10 Discrete Elements, Particles, and Automata

molecular dynamics (*MD*)
  start from fundamental physics
  femtosecond time steps
  can take days per microseconds
  Molecular dynamics simulations: advances and applications, [Hospital *et al.*, 2015]
  Picosecond to millisecond structural dynamics in human ubiquitin, [Lindorff-Larsen *et al.*, 2016]
  Molecular dynamics simulation for all, [Hollingsworth & Dror, 2018]
  satisfy spirit but not letter
  redefine molecule
  macromolecular dynamics
  mesh-free, computation happens where particles are
  handle complexity where PDE approximations can break down
  massively parallel
  trillions of particles [Tchipev *et al.*, 2019]

10.1 PARTICLE SYSTEMS

10.1.1 Smoothed Particle Hydrodynamics

Smoothed Particle Hydrodynamics (*SPH*)
  function $f(\vec{r})$
  expand integral as sum

$$\int f(\vec{r})\ dr = \sum_i V_i f_i \tag{10.1}$$

  interpolate with weighting function

$$f(\vec{r}) = \sum_i V_i f_i W(\vec{r} - \vec{r}_i) \tag{10.2}$$

  preserve normalization

$$\int W(\vec{r})\ d\vec{r} = 1 \tag{10.3}$$

kernel function approximation, Chapter 14
relate volume element to a density and mass, introducing particles
$V_i = m_i/\rho_i$

$$f(\vec{r}) = \sum_i \frac{m_i}{\rho_i} f_i W(\vec{r} - \vec{r}_i) \tag{10.4}$$

derivatives go into the sum

$$\nabla f(\vec{r}) = \sum_i \frac{m_i}{\rho_i} f_i \nabla W(\vec{r} - \vec{r}_i) \tag{10.5}$$

Navier-Stokes

$$\rho \left(\frac{\partial \vec{v}}{\partial t} + \vec{v} \cdot \nabla \vec{v} \right) = -\nabla p + \rho \vec{g} + \mu \nabla^2 \vec{v} \tag{10.6}$$

quantities relative to particles in Lagrangian (moving frame) vs Eulerian (fixed frame)
Lagrangian derivatives

$$\begin{aligned} \frac{Df}{Dt} &= \frac{f(\vec{x} + \vec{v}dt, t + dt) - f(\vec{x}, t)}{dt} \\ &\approx \frac{f(\vec{x}, t) + \frac{\partial f}{\partial t} dt + \vec{v} \cdot \nabla f dt - f(\vec{x}, t)}{dt} \\ &= \frac{\partial f}{\partial t} + \vec{v} \cdot \nabla f \end{aligned} \tag{10.7}$$

$$\rho \frac{D\vec{v}}{dt} = -\nabla p + \rho \vec{g} + \mu \nabla^2 \vec{v} \tag{10.8}$$

pressure

$$\nabla p(\vec{r}_j) = \sum_i \frac{m_i}{\rho_i} p_i \nabla W(\vec{r}_i - \vec{r}_j) \tag{10.9}$$

density

$$\begin{aligned} \rho(\vec{r}_j) &= \sum_i \frac{m_i}{\rho_i} \rho_i W(\vec{r}_i - \vec{r}_j) \\ &= \sum_i m_i W(\vec{r}_i - \vec{r}_j) \end{aligned} \tag{10.10}$$

viscosity, depends on relative velocity

$$\mu \nabla^2 \vec{v}(\vec{r}_j) = \mu \sum_i \frac{m_i}{\rho_i} (\vec{v}_i - \vec{v}_j) \nabla^2 W(\vec{r} - ver_i) \tag{10.11}$$

equation of state relates pressure to density
converted PDE to ODEs for particle motion
interpolate for fields

computation happens where the particles are, not over a grid

Smoothed particle hydrodynamics: theory and application to non-spherical stars, [Gingold & Monaghan, 1977]

Smoothed particle hydrodynamics: A review. In numerical modeling of nonlinear stellar pulsation: Problems and prospects, [Benz, 1990]

Smoothed particle hydrodynamics, [Monaghan, 1992]

Particle-based fluid simulation for interactive applications, [Müller *et al.*, 2003]

Astrophysical smooth particle hydrodynamics, [Rosswog, 2009]

Smoothed Particle Hydrodynamics, [Fraga Filho *et al.*, 2019]

10.1.2 Other Acronyms

Material Point Method (*MPM*)

used for solid mechanics

hybrid particle-grid method

Lagrangian particles represent material properties with internal state variables

Eulerian grid used for computing spatial derivatives in governing equations

alternate Lagrangian and Eulerian steps

used for a range of governing equations

Application of a particle-in-cell method to solid mechanics, [Sulsky *et al.*, 1995]

Issues with the Material Point Method for geotechnical modelling, and how to address them, [Augarde *et al.*, 2018]

The Material Point Method for Simulating Continuum Materials, [Jiang *et al.*, 2016]

Material point method: Overview and challenges ahead, [Sołowski *et al.*, 2021]

A material point method for viscoelastic fluids, foams and sponges, [Ram *et al.*, 2015]

Peridynamics (*PD*)

start from integral form of solid mechanics governing equations

use particle expansion to replace integral with sum

particles interact through bonds within a horizon

fracture represented by breaking bonds

Peridynamic theory of solid mechanics, [Silling & Lehoucq, 2010]

Convergence of peridynamics to classical elasticity theory, [Silling & Lehoucq, 2008]

A comparative review of peridynamics and phase-field models for engineering fracture mechanics, [Diehl *et al.*, 2022]

Recent progress in mathematical and computational aspects of peridynamics, [D'Elia *et al.*, 2018]

A review on the developments of peridynamics for reinforced concrete structures, [Hattori *et al.*, 2021]

A review of benchmark experiments for the validation of peridynamics models, [Diehl *et al.*, 2019]

10.2 DISCRETE ELEMENT METHODS

SPH starts from PDE and expands in particles

Section 4.1 found PDE from masses and springs

Problem 8.1 found masses and springs from FEA
can start from masses and springs
Discrete Element Method (*DEM*)
originally for granular media, contact interactions
become generic term
generalize to coarse-grained particles

10.2.1 Forces

particle sphere, shape, orientation
generalize to other force laws
memoryless, bonds, internal DOF
linear, nonlinear springs

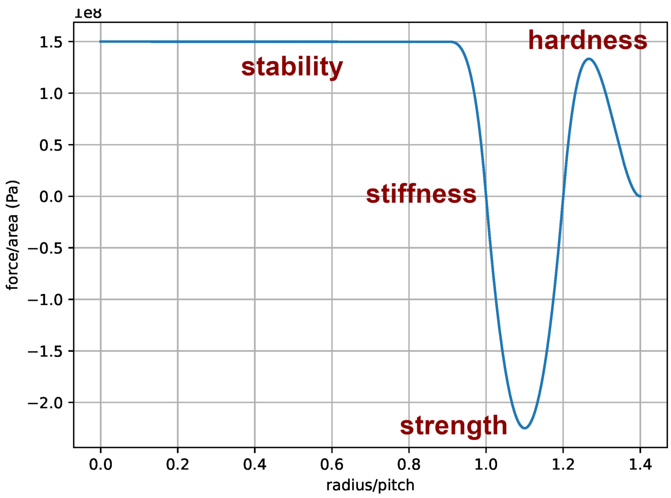

Figure 10.1. Force law.

Newton's equation of motion $\vec{F} = m\vec{a}$
add up forces on particle

10.2.2 Integration

Euler

$$\begin{aligned}
\vec{a}(t+h) &= \vec{F}\left(\vec{x}(t)\right)/m \\
\vec{v}(t+h) &= \vec{v}(t) + h\vec{a}(t) \\
\vec{x}(t+h) &= \vec{x}(t) + h\vec{v}(t)
\end{aligned} \tag{10.12}$$

second order error
semi-implicit

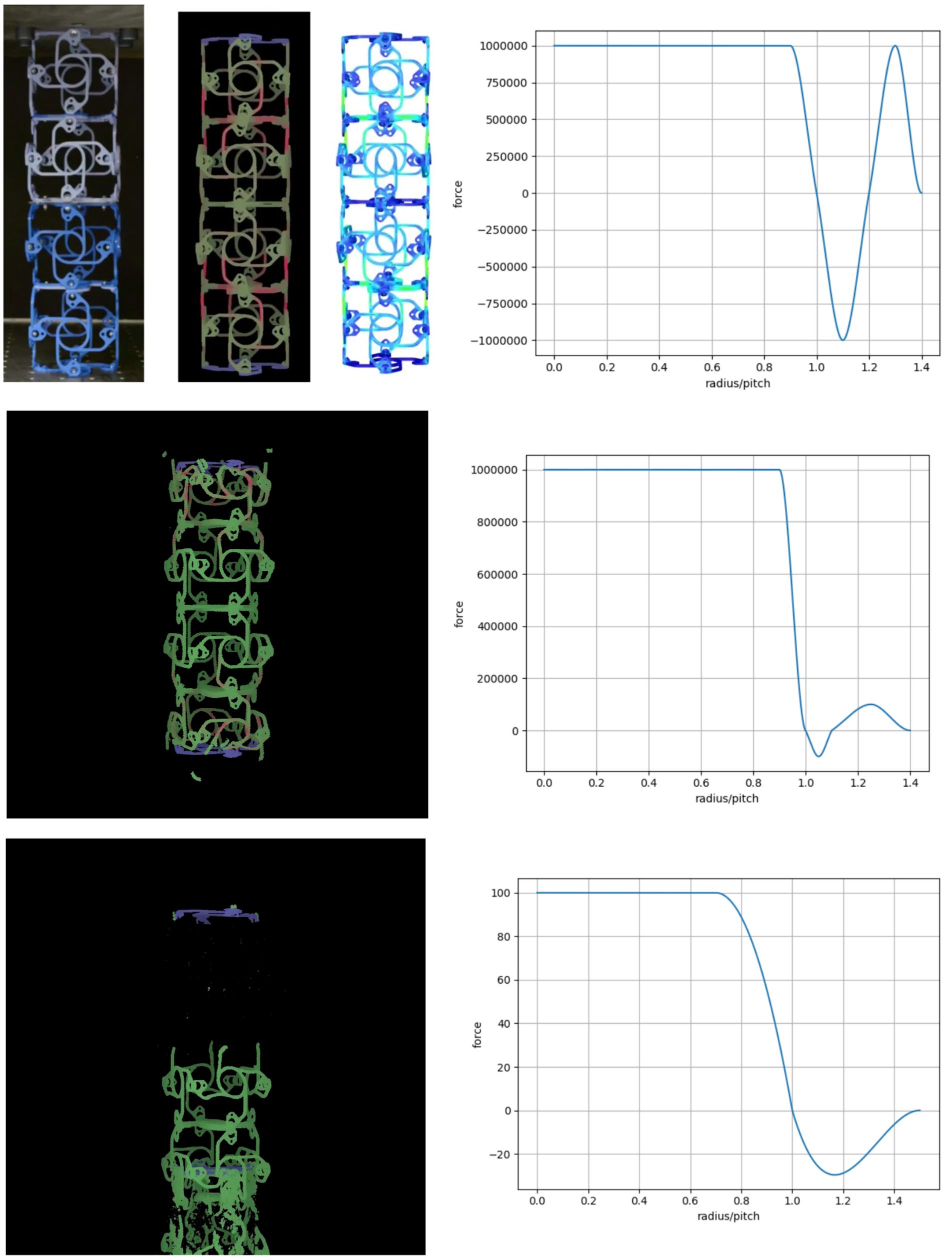

Figure 10.2. Force laws.

$$
\begin{aligned}
\vec{a}(t+h) &= \vec{F}\left(\vec{x}(t)\right)/m \\
\vec{v}(t+h) &= \vec{v}(t) + h\vec{a}(t) \\
\vec{x}(t+h) &= \vec{x}(t) + h\vec{v}(t+h)
\end{aligned}
\tag{10.13}
$$

same order
symplectic, conserve phase space volume
position Verlet
expand position forwards

$$\vec{x}(t+h) = \vec{x}(t) + h\vec{v}(t) + h^2\vec{a}(t)/2 + \ldots \tag{10.14}$$

backwards

$$\vec{x}(t-h) = \vec{x}(t) - h\vec{v}(t) + h^2\vec{a}(t)/2 + \ldots \tag{10.15}$$

add

$$\vec{x}(t+h) = 2\vec{x}(t) - \vec{x}(t-h) + h^2\vec{a}(t) \tag{10.16}$$

fourth order error
need to keep old position
can derive from semi-implicit

$$\begin{aligned}
\vec{x}(t+h) &= \vec{x}(t) + h\,(\vec{v}(t) + h\vec{a}(t)) \\
&\approx \vec{x}(t) + h((\vec{x}(t) - \vec{x}(t-h))/h + h\vec{a}(t)) \\
&= 2\vec{x}(t) - \vec{x}(t-h) + h^2\vec{a}(t)
\end{aligned} \tag{10.17}$$

velocity Verlet

$$\begin{aligned}
\vec{x}(t+h) &= \vec{x}(t) + h\vec{v}(t) + \frac{h^2}{2}\vec{a}(t) \\
\vec{v}(t+h) &= \vec{v}(t) + h\vec{a}(t) + \frac{h^2}{2}\frac{d\vec{a}}{dt} \\
\frac{d\vec{a}}{dt} &\approx (\vec{a}(t+h) - \vec{a}(t))/h \\
\vec{v}(t+h) &= \vec{v}(t) + \frac{h}{2}(\vec{a}(t+h) + \vec{a}(t))
\end{aligned} \tag{10.18}$$

need to keep old $\vec{a}$

$$\begin{aligned}
\vec{v}(t+h/2) &= \vec{v}(t) + h\vec{a}(t)/2 \\
\vec{x}(t+h) &= \vec{x}(t) + h\vec{v}(t+h/2) \\
\vec{x}(t+h) &\Rightarrow \vec{a}(t+h) \\
\vec{v}(t+h) &= \vec{v}(t+h/2) + h\vec{a}(t+h)/2
\end{aligned} \tag{10.19}$$

substitution gives same algorithm
saves memory
dissipation proportional to $\vec{v}$
internal, inertial, numerical

10.2.3 Stability

literally stiff problem

numerical stability must not propagate faster than the speed of sound

linear array of masses and springs speed equal to square root of the elastic force divided by the mass

convert to a time step by dividing lattice pitch by sound velocity

ms - μs range

10.2.4 Geometry

Maxwell criterion DOF vs constraints

rectangular lattice shear, cross-brace bookcase

sphere packing

2D triangular

3D HCP FCC

cannon ball stack

elastic constants not isotropic

pour in particles

sample with noise distribution

metamaterial vary force law

10.2.5 Sorting

$O(N^2)$ particle interactions, $O(N)$ sort

lexicographic sort into bins

fixed size, empty space

cumulative sum sort to find bin pointers

Problem 9.1

A discrete numerical model for granular assemblies, [Cundall & Strack, 1979]

Large-scale discrete element modeling in a fluidized bed, [Sakai *et al.*, 2010]

Discrete element method for modelling solid and particulate materials, [Tavarez & Plesha, 2007]

Discrete element method to simulate continuous material by using the cohesive beam model, [André *et al.*, 2012]

Computer" experiments" on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules, [Verlet, 1967]

Geometric numerical integration illustrated by the Störmer–Verlet method, [Hairer *et al.*, 2003]

XLV. On reciprocal figures and diagrams of forces, [Maxwell, 1864]

Particulate discrete element modelling: a geomechanics perspective, [O'Sullivan, 2011]

10.3 CELLULAR AUTOMATA

SPH and DEM use discrete representations, still require algorithms for solution

what about using computation as representation?
lattice gases
cellular automata
CA
The idea was developed by Ulam [Finkel & Edelman, 1985], von Neumann [von Neumann, 1966], and colleagues [Shannon & McCarthy, 1956] in the 1950s. A classic example of a CA is Conway's *Game of Life* [Gardner, 1970], in which occupied sites on a grid get born, survive, or die in succeeding generations based on the number of occupied neighboring sites. Any CA has the same elements: a set of connected sites, a discrete set of states that are allowed on the sites, and a rule for how they are updated. We will start by studying a slighly more complicated system that recovers the Navier-Stokes fluid equations, and then will consider the more general question of how cellular automata relate to computation.

10.3.1 Lattice Gas Automata

Hydrodynamics was one of the earliest and best-developed application areas of cellular automata. A cellular automata model of a fluid (traditionally called a lattice gas) is specified by the geometry of a lattice, by the discrete states permitted at each site, and by an update rule for how the states change based on their neighbors. Both partial differential equations and molecular dynamics models use real numbers (for the values of the fields, or for the particle positions and velocities). A lattice gas discretizes everything so that just a few bits describe the state of each site on a lattice, and the dynamics reduce to a simple look-up table based on the values of the neighboring sites. Each site can be considered to be a parcel of fluid. The rules for the sites implement a "cartoon" version of the underlying microscopic dynamics, but should be viewed as operating on a longer length scale than individual particles. We will see that the conservation laws that the rules satisfy determine the form of the equivalent partial differential equations for a large lattice, and that the details of the rules set the parameter values.

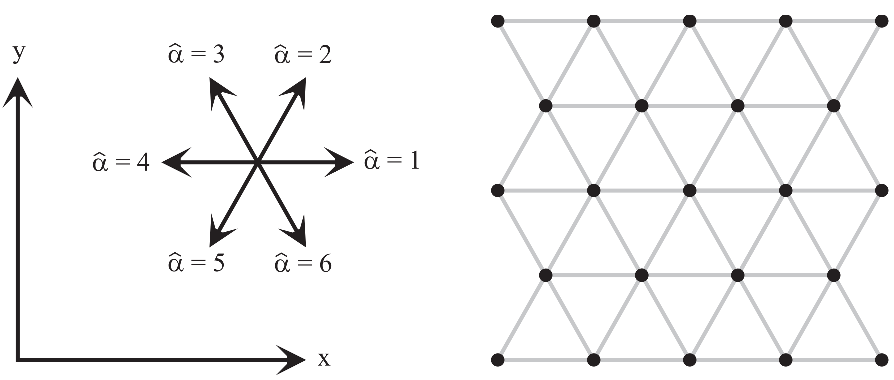

Figure 10.3. Direction indices for a triangular lattice.

A historically important example of a lattice gas is the FHP rule (named after its inventors, Frisch, Hasslacher, and Pomeau [Frisch *et al.*, 1986]). This operates in 2D on a triangular lattice, and the state of each site is specified by six bits (Figure 10.3). Each bit represents a particle on one of the six links around the site, given by the unit vectors $\hat{\alpha}$. On each link a particle can either be present or absent, and all particles have the same unit velocity: in the absence of collisions, they travel one lattice step ahead in one time step. The simple update rule proceeds in two stages, chosen to conserve particle number and momentum (Figure 10.4). First, collisions are handled. At the beginning of the step a particle on a link is considered to be approaching the site, and after the collision step a particle on a link is taken to be leaving the site. If a site has two particles approaching head-on, they scatter. A random choice is made between the two possible outgoing directions that conserve momentum (always choosing one of them would break the symmetry of the lattice). Because of the large number of sites in a lattice gas it is usually adequate to approximate the random decision by simply switching between the two outgoing choices on alternate site updates. If three particles approach symmetrically, they scatter. In all other configurations not shown the particles pass through the collision unchanged. After the collision step there is a transport step, in which each particle moves by one unit in the direction that it is pointing and arrives at the site at the far end of the link. While these rules might appear to be somewhat arbitrary, it will turn out that the details will not matter for the form of the governing equations, just the symmetries and conservation laws.

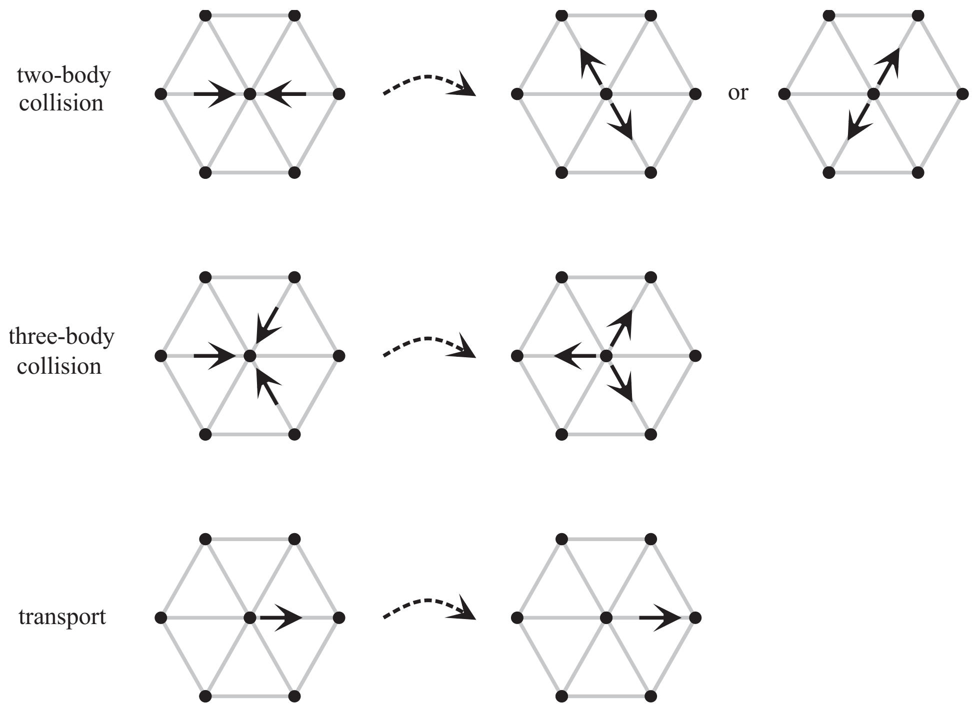

Figure 10.4. Update rules for an FHP lattice gas.

A simpler rule related to FHP is HPP (named after Hardy, de Pazzis, and Pomeau [Hardy *et al.*, 1976]), which operates on a square lattice. Each site is specified by four bits, and direction-changing collisions are allowed only when two particles meet head-on (unlike FHP, here there is only one possible choice for the exit directions after scattering). We will see that HPP and FHP, although apparently quite similar, behave very differently.

Let's label time by an integer T, the lattice sites by a vector $\vec{X}$, and the lattice directions by a unit vector $\hat{\alpha}$. If we start an ensemble of equivalent lattices off with the same update rule but different random initial conditions, we can define $f_\alpha(\vec{X}, T)$ to be the fraction of sites $\vec{X}$ at time T with a particle on link $\hat{\alpha}$. In the limit of a large ensemble, this fraction becomes the probability to find a particle on that link. Defining this probability will let us make a connection between the lattice gas and partial differential equations.

At each time step, in the absence of collisions, the fraction of particles at site $\vec{X}$ at time T pointing in direction $\hat{\alpha}$ will move one step in that direction:

$$f_\alpha(\vec{X} + \hat{\alpha}, T + 1) = f_\alpha(\vec{X}, T) \quad . \tag{10.20}$$

Let's introduce new rescaled variables $\vec{x} = \delta_x \vec{X}$ and $t = \delta_t T$ in terms of the (small) space step δ_x and time step δ_t. Substituting in these variables, collision-free transport becomes

$$f_\alpha(\vec{x} + \delta_x \hat{\alpha}, t + \delta_t) - f_\alpha(\vec{x}, t) = 0 \quad . \tag{10.21}$$

If the probability f_α varies slowly compared to δ_x and δ_t, we can expand equation (10.21) in δ_x and δ_t:

$$\frac{\partial f_\alpha(\vec{x}, t)}{\partial t} \delta_t + \hat{\alpha} \cdot \nabla f_\alpha(\vec{x}, t) \; \delta_x + \mathcal{O}(\delta^2) = 0 \quad . \tag{10.22}$$

Choosing to scale the variables so that $\delta_x = \delta_t$, to first order this becomes

$$\frac{\partial f_\alpha(\vec{x}, t)}{\partial t} + \hat{\alpha} \cdot \nabla f_\alpha(\vec{x}, t) = 0 \quad . \tag{10.23}$$

This equation says that the time rate of change of the fraction of particles at a point is equal to the difference in the rate at which they arrive and leave the point by straight transport (remember equation (8.6)). If collisions are allowed, the time rate of change of f_α will depend on both the spatial gradient and on a collision term Ω_α scattering particles in or out from other directions

$$\frac{\partial f_\alpha(\vec{x}, t)}{\partial t} + \hat{\alpha} \cdot \nabla f_\alpha(\vec{x}, t) = \Omega_\alpha(\vec{x}, t) \quad . \tag{10.24}$$

f_α is the distribution function to find a particle. The collision term Ω_α will in general depend on the distribution function for pairs of particles as well as the one-particle distribution function, and these in turn will depend on the three-particle distribution functions, and so forth. This is called the *BBGKY hierarchy* of equations (Bogolyubov, Born, Green, Kirkwood, Yvon [Boer & Uhlenbeck, 1961]). The *Boltzmann equation* approximates this by assuming that Ω_α depends only on the single-particle distribution functions f_α.

We derived equation (10.24) by making use of the fact that particles travel one lattice site in each time step, and then assuming that f_α varies slowly. Now let's add the conservation laws that have been built into the update rules. The total density of particles ρ at a site $\vec{x}$ is just the sum over the probability to find one in each direction

$$\sum_\alpha f_\alpha(\vec{x}) = \rho(\vec{x}) \quad , \tag{10.25}$$

and the momentum density is the sum of the probabilities times their (unit) velocities

$$\sum_\alpha \hat{\alpha} f_\alpha = \rho \vec{v} \quad . \tag{10.26}$$

Since our scattering rules conserve particle number (the particles just get reoriented), the number of particles scattering into and out of a site must balance

$$\sum_\alpha \Omega_\alpha(\vec{x}) = 0 \quad . \tag{10.27}$$

And since the rules conserve momentum, the net momentum change from scattering must vanish

$$\sum_\alpha \hat{\alpha}\Omega_\alpha = 0 \quad . \tag{10.28}$$

Therefore, summing equation (10.24) over directions,

$$\sum_\alpha \left[\frac{\partial f_\alpha(\vec{x},t)}{\partial t} + \hat{\alpha}\cdot\nabla f_\alpha(\vec{x},t)\right] = \sum_\alpha \Omega_\alpha \tag{10.29}$$

$$\frac{\partial}{\partial t}\sum_\alpha f_\alpha + \sum_\alpha \hat{\alpha}\cdot\nabla f_\alpha = 0$$

$$\frac{\partial \rho}{\partial t} + \nabla\cdot(\rho\vec{v}) = 0 \quad . \tag{10.30}$$

This is the familiar equation for the continuity of a fluid, and has arisen here because we've chosen scattering rules that conserve mass. A second equation comes from momentum conservation, multiplying equation (10.24) by $\hat{\alpha}$ and summing over directions

$$\frac{\partial}{\partial t}\sum_\alpha \hat{\alpha} f_\alpha + \sum_\alpha \hat{\alpha}(\hat{\alpha}\cdot\nabla f_\alpha) = 0 \quad . \tag{10.31}$$

The ith component of this vector equation is

$$\frac{\partial}{\partial t}\sum_\alpha \hat{\alpha}_i f_\alpha + \sum_\alpha\sum_j \hat{\alpha}_i\hat{\alpha}_j \frac{\partial f_\alpha}{\partial x_j} = 0 \quad . \tag{10.32}$$

Defining the momentum flux density tensor by

$$\Pi_{ij} \equiv \sum_\alpha \hat{\alpha}_i\hat{\alpha}_j f_\alpha \quad , \tag{10.33}$$

this becomes

$$\frac{\partial \rho v_i}{\partial t} + \sum_j \frac{\partial \Pi_{ij}}{\partial x_j} = 0 \quad . \tag{10.34}$$

We now have two equations, (10.30) and (10.34), in three unknowns, ρ, $\vec{v}$, and Π. To eliminate Π we can find the continuum form of the momentum flux density tensor by using a *Chapman–Enskog expansion* [Huang, 1987], a standard technique for finding approximate solutions to the Boltzmann equation. We will assume that f_α depends only on $\vec{v}$ and ρ and their spatial derivatives (and not on time explicitly), and so will do an expansion in all possible scalars that can be formed from them. The lowest-order terms of the deviation from the equilibrium uniform configuration are

$$f_\alpha = \frac{\rho}{6}\left(1 + 2\hat{\alpha}\cdot\vec{v} + A\left[(\hat{\alpha}\cdot\vec{v})^2 - \frac{1}{2}|\vec{v}|^2\right]\right.$$
$$\left. + B\left[(\hat{\alpha}\cdot\nabla)(\hat{\alpha}\cdot\vec{v}) - \frac{1}{2}\nabla\cdot\vec{v}\right] + \cdots\right) \quad . \tag{10.35}$$

The terms have been grouped this way to guarantee that the solution satisfies the density and momentum equations (10.25) and (10.26) (this can be verified by writing out the components of each term). In this derivation the only features of the FHP rule that we've used are the conservation laws for mass and momentum, and so all rules with these features will have the same form of the momentum flux density tensor (to this order), differing only in the value of the coefficients A and B.

The *Navier–Stokes* governing equation for a d-dimensional fluid with bulk as well shear viscosity (ignoring gravitational forces) is

$$\frac{\partial\rho\vec{v}}{\partial t} + \rho(\vec{v}\cdot\nabla)\vec{v} = -\nabla p + \eta\nabla^2\vec{v} + (\zeta + \frac{\eta}{d})\nabla(\nabla\cdot\vec{v}) \quad , \tag{10.36}$$

where p is the pressure, η is the shear viscosity, ζ is the bulk viscosity [Batchelor, 1967]. Using the Chapman-Enskog expansion to evaluate the momentum flux density tensor in equation (10.34) and comparing it with the Navier–Stokes equation shows that they agree if $\zeta = 0$, $\eta = \rho\nu = -\rho B/8$ (ν is the kinematic viscosity), $\mu = A/4$, and $p = \rho/2$. Further, in the Boltzmann approximation it is possible (with a rather involved calculation) to find the values of A and B for a given CA rule [Wolfram, 1986].

The simple conservation laws built into our lattice gas have led to the full Navier–Stokes equation; the particular rule determines the effective viscosity of the fluid. While the details of this calculation are complicated, there are some simple and important conclusions. The viscosity for the square lattice (HPP model) turns out to depend on direction and hence is not appropriate for most fluids, but the viscosity of the triangular lattice (FHP model) is isotropic. This profound implication of the lattice symmetry was not appreciated in the early days of lattice gas models, and helped point the way towards the realization that a simple lattice gas model could in fact be very general. In 3D the situation is more difficult because there is not a 3D lattice that gives an isotropic viscosity. However, it can still be achieved by using a quasiperidoic tiling that is not translationally periodic [Boghosian, 1999], or a cut through a higher-dimensional lattice such as the 4D Face-Centered Hyper-Cubic (FCHC) rule [Frisch *et al.*, 1987]. It is not possible to reduce the viscosity in a simple model like FHP (an attribute that is needed for modeling a problem such as air flow), but this can be done by adding more than one particle type or by increasing the size of the neighborhood used for the rule [Dubrulle *et al.*, 1991].

10.3.2 Lattice Boltzmann Method

LGA issues
 statistical, must average
 difficult to tune parameters
 Lattice Boltzmann Method (LBM)

a particle in LGA represented by bit string, in LBM replaced by a real number for the single-particle distribution function

combine equations 10.21 and 10.24, take as a numerical method for f_α

$$f_\alpha(\vec{x} + \delta_x \vec{\alpha}, t + \delta_t) = f_\alpha(\vec{x}, t) + \Omega_\alpha(\vec{x}, t) \tag{10.37}$$

generalized $\vec{\alpha}$ from a single lattice step to vector for multiple velocities

equivalent to a finite-difference form of the Boltzmann equation

again density

$$\sum_\alpha f_\alpha(\vec{x}) = \rho(\vec{x}) \tag{10.38}$$

momentum density

$$\sum_\alpha \vec{\alpha} f_\alpha = \rho \vec{v} \tag{10.39}$$

conservation mass

$$\sum_\alpha \Omega_\alpha(\vec{x}) = 0 \tag{10.40}$$

conservation momentum

$$\sum_\alpha \vec{\alpha}\, \Omega_\alpha = 0 \tag{10.41}$$

BGK approximation to collision term

$$\Omega_\alpha = \frac{f_\alpha - f_\alpha^{eq}}{\tau} \tag{10.42}$$

for f_α^{eq} can take Maxwell-Boltzmann distribution, power series expansion

LBM can be more accurate and faster than SPH, but can also be less robust and versatile

Lattice Boltzmann method for fluid flows, [Chen & Doolen, 1998]

Theory of the lattice Boltzmann method: From the Boltzmann equation to the lattice Boltzmann equation, [He & Luo, 1997]

Lattice-gas cellular automata and lattice Boltzmann models: an introduction, [Wolf-Gladrow, 2004]

Comparison of multiphase SPH and LBM approaches for the simulation of intermittent flows, [Douillet-Grellier *et al.*, 2019]

A model for collision processes in gases. I. Small amplitude processes in charged and neutral one-component systems, [Bhatnagar *et al.*, 1954]

Recovery of the Navier-Stokes equations using a lattice-gas Boltzmann method, [Chen *et al.*, 1992]

Lattice BGK models for Navier-Stokes equation, [Qian *et al.*, 1992]

The lattice Boltzmann method: principles and practice, [Timm *et al.*, 2016]

10.3.3 Computing Automata

Simulating a cellular automata is an particularly simple type of computation. Rather than the many kinds of memory, instructions, and processors in a conventional computer, it requires just storage for the bits on the lattice, and an implementation of the local update rule. This is an extreme form of a *SIMD* (Single Instruction Multiple Data) parallel computer. The update can be performed by using the bits at each site as an index into a look-up table, so the architecture reduces to memory cycling through a look-up table (where the sequence of the memory retrieval determines the lattice geometry). This means that relatively modest hardware can exceed the performance of general-purpose computers for CA problems [Toffoli & Margolus, 1991].

We've seen that a cellular automata computer can simulate fluids. A wide range of other physical systems also can be modeled by cellular automata rules; an important example is the rendering of 3D graphics [Toffoli & Quick, 1997]. Conversely, instead of using a computer to simulate a CA, a CA can be used to simulate a computer. One way to do this is by implementing Boolean logic in CA rules; this was used to first prove their computational universality [Banks, 1971]. This approach is attractive for hardware scaling because it builds in physical constraints; any technology that can perform the local cell updates can execute the same global programs [Gershenfeld *et al.*, 2010].

Alternatively, since CAs can model physical systems, and physical systems can compute, CAs can model physical systems that compute [Margolus, 1984]. Consider the billiard-ball CA in Figure 10.5 (the underlying lattice is not shown). This is similar to a lattice gas: billiard balls move on a lattice with unit velocity, and scatter off of each other and from walls. Two balls colliding generates the `AND` function, and if one of the streams of balls is continuous it generates the `NOT` function of the other input. These two elements are sufficent to build up all of logic [Hill & Peterson, 1993]. Memory can be implemented by delays, and wiring by various walls to guide the balls. The balls can be represented by four bits per site (one for each direction), with one extra bit per site needed to represent the walls.

This kind of computing has many interesting features [Fredkin & Toffoli, 1982]. No information is ever destroyed, which means that it is reversible (it can be run backwards to produce inputs from outputs) [Bennett, 1988], and which in turn means that it can be performed (in theory) with arbitrarily little dissipation [Landauer, 1961]. Reversibility is also essential for designing quantum cellular automata, since quantum evolution is reversible. A quantum CA is much like a classical CA, but it permits the sites to be in a superposition of their possible states [Lloyd, 1993]. This is a promising architecture for building quantum computers based on short-range interactions [Nielsen & Chuang, 2000].

For some, cellular automata are much more than just an amusing alternative to traditional models of computation [Fredkin, 1990]. Most physical theories are based on real numbers. This means that a finite volume of space contains an infinite amount of information, since its state must be specified with real numbers. But if there is an energetic cost to creating information (as there is in most theories), then this implies an infinite amount of energy in a finite space. This is obviously unacceptable; something must bound the information content of space. While such a notion can arise in quantum field theories, CAs start as discrete theories that do not have this problem, and so in many

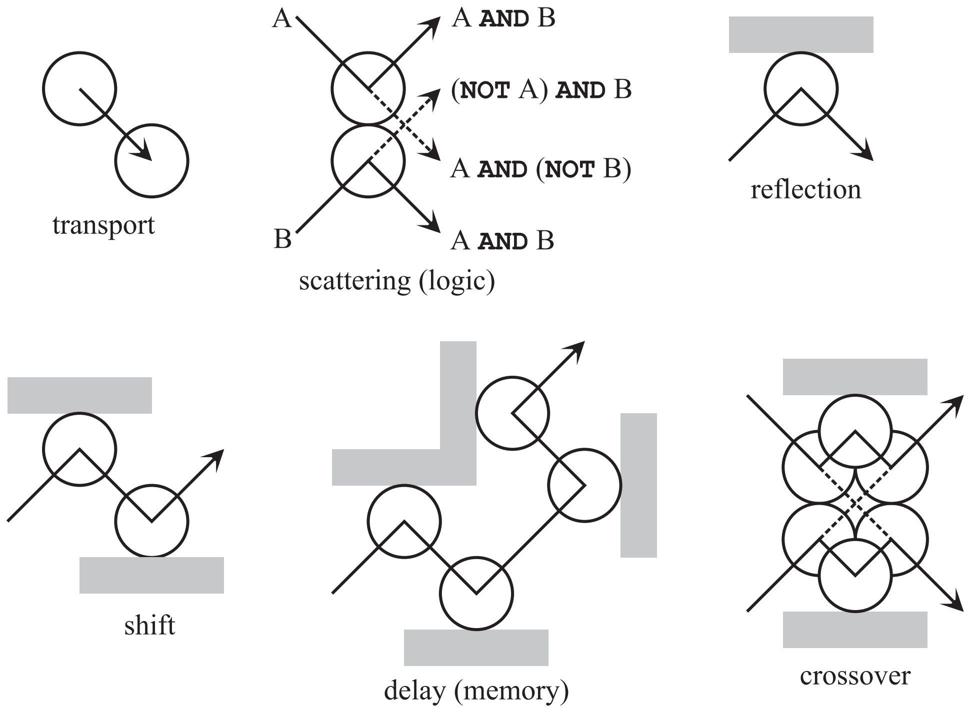

Figure 10.5. Billiard ball logic.

ways they are more satisfying than differential equations as a way to specify governing equations. There is nothing less basic about them than differential equations; which is more "fundamental" depends on whether you are solving a problem with a pencil or a computer.

10.4 SELECTED REFERENCES

[Doolen *et al.*, 1990] Doolen, Gary D. Frisch, Uriel, Hasslacher, Brosl, Orszag, Steven, & Wolfram, Stephen (eds) (1990). *Lattice Gas Methods for Partial Differential Equations*. Santa Fe Institute Studies in the Sciences of Complexity. Reading, MA: Addison-Wesley.

This collection includes many of the important articles, including [Wolfram, 1986] and [Frisch *et al.*, 1987], which work out the connection between lattice gases and hydrodynamics.

[Rothman & Zaleski, 2004] Rothman, Daniel H., & Zaleski, Stephane. (2004). *Lattice-Gas Cellular Automata: Simple Models of Complex Hydrodynamics*. Cambridge: Cambridge University Press.

Reviews the basic theory and extends it to porous media and fluids with multiple components.

10.5 PROBLEMS

(9.1) Redo the finite element beam-bending problem (8.2) with a discrete element method, and compare the results. Use a triangular lattice in 2D, with a linear elastic nearest-neighbor force law.

(9.2) Using a discrete element method, vary the force law to simulate dropping onto a surface bodies that are:

- (*a*) rigid
- (*b*) deformable
- (*c*) viscous
- (*d*) fluid