Physical and Mathematical Model of the Interactive LBM Airfoil Flow Simulator

Hamza Boudhair

1 Physical and Mathematical Model

The interactive flow simulator implemented in a single HTML/JavaScript file is based on a weakly compressible formulation of the Navier–Stokes equations, solved using a two-dimensional nine-velocity (D2Q9) lattice Boltzmann method (LBM). This section details the underlying governing equations, the numerical formulation, and the rationale for choosing this approach for a browser-based CFD demonstrator.

1.1 Continuum-level governing equations

At the continuum scale, the dynamics of an incompressible Newtonian fluid are governed by the mass and momentum conservation equations,

$$\nabla \cdot \mathbf{u} = 0, \tag{1}$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\frac{1}{\rho_0}\nabla p + \nu \nabla^2 \mathbf{u}, \tag{2}$$

where $\mathbf{u} = (u, v)$ is the velocity field, p is the hydrodynamic pressure, ρ_0 is a reference density, and ν is the kinematic viscosity. The present simulator does not integrate Eqs. (1)–(2) directly; instead, it uses a kinetic lattice Boltzmann formulation that recovers these equations in the low-Mach-number limit.

1.2 Lattice Boltzmann D2Q9 formulation

1.2.1 Discrete velocity set and distribution functions

In the D2Q9 model, the fluid state at each lattice node $\mathbf{x}$ and time t is described by a set of distribution functions $f_k(\mathbf{x}, t)$, $k = 0, \ldots, 8$, associated with a fixed set of discrete velocities $\mathbf{c}_k$:

$$\mathbf{c}_k \in \{(0,0), (\pm 1, 0), (0, \pm 1), (\pm 1, \pm 1)\}\,. \tag{3}$$

The time evolution of the distributions is governed by the lattice Boltzmann–BGK equation,

$$f_k(\mathbf{x} + \mathbf{c}_k\,\delta t,\, t + \delta t) - f_k(\mathbf{x}, t) = -\frac{1}{\tau}\left[f_k(\mathbf{x}, t) - f_k^{\mathrm{eq}}(\mathbf{x}, t)\right], \tag{4}$$

where τ is the non-dimensional relaxation time and f_k^{eq} is the local equilibrium distribution. The JavaScript implementation uses a single-relaxation-time (BGK) collision operator, which provides a compact and efficient update rule suitable for real-time visualization.

1.2.2 Equilibrium distribution and macroscopic variables

The equilibrium distribution f_k^{eq} is taken in its standard second-order low-Mach-number form,

$$f_k^{\mathrm{eq}}(\rho, \mathbf{u}) = w_k \, \rho \left[1 + \frac{\mathbf{c}_k \cdot \mathbf{u}}{c_s^2} + \frac{(\mathbf{c}_k \cdot \mathbf{u})^2}{2 c_s^4} - \frac{\mathbf{u}^2}{2 c_s^2} \right], \tag{5}$$

where ρ is the fluid density, $\mathbf{u}$ is the macroscopic velocity, c_s is the lattice speed of sound, and w_k are the D2Q9 weights,

$$w_0 = \frac{4}{9}, \quad w_{1\text{–}4} = \frac{1}{9}, \quad w_{5\text{–}8} = \frac{1}{36}. \tag{6}$$

In the lattice units used by the simulator, one has $c_s^2 = 1/3$ and $\delta t = \delta x = 1$, so Eq. (5) reduces to the explicit polynomial form implemented in the JavaScript code:

$$f_k^{\mathrm{eq}} = w_k \, \rho \left(1 + 3 \, \mathbf{c}_k \cdot \mathbf{u} + \frac{9}{2} \, (\mathbf{c}_k \cdot \mathbf{u})^2 - \frac{3}{2} \, \mathbf{u}^2 \right). \tag{7}$$

The macroscopic density and velocity are recovered from the zeroth- and first-order moments of the distribution functions,

$$\rho(\mathbf{x}, t) = \sum_{k=0}^{8} f_k(\mathbf{x}, t), \tag{8}$$

$$\rho \, \mathbf{u}(\mathbf{x}, t) = \sum_{k=0}^{8} \mathbf{c}_k \, f_k(\mathbf{x}, t). \tag{9}$$

These relations are evaluated at each time step to update the fields $(\rho, \mathbf{u})$ on the 400×200 lattice used by the simulator.

1.2.3 Viscosity, Reynolds number and Mach number

In the BGK lattice Boltzmann formulation, the kinematic viscosity ν is related to the relaxation time τ through

$$\nu = c_s^2 \left(\tau - \frac{1}{2} \right) \delta t. \tag{10}$$

With $c_s^2 = 1/3$ and $\delta t = 1$, Eq. (10) simplifies to

$$\nu = \frac{1}{3} \left(\tau - \frac{1}{2} \right). \tag{11}$$

In the implementation, the user-controlled "viscosity" parameter is directly identified with ν and the relaxation time is updated as

$$\tau = \frac{1}{2} + 3 \, \nu, \tag{12}$$

which is exactly the relation encoded in the JavaScript solver (line `tau = 0.5 + 3 * visc;`).

The inflow speed is prescribed in terms of a lattice Mach number,

$$\mathrm{Ma} = \frac{U_0}{c_s}, \tag{13}$$

where U_0 is the characteristic inlet velocity. In lattice units, the code sets

$$U_0 = \mathrm{Ma} \, c_s \approx 0.577 \, \mathrm{Ma}, \tag{14}$$

ensuring that Ma $\lesssim 0.5$ so that compressibility effects remain small and the incompressible Navier–Stokes limit is recovered. For a characteristic length L (the airfoil chord) and given ν, the Reynolds number is

$$\mathrm{Re} = \frac{U_0\, L}{\nu}, \tag{15}$$

and the simulator reports an approximate value (e.g. $\mathrm{Re} \approx 5 \times 10^4$) based on the current user-selected parameters.

1.3 Derived fields for visualization

1.3.1 Vorticity field

The vorticity field used for visualization corresponds to the out-of-plane component in two dimensions,

$$\omega_z = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}, \tag{16}$$

which is approximated on the lattice by centered finite differences,

$$\omega_z(i,j) \approx \frac{v(i+1,j) - v(i-1,j)}{2\,\delta x} - \frac{u(i,j+1) - u(i,j-1)}{2\,\delta x}. \tag{17}$$

The "vorticity field" rendering mode maps $|\omega_z|$ (combined with the local speed magnitude) to a color scale to highlight shear layers, vortex shedding and separated regions downstream of the airfoil.

1.3.2 Velocity magnitude and pressure field

The "velocity magnitude" visualization uses

$$|\mathbf{u}| = \sqrt{u^2 + v^2}, \tag{18}$$

scaled and mapped to a colour palette to show acceleration and deceleration in the flow.

The "pressure field" visualization leverages the near-incompressible equation of state for the lattice Boltzmann fluid,

$$p = c_s^2\,(\rho - \rho_0), \tag{19}$$

with $\rho_0 \approx 1$. In practice, the simulator plots a normalized measure of $|\rho - \rho_0|$ to obtain qualitative pressure distributions over the airfoil surface and in the wake.

1.4 Boundary conditions and airfoil geometry

1.4.1 Bounce-back no-slip boundary

Solid boundaries, including the airfoil surface, are treated with a standard bounce-back scheme. A Boolean mask identifies obstacle nodes, and populations streaming into a solid cell are reflected back into the opposite direction,

$$f_k(\mathbf{x}_{\mathrm{fluid}}, t + \delta t) = f_{\bar{k}}(\mathbf{x}_{\mathrm{fluid}}, t), \tag{20}$$

where $\bar{k}$ denotes the direction opposite to k. This enforces a no-slip condition at the fluid–solid interface in a simple and numerically robust manner.

1.4.2 Inlet and outlet conditions

At the left boundary ($x = 0$), a constant inflow velocity $\mathbf{u} = (U_0, 0)$ is imposed by resetting the incoming distributions to their equilibrium values $f_k^{\mathrm{eq}}(\rho_0, \mathbf{u})$.

At the right boundary ($x = L_x$), a simple convective (zero-gradient) condition is mimicked by copying the populations from the penultimate column. Although approximate, this treatment is sufficient for qualitative demonstrations in a finite domain and keeps the algorithm lightweight.

1.4.3 Airfoil geometry via Joukowski-type mapping

The airfoil shape is generated from a Joukowski-type conformal mapping applied to a circle in the complex plane, which produces a cambered airfoil geometry with a controllable thickness ratio. Denoting the complex coordinate by

$$z = x + iy, \tag{21}$$

the classical Joukowski transform can be written as

$$\zeta = z + \frac{1}{z}, \tag{22}$$

which maps a circle in the z-plane into a lifting airfoil in the ζ-plane. In practice, the implementation uses a parametrized circle with eccentricity and camber, then applies a Joukowski-type mapping and scales the resulting coordinates to match a chord length L on the numerical grid.

The angle of attack is introduced by rotating the mapped airfoil by a user-specified angle α before projection onto the lattice:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}. \tag{23}$$

The resulting polygonal contour is discretized into a set of points and rasterized into the lattice mask. This procedure allows interactive control of stall and separation behavior in the flow visualization by simply adjusting the slider that controls the angle of attack.

1.5 Rationale for the LBM–D2Q9 HTML/JavaScript implementation

The lattice Boltzmann D2Q9 formulation with a BGK collision operator was chosen because it offers an excellent compromise between physical fidelity and computational simplicity:

- **Locality of operations:** collision and streaming involve only nearest-neighbour nodes, enabling fast updates without solving a global Poisson equation for pressure.
- **Real-time performance:** the method is fully explicit and highly parallelizable, which makes it suitable for execution at interactive frame rates in JavaScript, even on modest hardware.
- **Simplicity of implementation:** a complete CFD solver fits into a single HTML/JS file without external libraries, ensuring browser compatibility and ease of distribution.
- **Accurate low-Mach aerodynamics:** the LBM correctly reproduces key phenomena of incompressible airfoil flows, such as boundary layers, separation, recirculation bubbles and vortex shedding, provided that the Mach number remains sufficiently small.
- **Robust geometry handling:** the bounce-back method requires only a Boolean mask to represent solid boundaries, which is ideal for implementing complex shapes and interactive airfoil motion.
- **Educational and demonstrative value:** real-time visualization of vorticity, velocity magnitude and pressure fields enhances the intuitive understanding of aerodynamic concepts such as lift generation, stall and wake dynamics.

The choice of `<!DOCTYPE html>` and a self-contained web implementation ensures that the simulator is:

- platform-independent and instantly usable in any modern browser;

- GPU-accelerated through the Canvas API and `ImageBitmap`;
- lightweight and distributable as a single file;
- ideally suited for teaching, live demonstrations and interactive CFD exploration.

Overall, the HTML/JavaScript + LBM–D2Q9 approach provides a physically meaningful yet computationally lightweight framework that is particularly well adapted to real-time visualization of flow around airfoils and to the communication of core aerodynamic concepts in an accessible, browser-based environment.