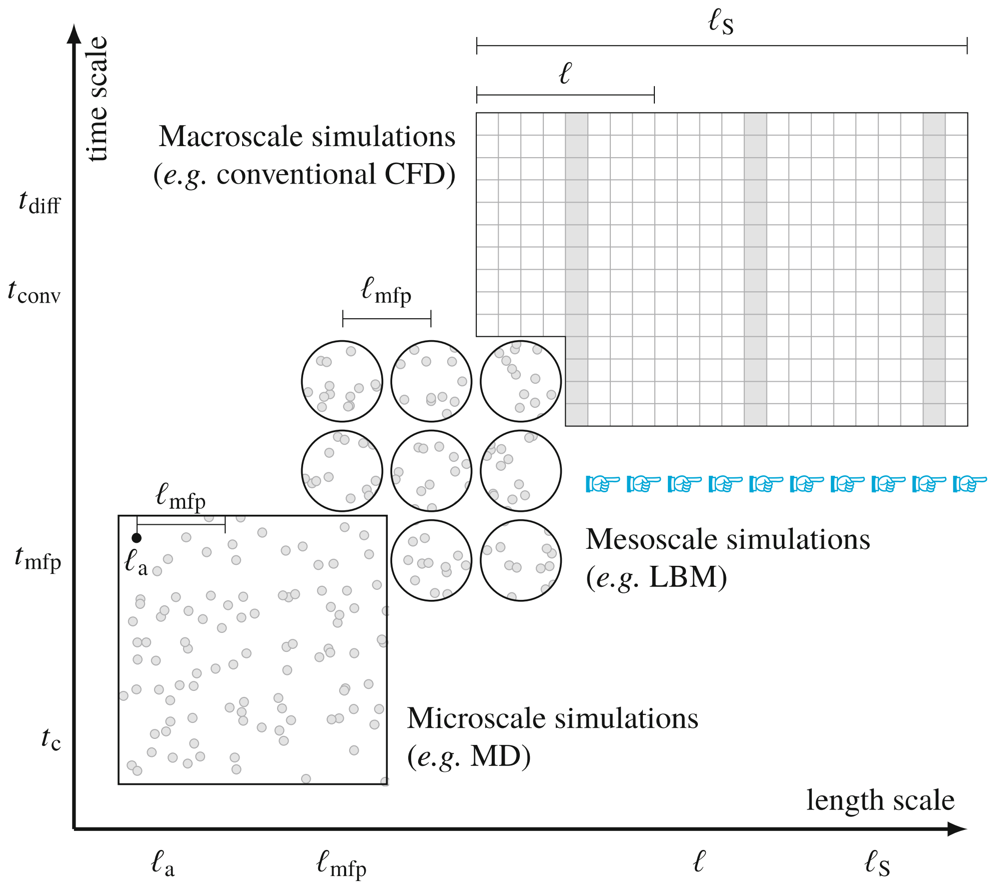


☞☞☞☞☞☞☞☞☞☞☞☞☞☞ Navier-Stokes solver



Left: DOI: 10.1007/978-3-319-44649-3, right: https://eaglepubs.erau.edu

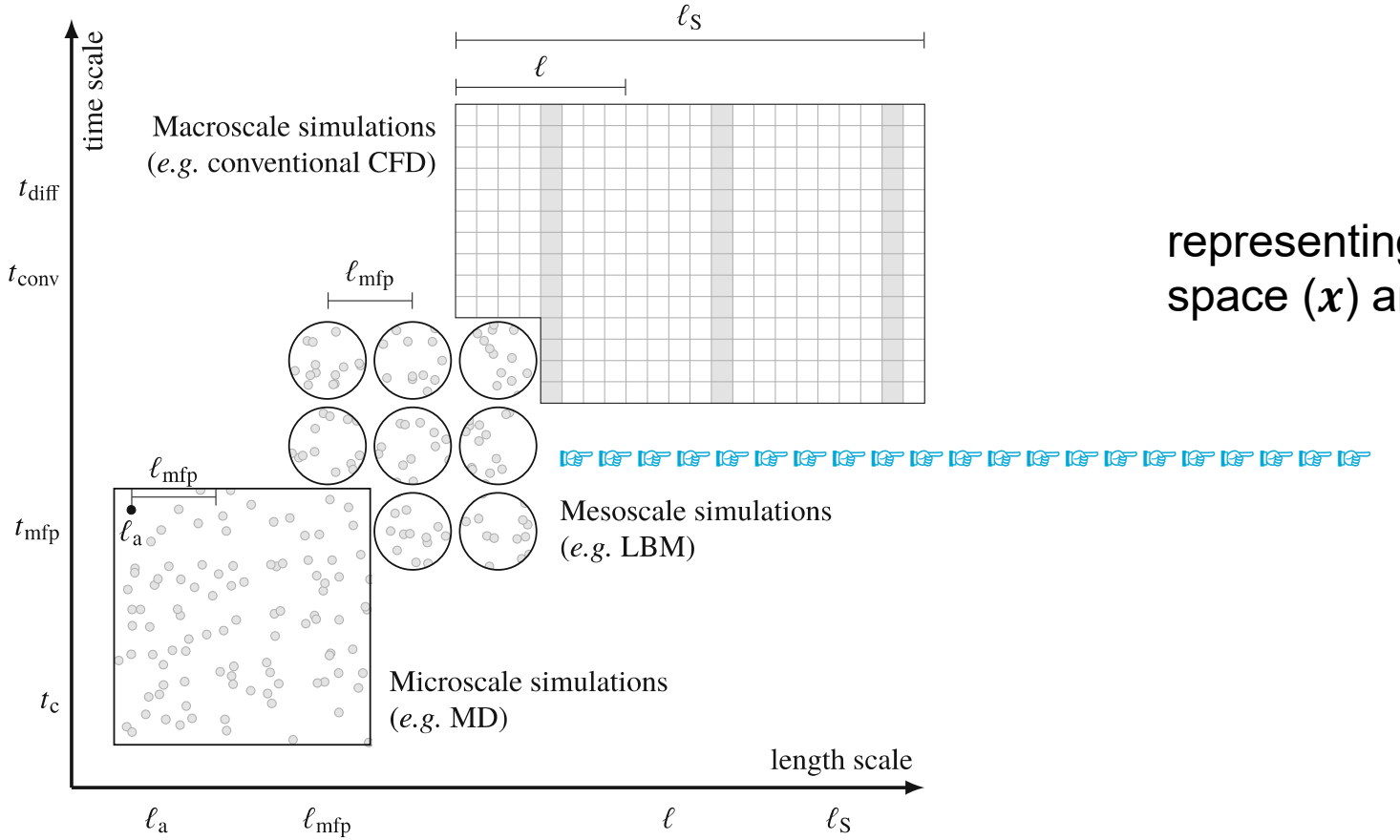
Different fluid models



Lattice Boltzmann solver

Left: DOI: 10.1007/978-3-319-44649-3, right: https://www.3ds.com/products/simulia/powerflow

Different fluid models



Kinematic theory based on distribution functions

$$f(\boldsymbol{x}, \boldsymbol{v}, t) \qquad \left[\frac{\mathrm{kg\ s^3}}{\mathrm{m^6}}\right]$$

representing the density of mass in the physical space ($\boldsymbol{x}$) and the velocity space ($\boldsymbol{v}$) at time t

Boltzmann equation

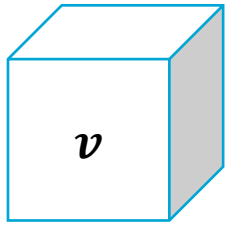
$$\frac{\partial f}{\partial t} + \boldsymbol{e} \cdot \nabla f = \Omega_{\mathrm{coll}}$$

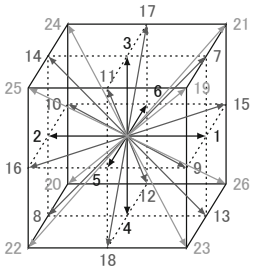
Left: DOI: 10.1007/978-3-319-44649-3

From Boltzmann equation to lattice Boltzmann method

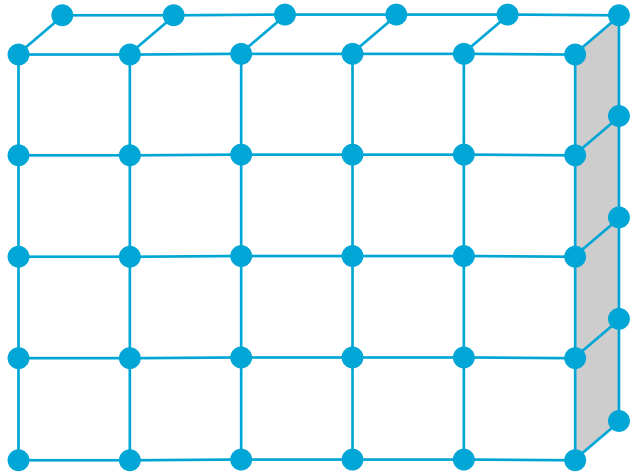
$$\frac{\partial f}{\partial t} + \boldsymbol{e} \cdot \nabla f = \Omega$$

Discretize in velocity space → Discretize in space and time





$f(\boldsymbol{x}, \boldsymbol{v}, t)$ → $f_\alpha(\boldsymbol{x}, t) = f(\boldsymbol{x}, \boldsymbol{c}_\alpha, t)$

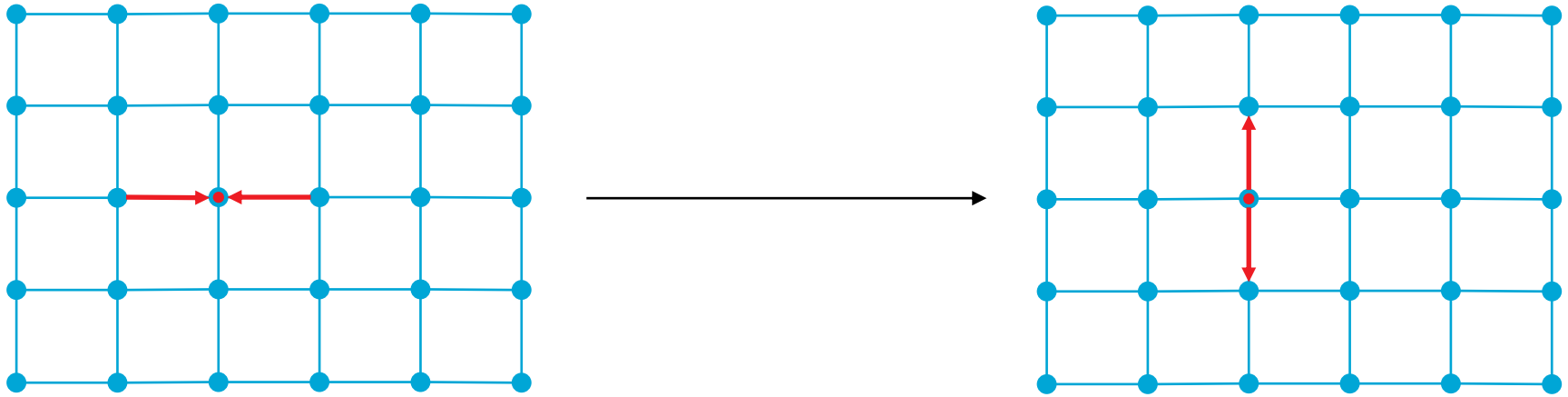


$f_\alpha(\boldsymbol{x}_i, t^n)$

The lattice Boltzmann method

$$\frac{\partial f_\alpha}{\partial t} + \boldsymbol{c_\alpha} \cdot \nabla f_\alpha = \Omega_\alpha$$

Collision of all f_α at grid point $\boldsymbol{x}_i$

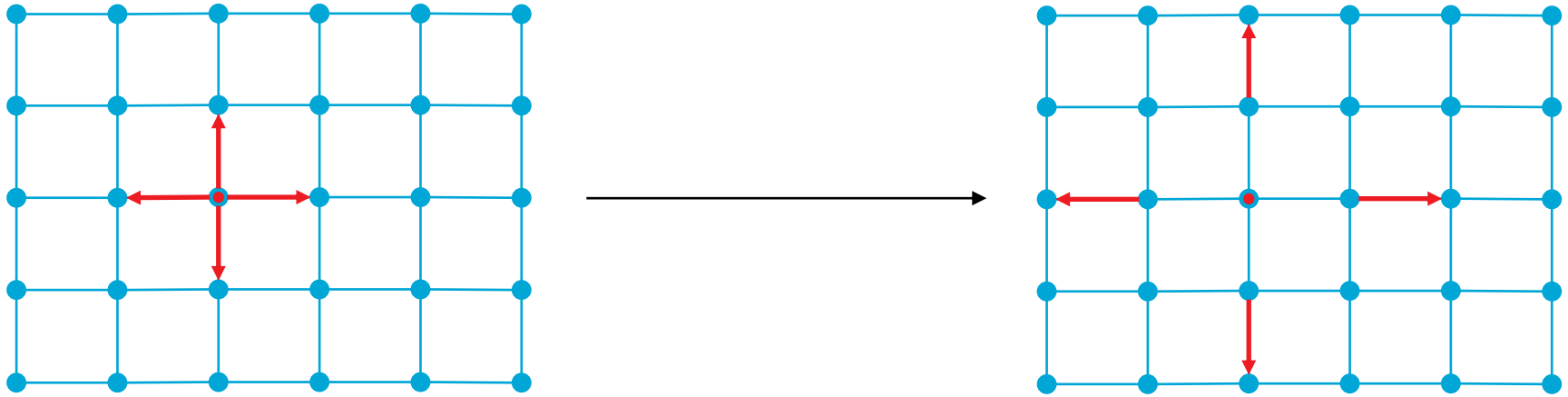


$$f_\alpha^*(\boldsymbol{x}_i, t^n) = f_\alpha(\boldsymbol{x}_i, t^n) + \Delta t \Omega_\alpha(\boldsymbol{x}_i, t^n)$$

The lattice Boltzmann method

$$\frac{\partial f_\alpha}{\partial t} + \boldsymbol{c}_\alpha \cdot \nabla f_\alpha = \Omega_\alpha$$

Streaming of all f_α along direction $\boldsymbol{c}_\alpha$

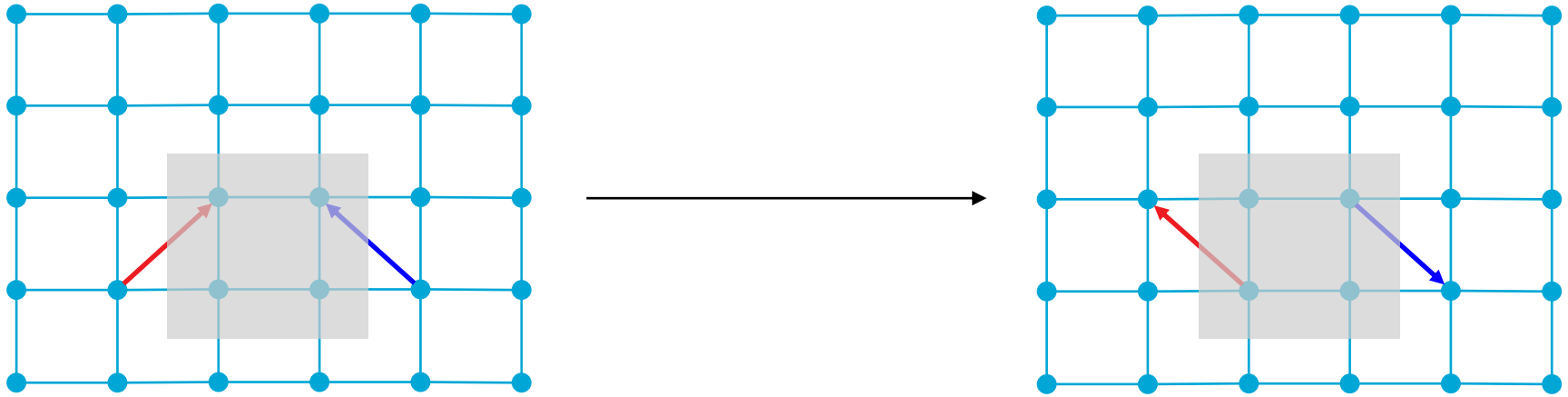


$$f_\alpha(\boldsymbol{x}_i + \boldsymbol{c}_\alpha \Delta t, t^n + \Delta t) = f_\alpha^*(\boldsymbol{x}_i, t^n)$$

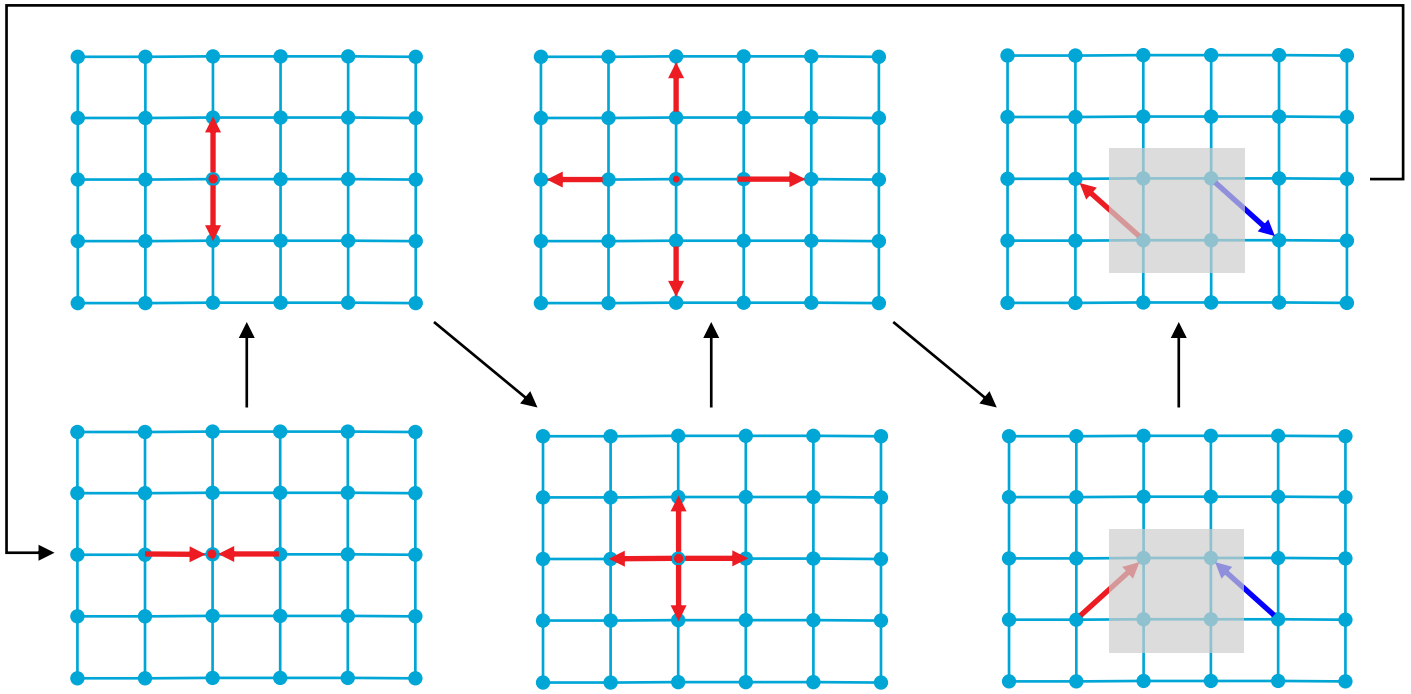
The lattice Boltzmann method

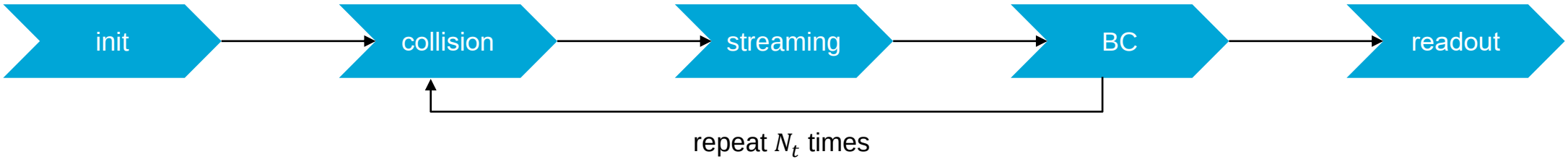
$$\frac{\partial f_\alpha}{\partial t} + \boldsymbol{c_\alpha} \cdot \nabla f_\alpha = \Omega_\alpha$$

Reflection or bounce-back boundary

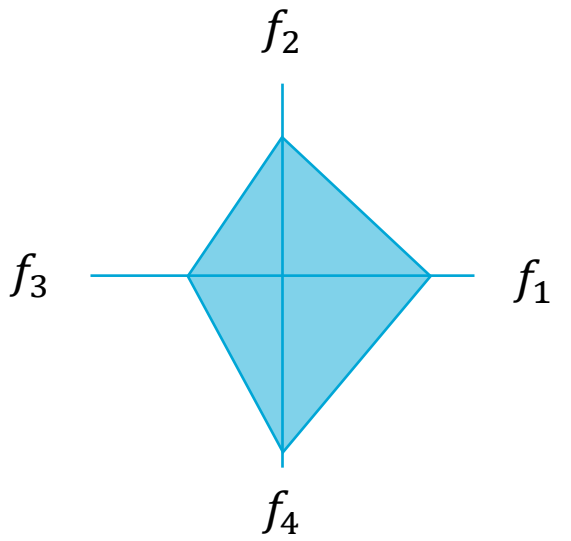


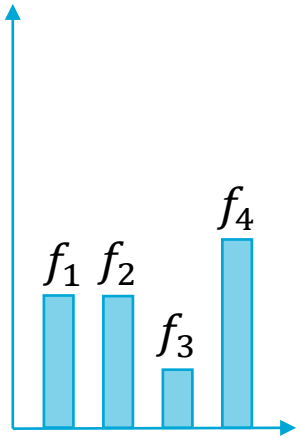
The lattice Boltzmann method





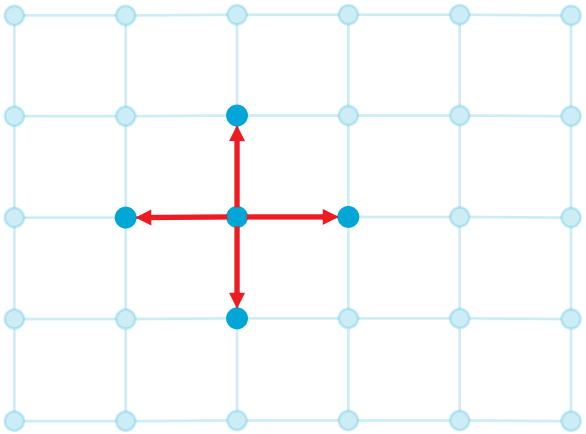
The *quantum* lattice Boltzmann method

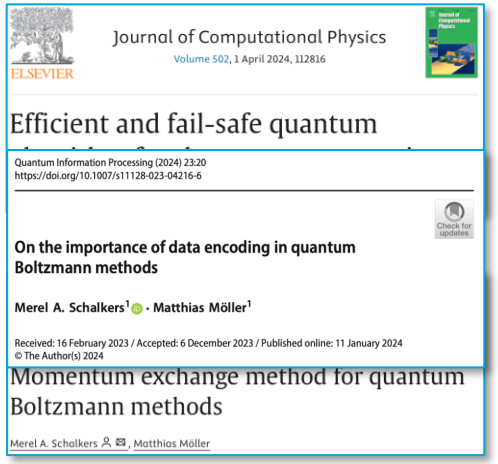




Amplitude based encoding

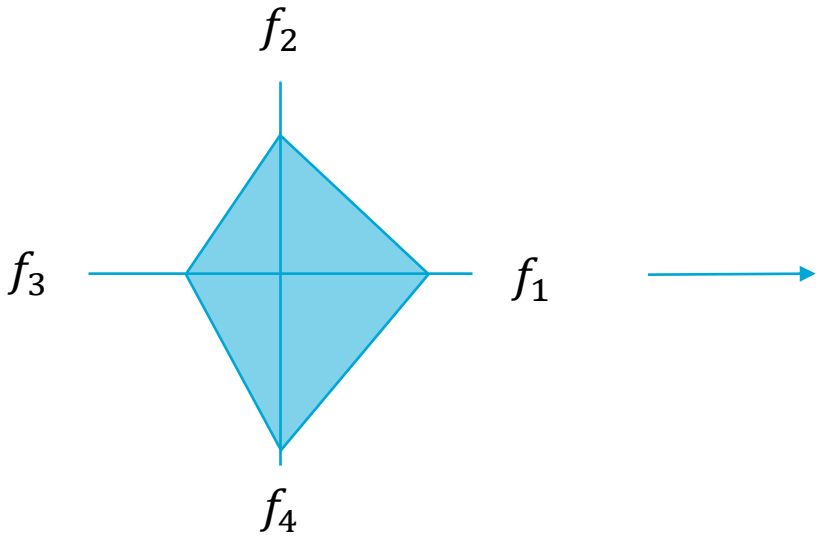
$$\sum_{\alpha} f_{\alpha}|\alpha\rangle|x\rangle|y\rangle$$

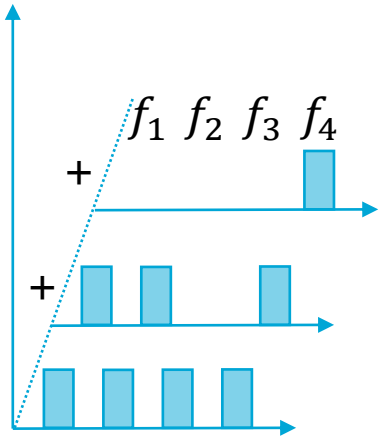



Journal of Computational Physics
Volume 502, 1 April 2024, 112816
ELSEVIER
Efficient and fail-safe quantum
Quantum Information Processing (2024) 23:20
https://doi.org/10.1007/s11128-023-04216-6
On the importance of data encoding in quantum Boltzmann methods
Merel A. Schalkers · Matthias Möller
Received: 16 February 2023 / Accepted: 6 December 2023 / Published online: 11 January 2024
© The Author(s) 2024
Momentum exchange method for quantum Boltzmann methods
Merel A. Schalkers, Matthias Möller

- ☞ Collisionless (=streaming) QLBM with correct boundary treatment
- ☞ Proved that collision cannot be implemented as unitary operator
- ☞ Quantum momentum exchange method to read-out forces

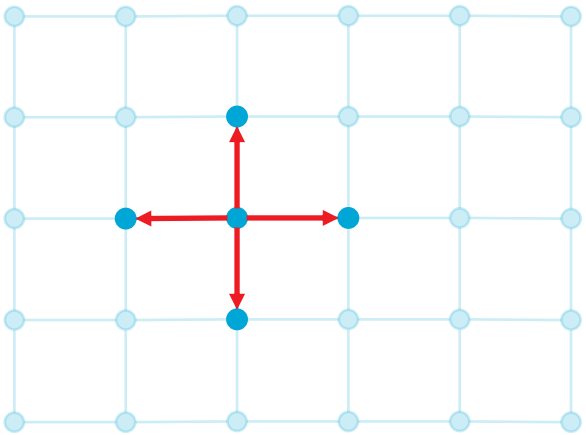
The *quantum* lattice Boltzmann method

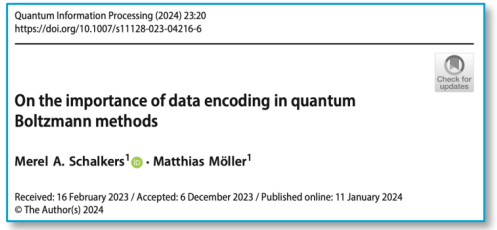




Basis state encoding

$$(|1111\rangle + |1101\rangle + |0001\rangle)|x\rangle|y\rangle$$





Quantum Information Processing (2024) 23:20
https://doi.org/10.1007/s11128-023-04216-6

On the importance of data encoding in quantum Boltzmann methods

Merel A. Schalkers[1] · Matthias Möller[1]

Received: 16 February 2023 / Accepted: 6 December 2023 / Published online: 11 January 2024
© The Author(s) 2024

- ☞ Proved that streaming cannot be implemented as unitary operator

- ☞ Proposed novel space-time encoding ("extended basis state encoding") that enables the implementation of streaming and collision as efficient quantum circuits

Space-time QLBM

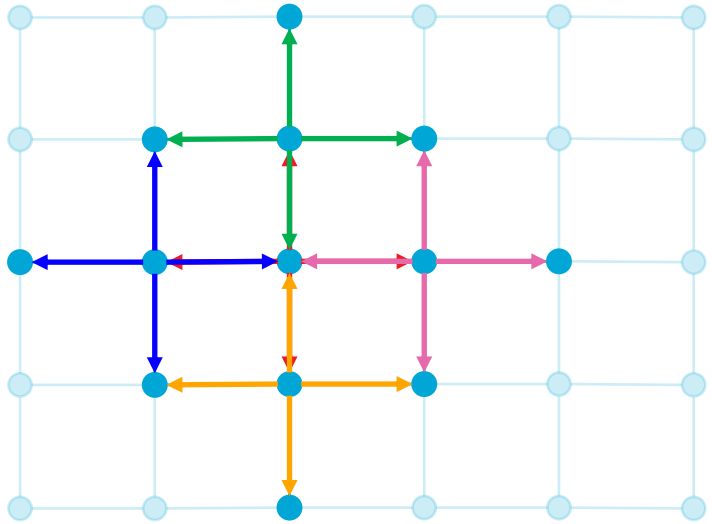
Quantum Information Processing (2024) 23:20
https://doi.org/10.1007/s11128-023-04216-6

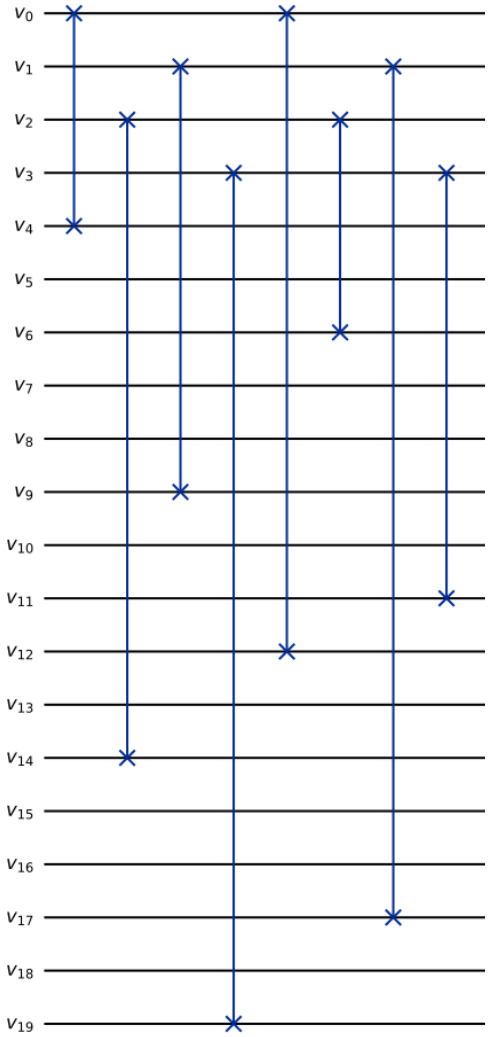
Check for updates

On the importance of data encoding in quantum Boltzmann methods

Merel A. Schalkers[1] · Matthias Möller[1]

Received: 16 February 2023 / Accepted: 6 December 2023 / Published online: 11 January 2024





streaming

Space-time QLBM

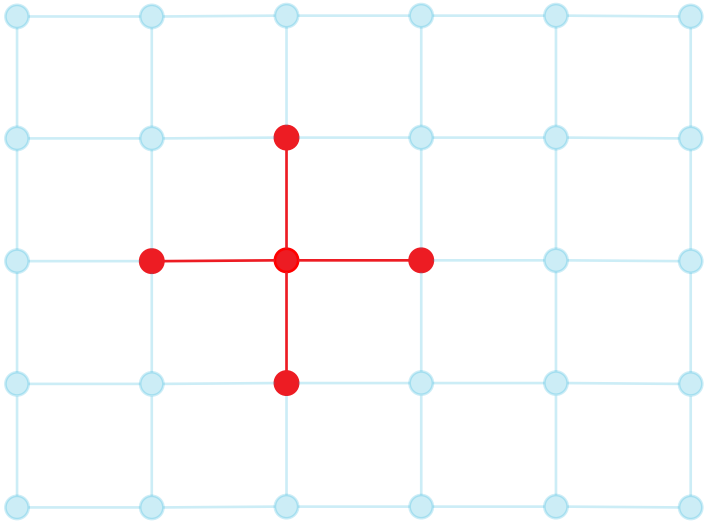
Quantum Information Processing (2024) 23:20
https://doi.org/10.1007/s11128-023-04216-6

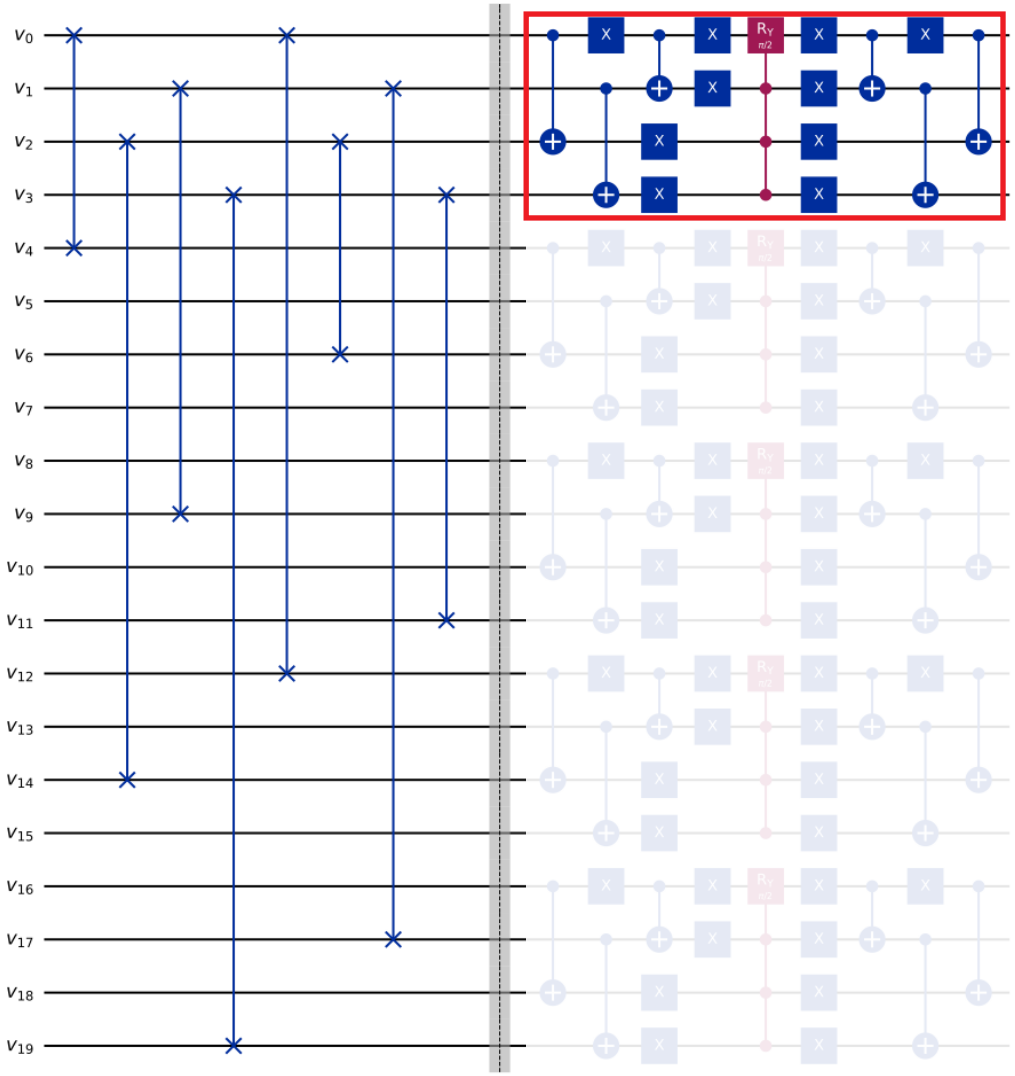
Check for updates

On the importance of data encoding in quantum Boltzmann methods

Merel A. Schalkers[1] · Matthias Möller[1]

Received: 16 February 2023 / Accepted: 6 December 2023 / Published online: 11 January 2024





streaming collision

Space-time QLBM

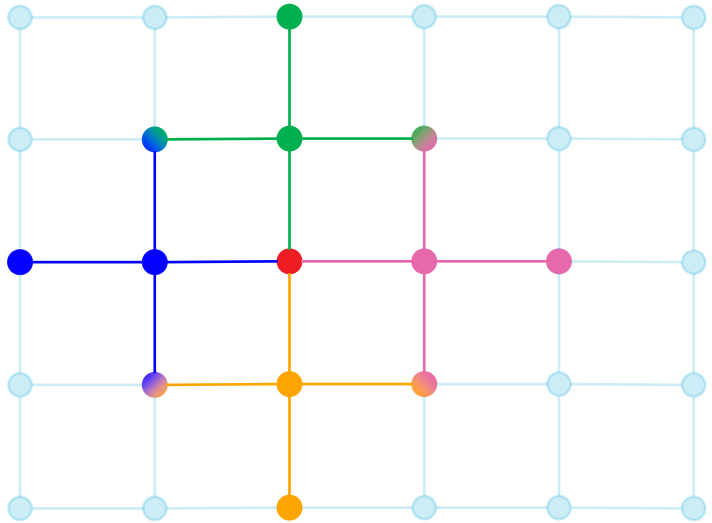
Quantum Information Processing (2024) 23:20
https://doi.org/10.1007/s11128-023-04216-6

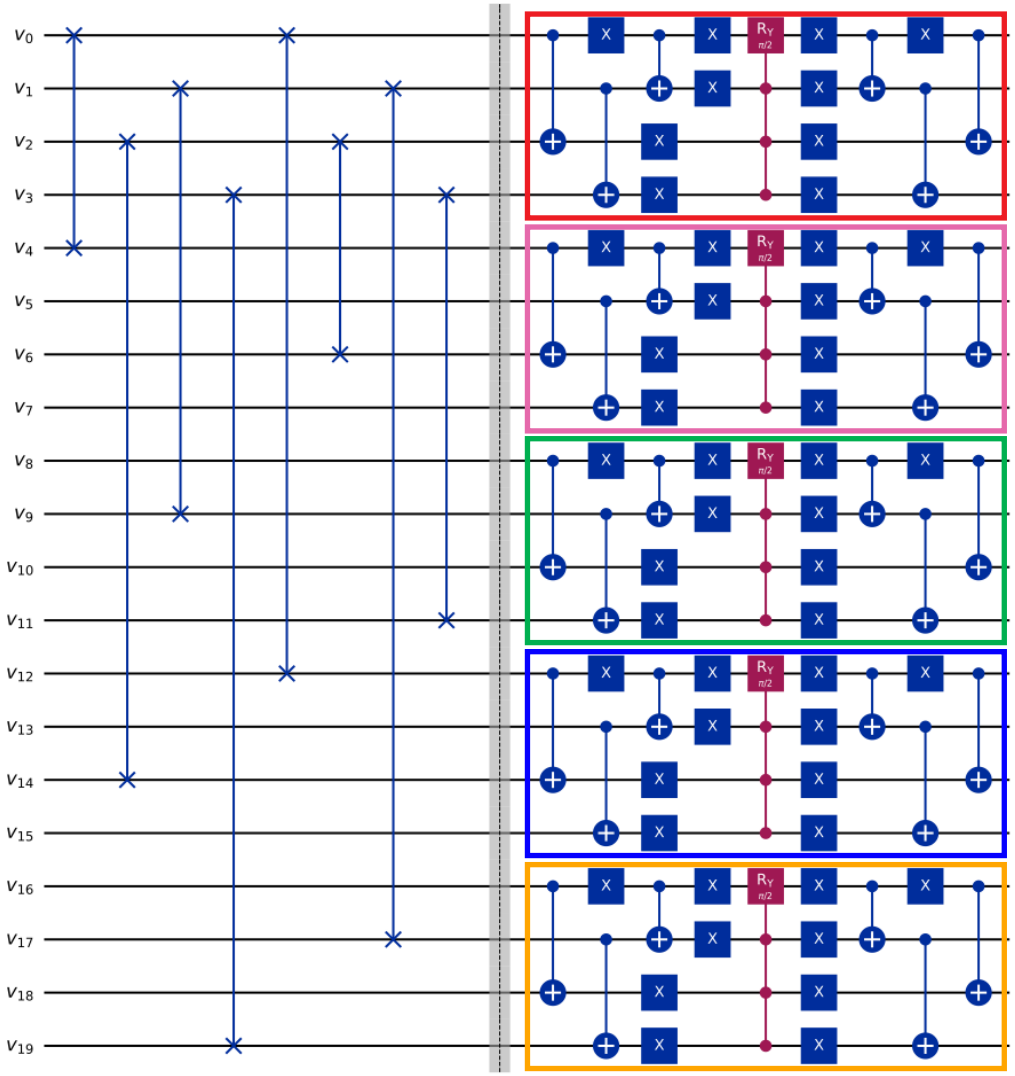
Check for updates

On the importance of data encoding in quantum Boltzmann methods

Merel A. Schalkers[1] · Matthias Möller[1]

Received: 16 February 2023 / Accepted: 6 December 2023 / Published online: 11 January 2024





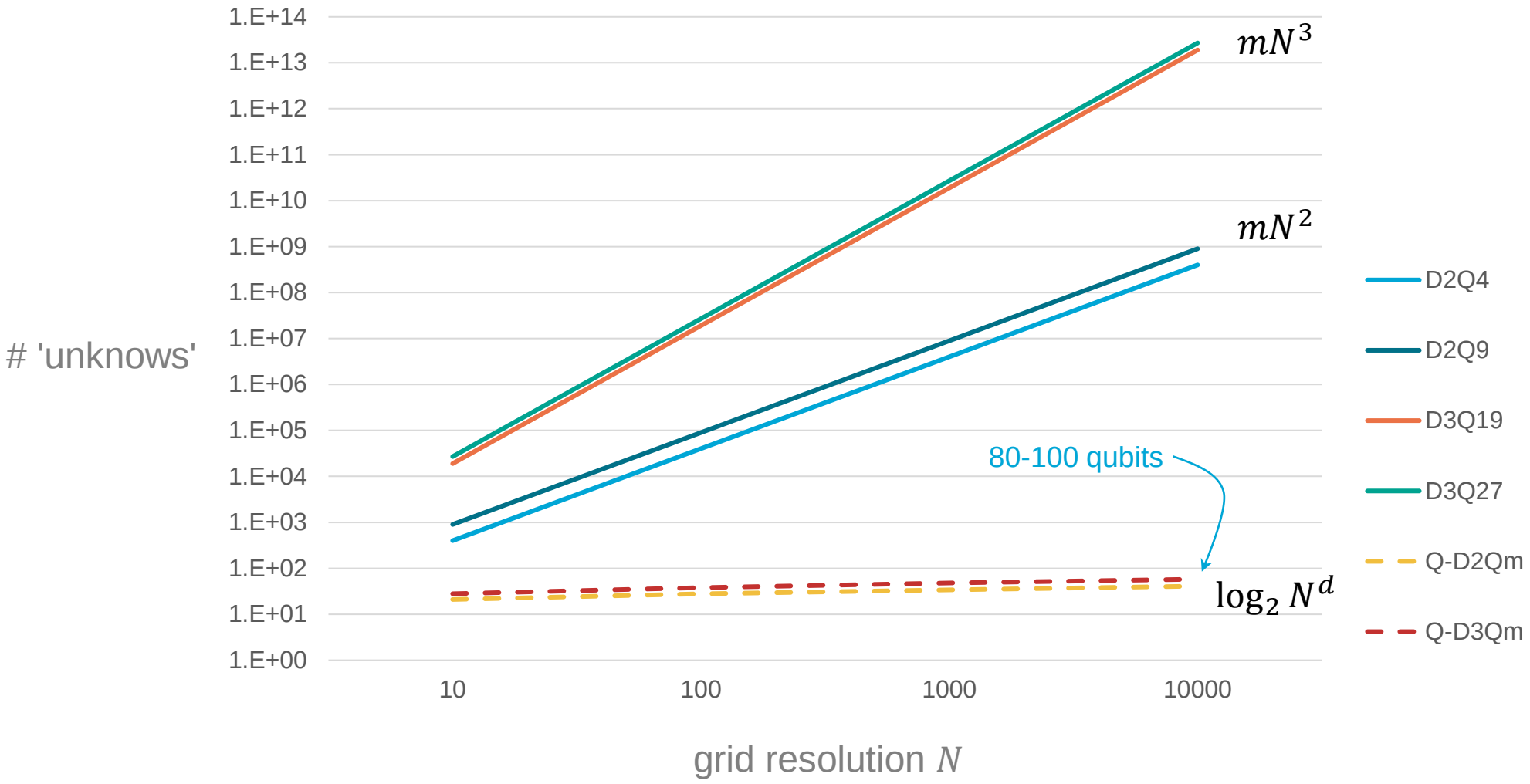
Our QLBM variants

Complexity analysis for D_2Q_4

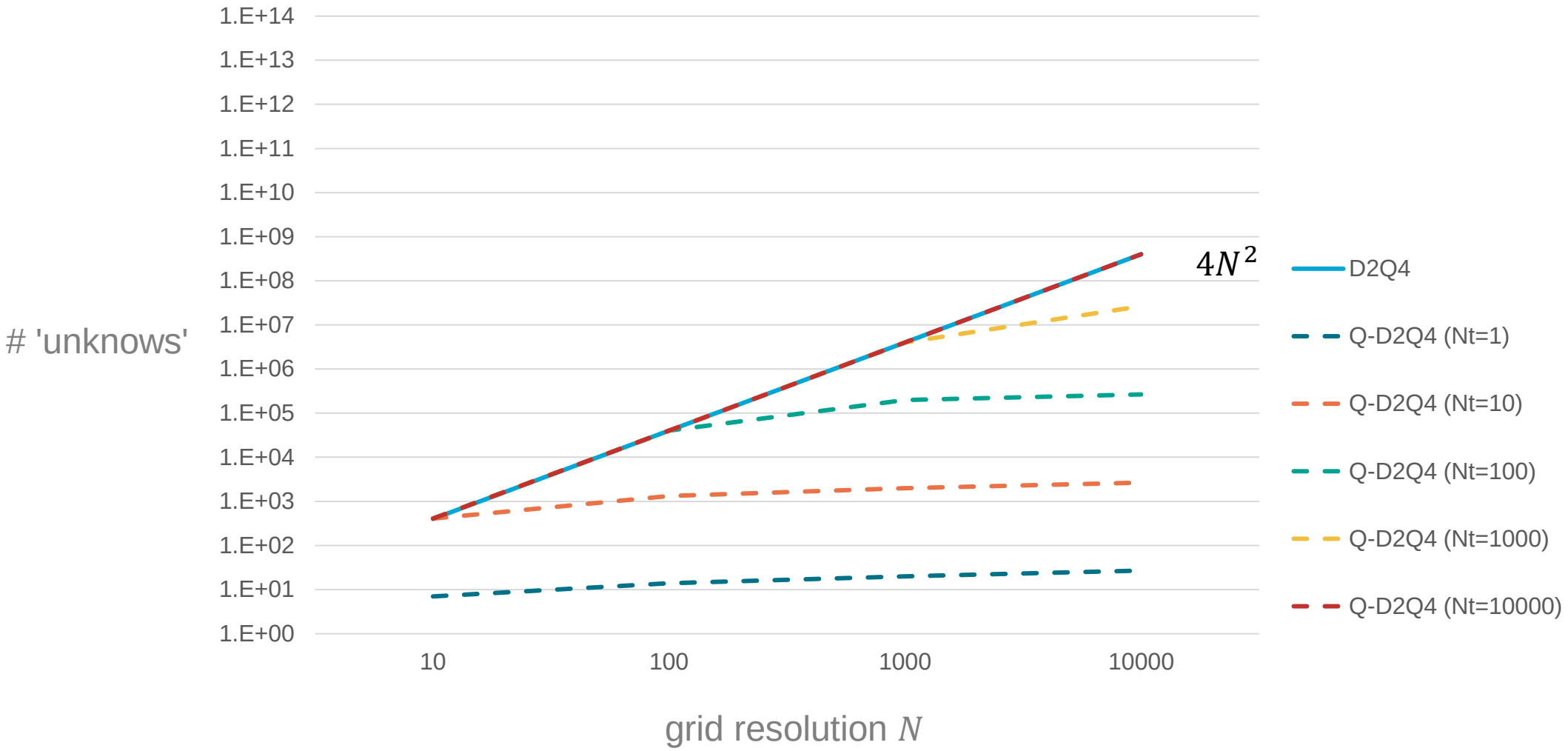
	Amplitude-based encoding	Space-time encoding
#qubits	$O(\log_2 N_g)$	$O(N_t^2 + \log_2 N_g)$ *)
qops streaming	$O\left(N_t(\log_2 N_g)^2\right)$	$\sum_{t=0}^{N_t} \lceil \log_2(N_t - t) \rceil$
qops collision	n/a	$20N_t$ **)

*) upper bound $4N_g + \log_2 N_g$ **) factor depends on the concrete implementation

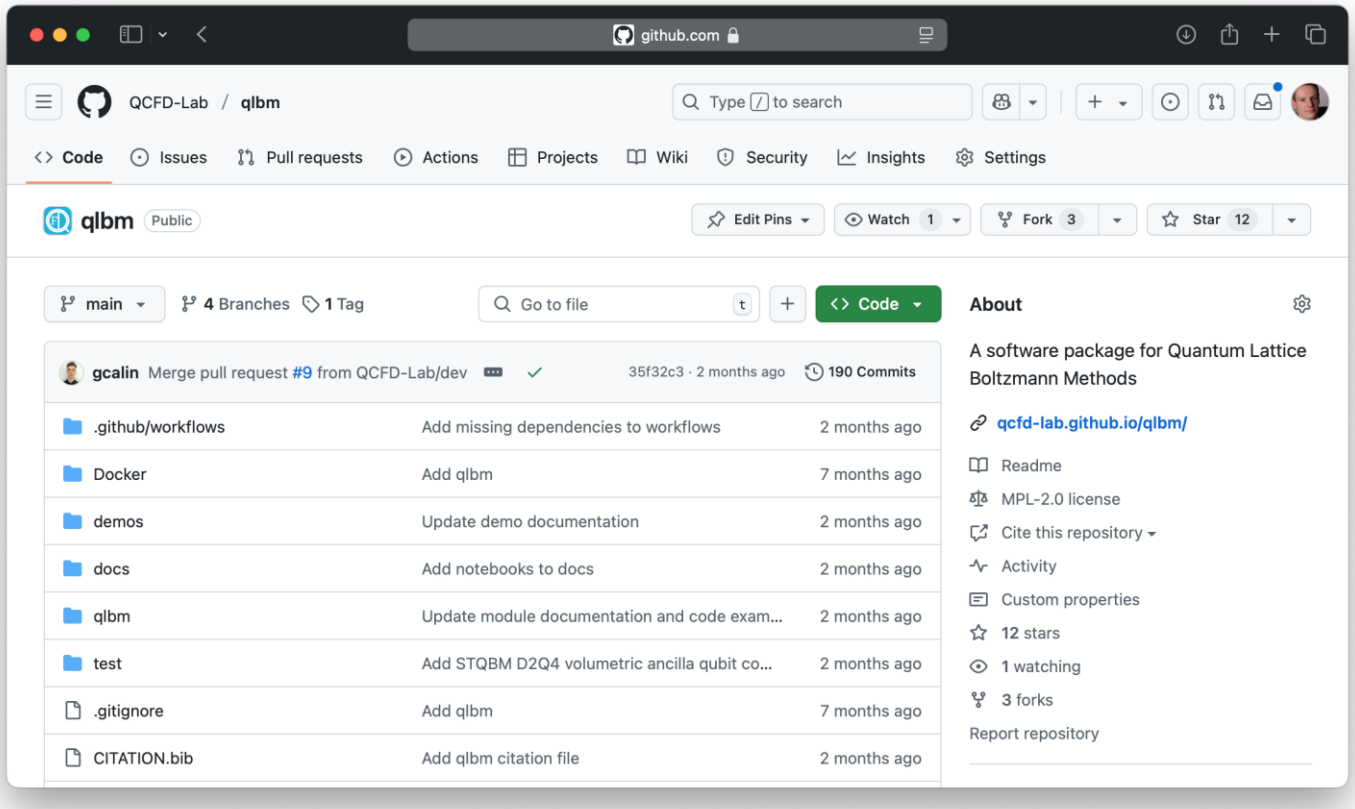
Memory footprint: amplitude-based encoding



Memory footprint: space-time encoding



The QLBM software package



Building on top of Qiskit, Tket and Qulacs simulator

pip install qlbm

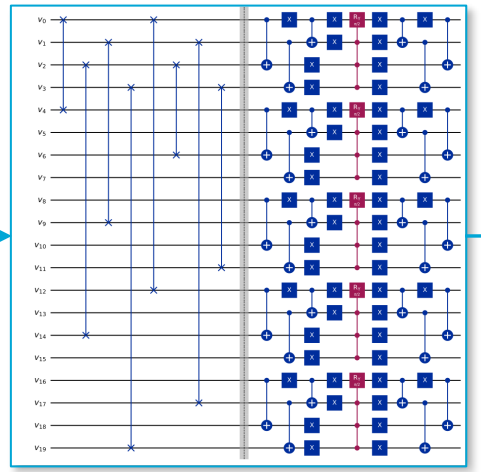
QLBM

The QLBM software package

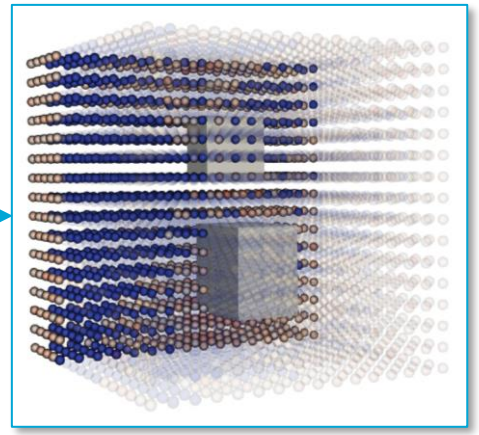
QLBM

```
{ "lattice":
    { "dim":
       { "x": 16,
         "y": 16 },
      "velocities":
       {  "x": 4,
          "y": 4 }
   },
  "geometry": [
    { "x": [9, 12],
      "y": [9, 12],
      "boundary": "bounceback" },
    { "x": [9, 12],
      "y": [3, 6],
      "boundary": "specular" }
   ]
}
```

High-level specification



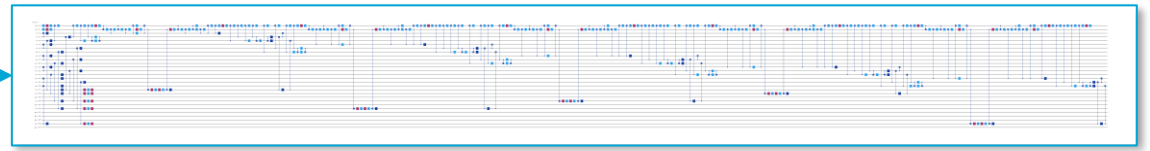
High-level circuits



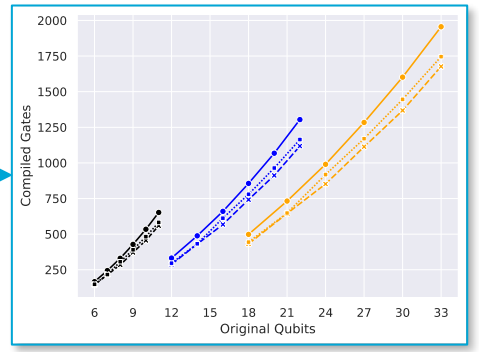
Simulation

TKET™

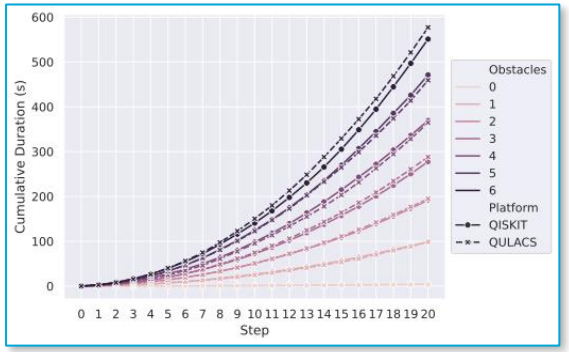
ParaView



Hardware optimized executable circuit



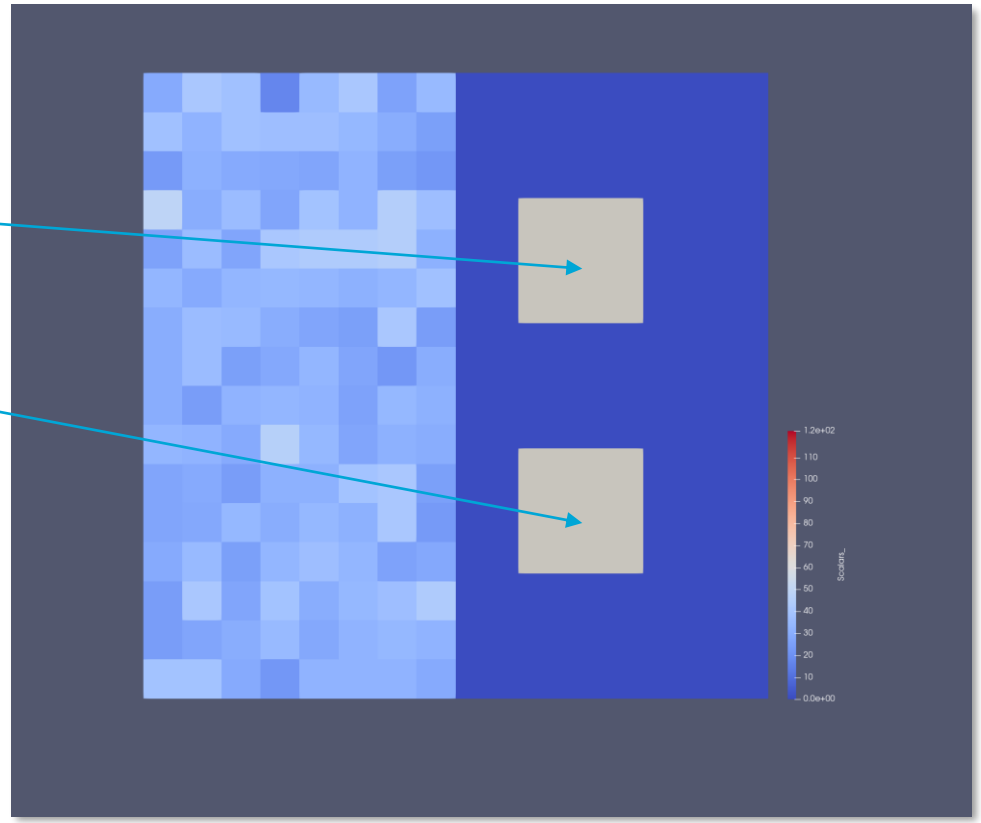
Resource estimation



Performance benchmarking

The QLBM software package

```
lattice = CollisionlessLattice(
   { "lattice":
        { "dim": {"x": 16, "y": 16},
          "velocities": {"x": 4, "y": 4}
        },
     "geometry": [
        { "x": [9, 12],
          "y": [9, 12],
          "boundary": "bounceback" },
        { "x": [9, 12],
          "y": [3, 6],
          "boundary": "specular" },
      ]})
```



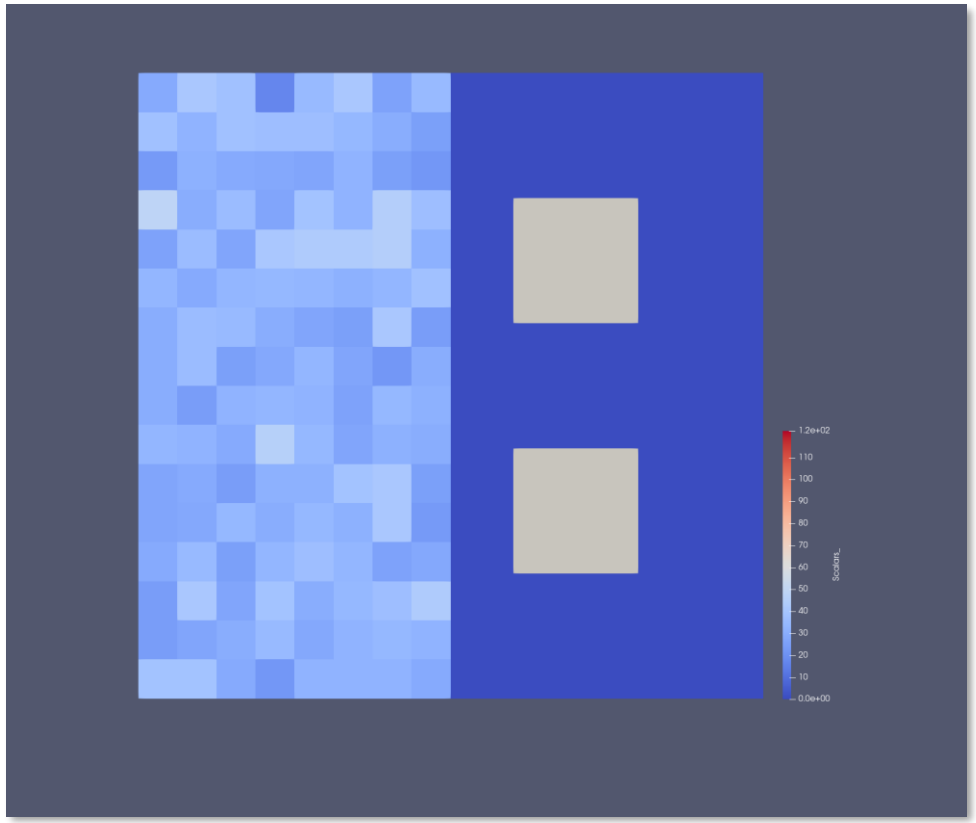
The QLBM software package

```
lattice = CollisionlessLattice(
   { "lattice":
       { "dim": {"x": 16, "y": 16},
         "velocities": {"x": 4, "y": 4}
       },
     "geometry": [
       { "x": [9, 12],
         "y": [9, 12],
         "boundary": "bounceback" },
       { "x": [9, 12],
         "y": [3, 6],
         "boundary": "specular" },
      ]})

cfg = SimulationConfig(…)
cfg.prepare_for_simulation()



runner = QiskitRunner(cfg, lattice)
runner.run(NUM_STEPS, NUM_SHOTS)
```



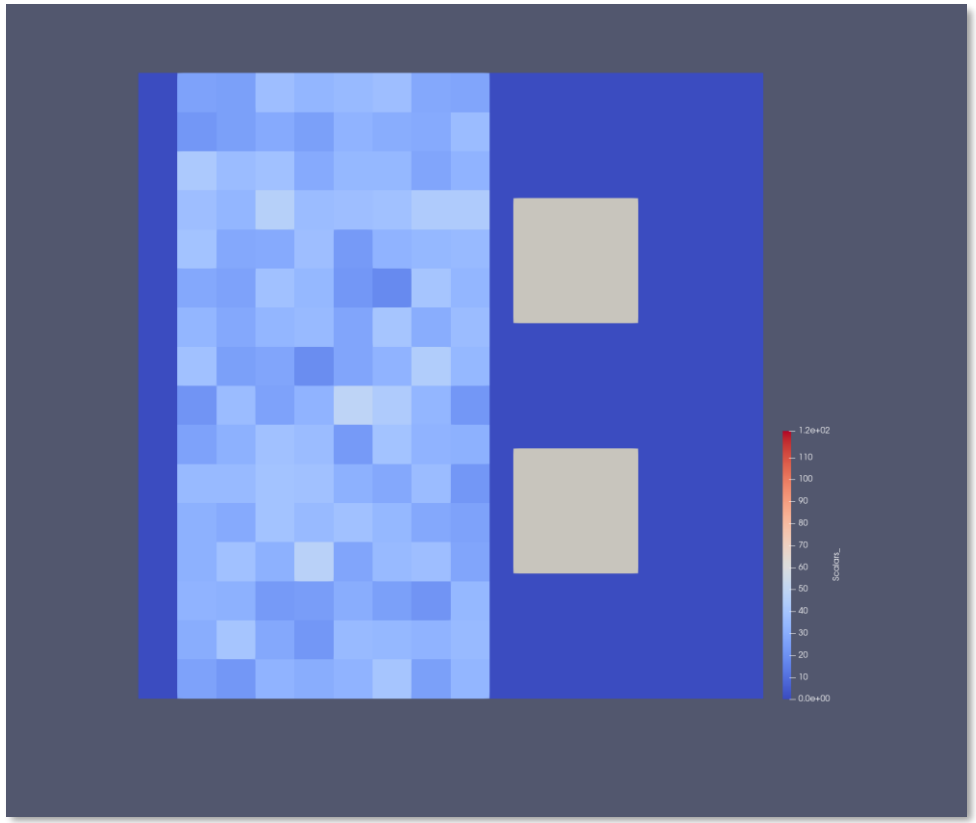
The QLBM software package

```
lattice = CollisionlessLattice(
   { "lattice":
       { "dim": {"x": 16, "y": 16},
         "velocities": {"x": 4, "y": 4}
       },
     "geometry": [
       { "x": [9, 12],
         "y": [9, 12],
         "boundary": "bounceback" },
       { "x": [9, 12],
         "y": [3, 6],
         "boundary": "specular" },
      ]})

cfg = SimulationConfig(…)
cfg.prepare_for_simulation()



runner = QiskitRunner(cfg, lattice)
runner.run(NUM_STEPS, NUM_SHOTS)
```



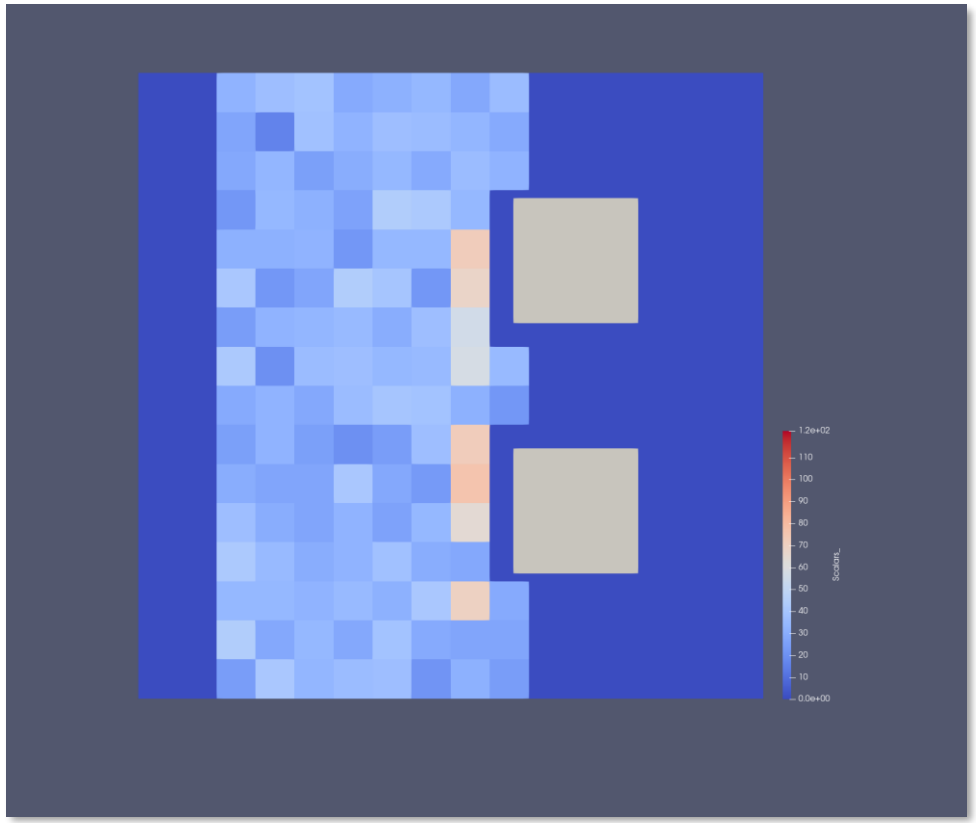
The QLBM software package

```
lattice = CollisionlessLattice(
   { "lattice":
       { "dim": {"x": 16, "y": 16},
         "velocities": {"x": 4, "y": 4}
       },
     "geometry": [
       { "x": [9, 12],
         "y": [9, 12],
         "boundary": "bounceback" },
       { "x": [9, 12],
         "y": [3, 6],
         "boundary": "specular" },
      ]})

cfg = SimulationConfig(…)
cfg.prepare_for_simulation()



runner = QiskitRunner(cfg, lattice)
runner.run(NUM_STEPS, NUM_SHOTS)
```



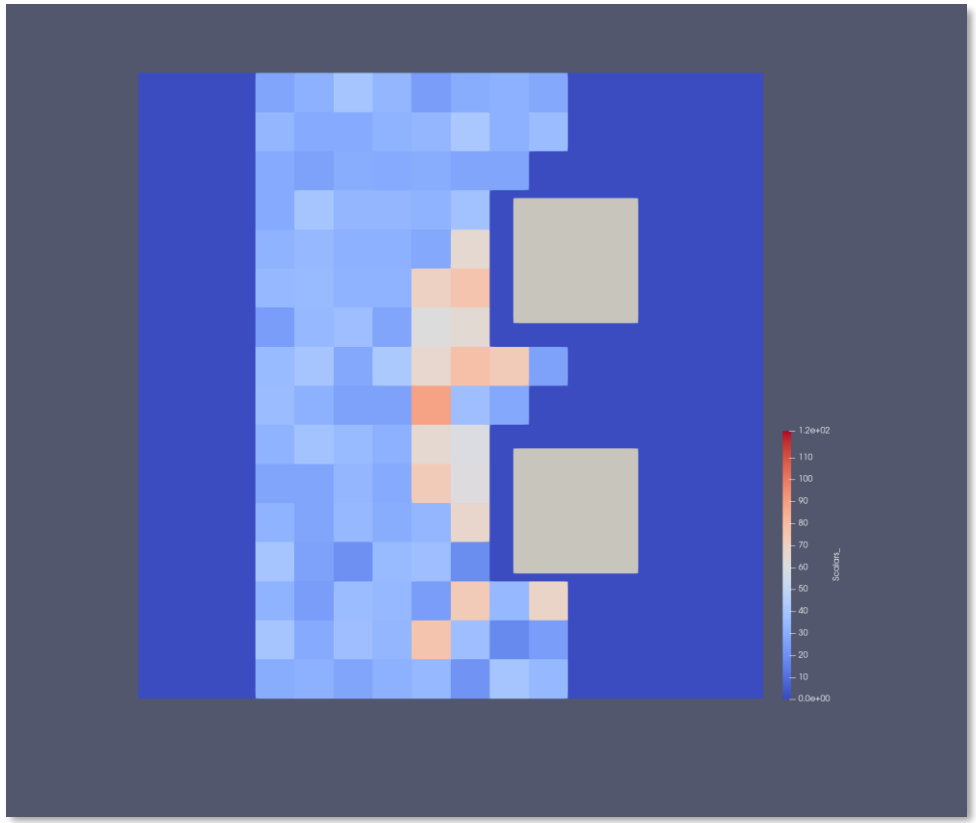
The QLBM software package

```
lattice = CollisionlessLattice(
   { "lattice":
       { "dim": {"x": 16, "y": 16},
         "velocities": {"x": 4, "y": 4}
       },
     "geometry": [
       { "x": [9, 12],
         "y": [9, 12],
         "boundary": "bounceback" },
       { "x": [9, 12],
         "y": [3, 6],
         "boundary": "specular" },
      ]})

cfg = SimulationConfig(…)
cfg.prepare_for_simulation()



runner = QiskitRunner(cfg, lattice)
runner.run(NUM_STEPS, NUM_SHOTS)
```



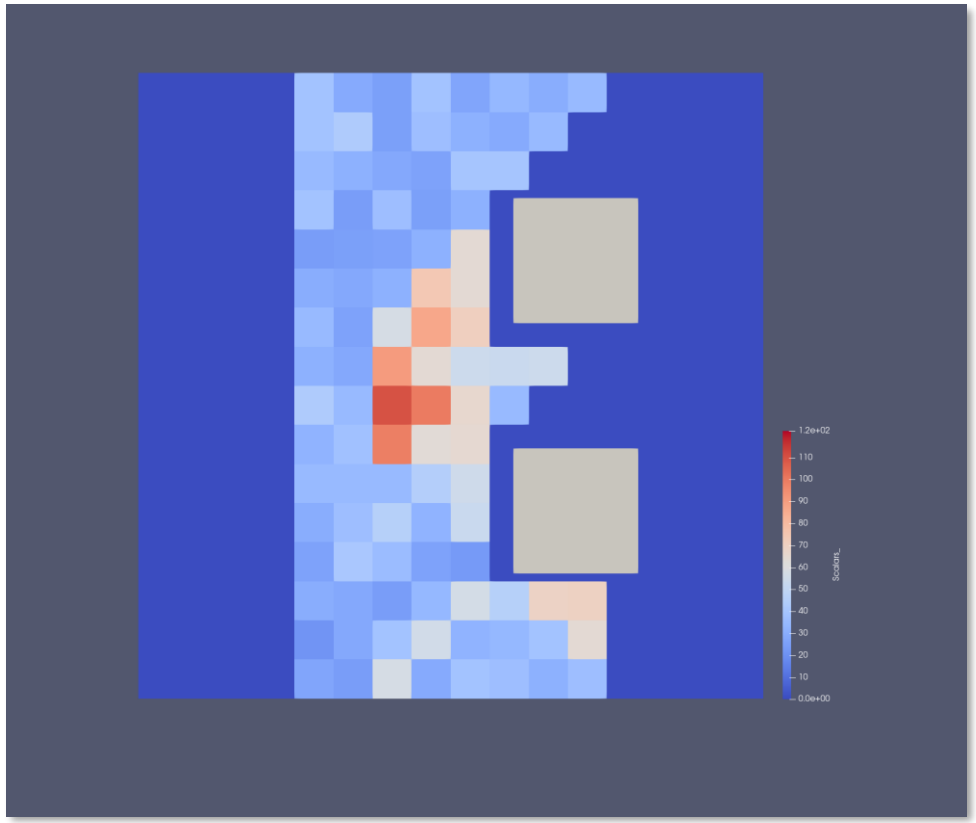
The QLBM software package

```
lattice = CollisionlessLattice(
   { "lattice":
       { "dim": {"x": 16, "y": 16},
         "velocities": {"x": 4, "y": 4}
       },
     "geometry": [
       { "x": [9, 12],
         "y": [9, 12],
         "boundary": "bounceback" },
       { "x": [9, 12],
         "y": [3, 6],
         "boundary": "specular" },
      ]})

cfg = SimulationConfig(…)
cfg.prepare_for_simulation()



runner = QiskitRunner(cfg, lattice)
runner.run(NUM_STEPS, NUM_SHOTS)
```



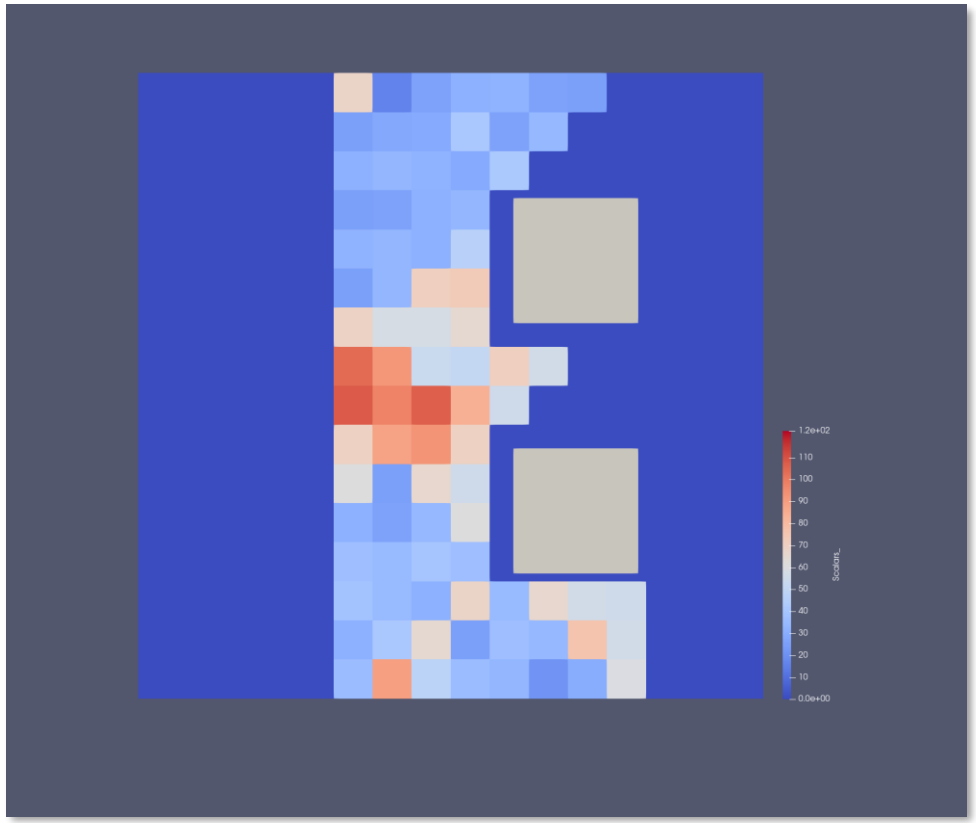
The QLBM software package

```
lattice = CollisionlessLattice(
   { "lattice":
       { "dim": {"x": 16, "y": 16},
         "velocities": {"x": 4, "y": 4}
       },
     "geometry": [
       { "x": [9, 12],
         "y": [9, 12],
         "boundary": "bounceback" },
       { "x": [9, 12],
         "y": [3, 6],
         "boundary": "specular" },
      ]})

cfg = SimulationConfig(…)
cfg.prepare_for_simulation()



runner = QiskitRunner(cfg, lattice)
runner.run(NUM_STEPS, NUM_SHOTS)
```



The QLBM software package

SpaceTimeLattice

```
lattice = CollisionlessLattice(
   { "lattice":
       { "dim": {"x": 16, "y": 16},
         "velocities": {"x": 4, "y": 4}
       },
     "geometry": [
       { "x": [9, 12],
         "y": [9, 12],
         "boundary": "bounceback" },
       { "x": [9, 12],
         "y": [3, 6],
         "boundary": "specular" },
      ]})

cfg = SimulationConfig(…)
cfg.prepare_for_simulation()
```

QulacsRunner

```
runner = QiskitRunner(cfg, lattice)
runner.run(NUM_STEPS, NUM_SHOTS)
```

QLBM – A Quantum Lattice Boltzmann Software Framework

A Preprint

Călin A. Georgescu
Delft University of Technology
Mekelweg 4, 2628CD, Delft
c.a.georgescu@tudelft.nl

Merel A. Schalkers
Delft University of Technology
Mekelweg 4, 2628CD, Delft
m.a.schalkers@tudelft.nl

Matthias Möller
Delft University of Technology
Mekelweg 4, 2628CD, Delft
m.moller@tudelft.nl

December 2, 2024

Abstract

We present QLBM, a Python software package designed to facilitate the development, simulation, and analysis of Quantum Lattice Boltzmann Methods (QBMs). QLBM is a modular framework that introduces a quantum component abstraction hierarchy tailored to the implementation of novel QBMs. The framework interfaces with state-of-the-art quantum software infrastructure to enable efficient simulation and validation pipelines, and leverages novel execution and pre-processing techniques that significantly reduce the computational resources required to develop quantum circuits. We demonstrate the versatility of the software by showcasing multiple QBMs in 2D and 3D with complex boundary conditions, integrated within automated benchmarking utilities. Accompanying the source code are extensive test suites, thorough online documentation resources, analysis tools, visualization methods, and demos that aim to increase the accessibility of QBMs while encouraging reproducibility and collaboration. The source code of QLBM is publicly available under a permissive MPL 2.0 license at https://github.com/QCFD-Lab/qlbm.

Keywords Quantum computing · Lattice Boltzmann method · Quantum software · Computational fluid dynamics

1 Introduction

The field of Quantum Computing (QC) [50] has received a staggering amount of attention in recent decades from researchers and practitioners alike. Ever since the formulation of the first quantum algorithms in the early 1990s, quantum computing captured the interest and attention of scientists attempting to accelerate solvers for high impact, real-life problems. It was algorithms like those of Deutsch and Jozsa [23], Bernstein and Vazirani [9], Grover [30], and Shor [69] that initiated a wave of research aiming to understand how QC can revolutionize the status quo. Two properties make the quantum computing paradigm especially attractive for the increasingly demanding large-scale computational demands of today – exponential information compression and quantum parallelism. The former is a core property of the basic unit of quantum information: the quantum bit or qubit. Unlike classical bits, n qubits encode a superposition that can be represented through a 2^n-dimensional vector belonging to a complex Hilbert space. The latter, quantum parallelism, refers to the ability of quantum computers to simultaneously encode and update multiple results in a single computational step. Thanks to these two properties, QC carries the potential to augment the current computational landscape with a drastically different yet complementary archetype.

The drive to accelerate classical solvers by means of quantum computing has led to novel quantum algorithms that target nearly every branch of computational science. From quantum chemistry [53, 34, 16] to deep learning [67, 38], data mining [59, 46], and finance [52], quantum algorithms promise to augment or improve upon classical methods. One field where quantum computing advantages are particularly appealing is that of computational fluid dynamics (CFD). State-of-the-art CFD simulations are extremely memory- and compute-intensive applications that require tremendous amounts of resources to tackle modern engineering tasks. It is this computational capacity bottleneck, together with

Conclusions

- QLBM end-to-end implementation: *initialization + streaming + collision + BC + QoI read-out*
- Open-source QLBM software framework v0.0.5 released earlier this week

Thank you!　ありがとう

1st International Conference on
Applied Quantum Methods in Computational Science and Engineering

October 8-10, 2025, Aachen

ECCOMAS Thematic Conference