Simulation of Coral Growth

Kenneth Rohde Christiansen and Aard Keimpema

Department of Mathematics and Computing Science
Rijksuniversiteit Groningen
Blauwborgje 3
NL-9747 AC Groningen

{k.r.christiansen, a.keimpema}@student.rug.nl

Abstract
In this paper we present a method to simulate the growth of coral reefs in two dimensions. This method is based on the lattice-Boltzmann method for simulating the flow of fluids. The lattice-Boltzmann method will be introduced and it will be described how it has been adapted to the given problem. Our two dimensional coral growth algorithm is based on the three dimensional method of Kaandorp et. al[7][8]. We will discuss the Kaandorp algorithm in detail and describe how we adapted it to two dimensions. The paper will be concluded with a discussion of the methods used and the results of our simulation.

Keywords: Coral Reefs, Lattice-Boltzmann, Navier-Stokes, Simulations, Computational Science, Computer Science.

1 Introduction

The lattice-Boltzmann method is a widely-used of method to simulate to flow of fluids and is the central subject of our assignment for the computational science course at the University of Groningen. The objective of this assignment is to do literate research on the lattice-Boltzmann method and apply it to a real world problem.

As both of the authors are fascinated with coral growth simulations after watching sample videos on the internet, we decided to look into using a lattice-Boltzmann method to simulate coral growth. The mentioned videos are part of the Ph.D. thesis by Merks and are available on his website [10]. The work by Merks is based on a method developed by Kaandorp et. al [7][8] and it is these works by Kaandorp that we took as a starting point for our own work. As a three-dimensional simulation would be too computationally intensive, we had to adopt Kaandorps three dimensional methods to two dimensions.

We hope that this paper will give you insight into the usefulness of the lattice-Boltzmann method, the problems of two dimensional simulation and computer simulations in general, as well as some idea on how to simulate coral growth.

The structure of the paper is as follows. In section two we will introduce and describe the lattice-Boltzmann method, discuss the choice of lattice, and further look at the our implementation of the method. In section three we will introduce real life coral growth, the Kaandorp and Sloot-method for simulation coral growth in three dimensions, our two dimensional derived method, as well as look at the results accomplished. In section four we will conclude the paper with discussion of the methods used and the usefulness of our own algorithm.

2 The lattice-Boltzmann method

2.1 Theory

The flow of an incompressible fluid is described by the Navier-Stokes equation

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

and the continuity equation

$$\nabla \cdot \mathbf{u} = 0, \tag{2}$$

where $\mathbf{u}$ is the fluid velocity, ν the kinematical shear viscosity, $P = p/\rho_0$ the kinematical pressure, p the pressure and ρ_0 the mass density. The values of these constants are characteristic for a certain fluid, e.g. $\nu_{\text{water}} = 10^{-6}\,\text{m}^2/\text{s}$ and $\nu_{\text{methanol}} = 0.74 \times 10^{-6}\,\text{m}^2/\text{s}$. Interestingly enough all these liquids are described by the same Navier-Stokes equations even though their molecular makeups are quite different.

To simulate the flow of a (incompressible) liquid one now only needs to solve the Navier-Stokes equations for various times t. In principal this could be done numerically by using e.g. the Crank-Nicolson [1] method. In practice however, this is not feasible. Apart from the obvious problem with efficiency in repeated solving of equation (1) it is also difficult to compute complex boundary conditions using this approach.

Fortunately there are alternatives to solving the Navier-Stokes equation directly. There are a number of conditions a method for simulating the Navier-Stokes equation must satisfy. The first two conditions come from elementary mechanics, namely mass and momentum must be conserved. Furthermore, the method should be isotropic in the sense that a fluid without some external force acting on it has no preferred direction. This in contrast to solid state systems where e.g. in a metal the atoms assume a lattice structure and an electron traversing the lattice can only move along the lattice directions.

In 1986 Frisch, Hasslacher and Pomeau proposed a method based on cellular automata [2], named the FHP method after the authors. In cellular automata we discretise the space into a lattice. In figure 1 we show the FHP lattice, which has a hexagonal topology. Later we discus lattice topologies in more detail, for now we just note that the hexagonal lattice is symmetric in all directions. At each lattice site there can be at most one fluid particle, thus a kind of Pauli exclusion principal is invoked. Time is also discretised, thus time progresses in finite steps Δt. A typical simulation goes as follows: Each particle has a momentum along one of the lattice vectors (except of course stationary particles). At each time step the particle will move one lattice site in the direction of its momentum. When particles collide at a lattice site the momenta of the particles are modified. In general we call methods based on cellar automata Lattice Gas Automata (LGA).

While LGA are very practical in the sense that it is simple to implement it also has a number of problems. For example the method is very sensitive to noise making it difficult to obtain a high level of accuracy. A considerable refinement over LGA is methods based on the Boltzmann equation, dubbed lattice-Bolzman (LB) methods. These methods solve the problems concerning LGA but are still simple to implement as we shall shortly see. The LB method was first developed by McNamara and Zanetti in 1988 who derived the method from LGA. In 1993 Shan en Chen[6] developed a LB method in the context of flow in porous media.

In the LB method, space is again discretized onto a lattice. At each lattice site there can be a certain density of moving or immobile particles. This in contrast to LGA where there could be only a single particle at each lattice site. The basic idea of the LB method consists of a two phase approach.

- **First phase** (propagation phase): Particles jump from on lattice site to the other.
- **Second phase** (collision phase): Particles collide with each other and their momenta change.

The dynamics of a fluid consisting of S different phases on a lattice with b lattice directions is governed by the following lattice-Boltzmann equation,

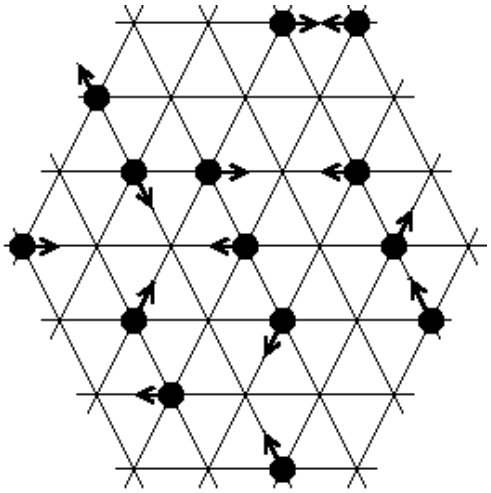

Figure 1: The (hexagonal) FHP lattice, particles can only move along the lattice vectors and thus have discrete momenta.

$$n_a^\sigma(\mathbf{x}+\mathbf{e}_a, t+1) - n_a^\sigma(\mathbf{x}, t) = \Omega_a^\sigma(\mathbf{x}, t) \tag{3}$$

$$\sigma = 1, \cdots, S, \quad a = 0, \cdots, b. \tag{4}$$

Here $n_a^\sigma(\mathbf{x}, t)$ is the density of particles of phase σ moving in direction $\mathbf{e}_a$, where $\mathbf{e}_0$ indicates an immobile particle. For $1 \leqslant i \leqslant b$ we have $|\mathbf{e}_i| = c$ and $|\mathbf{e}_0| = 0$, where c is the lattice constant. $\Omega_a^\sigma(\mathbf{x}, t)$ is the collision operator, which is defined as

$$\Omega_a^\sigma(\mathbf{x}, t) = -\frac{1}{\tau_\sigma}\left[n_a^\sigma(\mathbf{x}, t) - n_a^{\sigma(eq)}(\mathbf{x}, t)\right], \tag{5}$$

where $n_a^{\sigma(eq)}(\mathbf{x}, t)$ is the equilibrium particle density. Thus the LB method will eventually reach an equilibrium state and will then remain unchanged. The equilibrium particle density is given by

$$n_0^{\sigma(eq)}(\mathbf{x}) = n^\sigma(\mathbf{x})\left[d_0 - \frac{1}{c^2}\mathbf{u}^2\right], n_a^{\sigma(eq)}(\mathbf{x}) = n^\sigma(\mathbf{x})\left[\frac{1-d_0}{b} + \frac{D}{c^2 b}\mathbf{e}_a \cdot \mathbf{u} + \frac{D(D+2)}{2c^4 b}\mathbf{e}_a\mathbf{e}_a : \mathbf{u}\mathbf{u} - \frac{D}{2bc^2}\mathbf{u}^2\right], \tag{6}$$

where $n^\sigma = \sum_a \mathbf{n}_a^\sigma$ is the number density, $\mathbf{u}$ is the average particle momentum, d_0^σ is a constant that depends on the compressibility of the fluid and D the dimension of the lattice. The average particle momentum is given by

$$\mathbf{u} = \frac{\sum_\sigma^S (m^\sigma \sum_a n_a^\sigma \mathbf{e}_a/\tau_\sigma)}{\sum_\sigma^S m^\sigma n^\sigma(\mathbf{x})/\tau_\sigma} \tag{7}$$

Up until now no interactions are included. Interactions are modeled as forces acting on particles. Since force is proportional to the change of momentum we simply put

$$n^\sigma(\mathbf{x})\mathbf{u}'(\mathbf{x}) = n^\sigma \mathbf{u}(\mathbf{x}) + \tau_\sigma \frac{dp^\sigma}{dt}(\mathbf{x}), \tag{8}$$

where $n^\sigma(\mathbf{x})\mathbf{u}(\mathbf{x})$ is the particle density before the collision phase. There are two kinds of interactions

1. Fluid-Fluid interactions which models the interaction between the fluid components. For example in a oil-water mixture the water pushes the oil to the surface because of its greater density.

2. Fluid-Solid interactions which model the interaction between the fluid and solid objects present in the fluid.

Fluid-fluid interaction is modeled by an interaction constant $G_{\sigma\sigma'}$ where $G_{\sigma\sigma'} = 0$ for $\sigma = \sigma'$, for a two component fluid mixture the interaction becomes

$$G_{\sigma\sigma'} = \begin{bmatrix} 0 & \alpha \\ \alpha & 0 \end{bmatrix},$$

where α is a constant. The force exerted between fluid components then becomes

$$\frac{dp^{\sigma}}{dt}(\mathbf{x}) = -n^{\sigma}(\mathbf{x}) \sum_{\sigma'}^{S} G_{\sigma\sigma'} \sum_{a=0}^{b} n^{\sigma'}(\mathbf{x} + \mathbf{e}_a)\mathbf{e}_a. \tag{9}$$

If $G_{\sigma\sigma'}$ is large enough the fluid components will completely separate from each other.

Fluid-solid interaction can be divided into two categories

1. The solid attracts the fluid (wetting fluid)
2. The solid repels the fluid (non-wetting fluid)

The fluid-solid interaction is modeled by an interaction constant G_a^{σ} which is negative for wetting- and positive for non-wetting fluids. The interaction force is given by

$$\frac{dp^{\sigma}}{dt}(\mathbf{x}) = -n^{\sigma}(x) \sum_{a} G_a^{\sigma} s(\mathbf{x} + \mathbf{e}_a)\mathbf{e}_a. \tag{10}$$

In addition to the fluid-solid interaction force we also need to ensure that the liquid does not actually penetrate the solid, in other words we have to handle the collision between the solid and the liquid. This can be done by simply reversing the momentum of a particle when it collides with a solid, this is called the bounce-back rule. Thus if a particle has momentum vector $\mathbf{e}_a$ before the collision it will have momentum $-\mathbf{e}_a$ after the collision.

2.2 The lattice

The choice of the lattice structure is an essential step in the simulation. The simplest approach would be to use a cubic lattice. In a cubic lattice all lattice sites are equidistancely spaced, particles can move horizontally, vertically and diagonally thus each lattice site has eight neighbours. Including also stationary particles the lattice can then be represented by nine lattice vectors $\mathbf{e}_a$ as illustrated in figure 2. The implementation of a square lattice is extremely simple; in essence one only needs to maintain a two dimensional array. However, the square lattice has a major problem in that not all lattice vectors have the same size. The diagonal lattice vectors are a factor $\sqrt{2}$ longer then the horizontal and vertical lattice sites[1]. This makes it very difficult to simulate isotropic fluid flow on such a lattice.

A more suited candidate lattice for fluid flow is the hexagonal lattice. In a hexagonal lattice each lattice site has six neighbours which are all at equal distance from the lattice site. Thus the lattice can be represented by seven lattice vectors $\mathbf{e}_a$ as is illustrated in figure 3. While the hexagonal lattice solves the isotropicy problem of the square lattice, it is also more difficult to program. Typically the hexagonal lattice is stored in a two dimensional array but now extra code is needed to correctly compute the six neighbours of a lattice site. The extra code adds to the complexity of the simulation which can result in extra debug time.

[1] $|\mathbf{e}_0| = 0$ as it does not constitute movement.

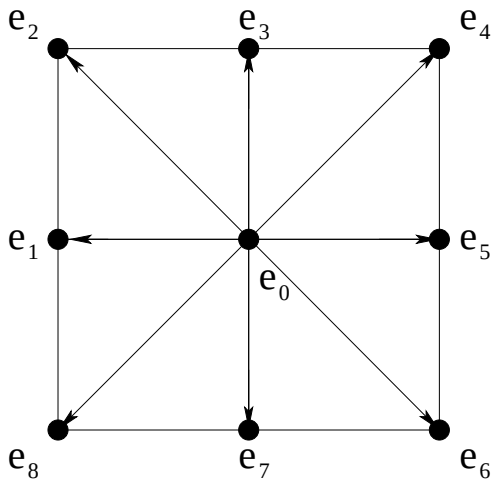

Figure 2: A square lattice, particles can move from one site to its nearest neighbours along lattice vectors $\mathbf{e}_\alpha$.

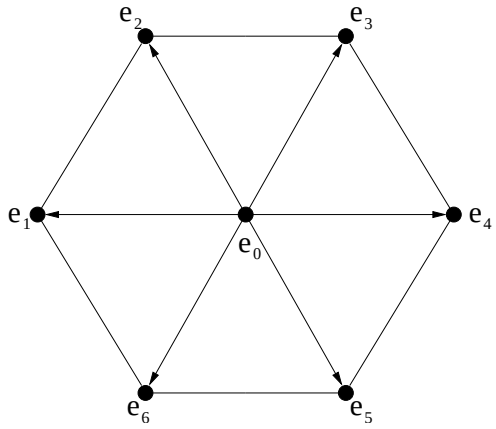

Figure 3: A hexagonal lattice, each lattice site has six neighbours that can be reached along the lattice vectors $\mathbf{e}_\alpha$.

2.3 Implementation

We will now present our implementation of the lattice-Boltzmann method. The implementation was done in C++ as it is a well established language that both authors master. The actual implementation is a one-to-one translation of the lattice-Boltzmann method as described in section 2.1 into source code.

1. Initialize the data structures
2. do N time steps
3. Create equilibrium lattice densities $n_\alpha^{\sigma(eq)}$.
4. Compute new particle densities n_α.
5. Apply boundary conditions.

Figure 4: Logical structure of the simulation program.

In figure 4 we show the logical structure of our simulation program. Something which needs extra expla-

nation are the boundary conditions. Here there are basically two choices, either we keep the boundaries at some constant value or we apply periodic boundary conditions. The type used in a simulation is determined by the problem that simulation solves.

During the testing phase we adopted periodic boundary conditions, thus any particle leaving on one end of scene would re-emerge from the other end. As the lattice-Boltzmann method conserves momentum and no particles leave the system, the total momentum of the system will be conserved. This was one of the most important tests during the debug phase. An important consequence of the (local) conservation of momentum is that the part of the liquid without momentum $n_0^{\sigma(eq)}$ diffuses isotropically. Other tests include the conservation of total particle density.

Because the results of the simulation also have to be presented at an oral presentation the results of the simulation have to be output in such a form that it is possible to make a movie of the simulation. Since it is impossible to do the simulation in real time on a standard computer, we choose to simply create a BMP image of each time step. These images were then later combined into a video, using the tool Microsoft Movie Maker (which is a standard part of Windows XP). As the BMP image format is a very simple the actual implementation was straightforward.

2.4 Results

In the following we will demonstrate some example simulations we performed using our simulation. In all examples we use a simulation area of 200×200 sites on a hexagonal lattice with two fluids present. In all simulations we have used the following parameters[2] :

- mean collision times $\tau_0 = \tau_1 = 10$,
- masses $m^0 = m^1 = 1$,
- compressibility parameter $d_0 = 0.9$,
- fluid-fluid interaction strength $G_{\sigma\bar{\sigma}}$=-0.2 with $\sigma \neq \bar{\sigma}$,
- solid-fluid interaction strengths $G^0_\alpha = 0.05$ and $G^1_\alpha = -0.05$ with $\alpha = 1, \ldots, 6$ the lattice directions.

These parameters were obtained more or less by trial and error and do not resemble directly a particular physical fluid.

The first simulation we present is a demonstration that the simulation indeed preserves momentum and particle density. In this simulation we have an area of 200×200 sites with in the middle a square with a width of 10 sites. We surround the box with a liquid with a uniform density of $n_0 = 0.6$ over the entire area. Periodic boundary conditions are applied. The system is shown in figure 5. We test particle density N and momentum conservation **P** by doing 400 time steps and then printing out N and **P**. The results are shown in table 1, except from a small error caused by the accuracy of double precision floating point numbers, the simulation performs as expected.

[2]These parameters are defined in section 2.1.

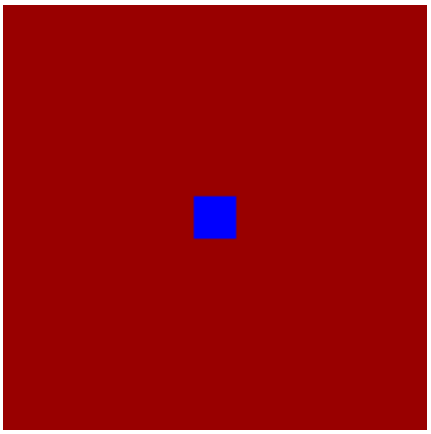

Figure 5: Test system of 200×200 sites and a uniform density of $n_0 = 0.6$ over all lattice sites. The square in the center has a width of 10 sites.

Time step	N	**P**
0	23759.9999999891224434	$(0.0000000000000000,\ 0.0000000000000000)$
400	23760.0000000000909495	$(0.0000000000000008,\ 0.0000000000000007)$
800	23760.0000000000327418	$(0.0000000000000037,\ 0.0000000000000016)$
1200	23760.0000000000291038	$(0.0000000000000041,\ 0.0000000000000025)$
1600	23759.9999999999854481	$(0.0000000000000059,\ 0.0000000000000011)$
2000	23760.0000000002401066	$(0.0000000000000030, -0.0000000000000013)$

Table 1: Total number of particles N and total momentum **P** for a system of 200×200 sites and initially at time step 0 an uniform density of $n_0 = 0.6$ over all lattice sites. In the center of the system there is a square of width 10 as shown in figure 5.

As a second example we demonstrate the following system. We have a sphere of radius 10 in a lattice of 200×200 sites surrounded by a liquid in equilibrium, see figure 6. To get the liquid into equilibrium we let the system run for 1600 time steps. We apply periodic boundary conditions at the left and right border, but we keep the bottom and top edge fixed. With $\mathbf{n}^0 = (0.6, 0, 0, 0, 0, 0.01, 0.01)$ at the top edge, where n_α is as it is described for the hexagonal lattice in section 2.2. Thus in this case we have a density $n_0 = 0.6$ of immobile particles, and densities $n_5 = 0.01$ going south-east and $n_6 = 0.01$ going south-east. These values have been chosen so that the total number of particles in the area remains constant and that there is not net momentum in the system.

When the liquid is in equilibrium we let a second fluid invade from the upper edge of the system, see figure 7. We do this applying the boundary condition $\mathbf{n}^1 = (0.3, 0, 0, 0, 0, 0.02, 0.02)$ to the upper edge. In figures 8 and 9 we show snapshots of how the simulation proceeds. Notice that because of the reflections of fluid against the sphere, the invading fluid curls around the sphere.

There most noticeable artifacts of the lattice-Boltzmann method are the seemingly unnatural sharp highlighting/shadowing effect at the top and bottom sides of the sphere. This is caused by particles colliding against the box. Because we have a stream going downwards a stream of particles collides against the top of the box and bounces back in the same direction. For this reason we see a shadowing effect below the sphere. Of course this effect is by itself a natural phenomenon, but because we use discrete momentum vectors we get unnatural looking sharp bands of higher/lower intensity.

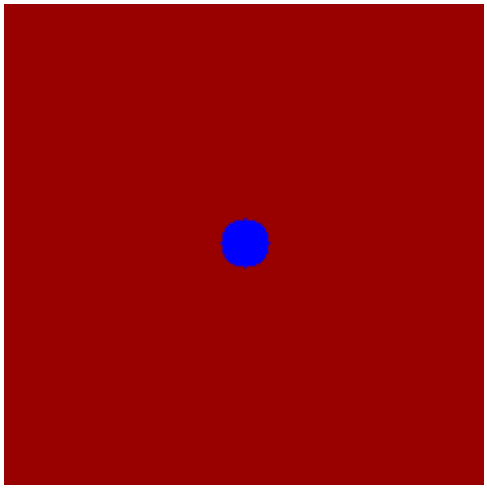

Figure 6: System of dimension 200 $\times$ 200 with a solid sphere of radius 10 surrounded by a liquid in equilibrium.

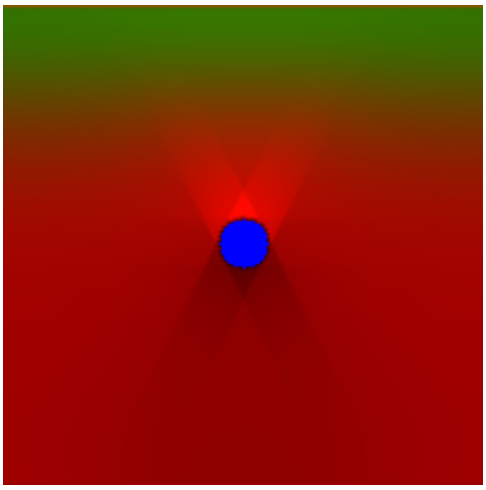

Figure 7: An invading fluid enters the system, the invading fluid has momentum has a net momentum of 0.04 in the vertical direction.

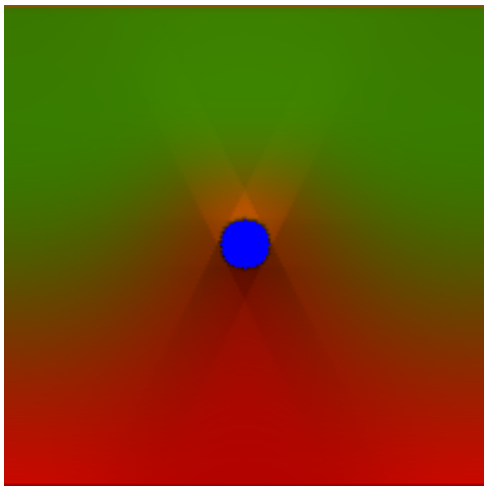

Figure 8: The invading fluid curls around the solid sphere.

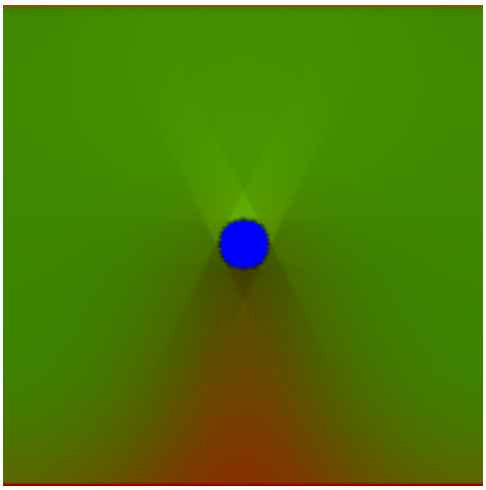

Figure 9: The invading fluid has now almost completely filled the system.

3 Coral reefs

Corals are in-fact animals, which are in the same family as anemones and jellyfish. A reef begins when coral larvae settle on the ocean floor. They then multiply and mature into so-called coral polyps, which are tiny anemone-like creatures with stinging tentacles. As the polyps grow, they extract calcium carbonate from the sea water, which they secrete as tiny skeletal cups. The large masses of corals in reefs are in fact colonies of polyps whose external skeletons are fused together. Not all corals build reefs. In fact, only the subspecies stony corals do this. This means than when we are talking about the visualization of coral growth, then we actual mean growth of the skeleton of the stony coral species.

3.1 Simulating coral growth

In 2001 Kaandorp and Sloot published a work on the simulation of coral growth [8], this method is a follow up of an earlier work [7]. We will now give a short description of their method. There are two effects that contribute to absorption of nutrients by corals: flow and diffusion. The relative importance of both effects can be quantized by means of the Péclet number Pe

$$Pe = \frac{\bar{\mu} l}{D}, \tag{11}$$

where $\bar{\mu}$ is the average flow velocity, l a characteristic length and D the diffusion constant of the substance. If Pe is small then the particle moves mainly by diffusion, on the other hand a large Péclet number means that the substance moves mainly by flow. As nutrients are large particles they are expected to diffuse slowly so that the effect of flow is dominant in coral growth. The fluid flow is computed using the lattice-Boltzmann method. The nutrients behave like a normal lattice-Boltzmann liquid, but also have a probability of Δ/ρ to remain in the same lattice site. Here Δ is a constant and ρ is the particle density. By varying Δ we can vary the rate of diffusion of the nutrient. When a nutrient collides with the coral it is absorbed.

The coral itself is represented by a triangulated polygon surface. The simulation starts with an initial seed object, this seed object then grows according to the amount of nutrient absorbed. But as natural coral has a certain maximum thickness also the method should ensure that local surface curvature does not exceed some preset value. This is done by measuring the local contact with the environment, point with a high local curvature have low contact with the environment. The local curvature is measured by measuring the curvature in a number of tangential directions (typically 6-8 direction). We call the minimum curvature low_norm_curv and the average curvature av_norm_curv. We define a function $h_2(low\_norm\_curv, av\_norm\_curv)$ given by

$$h_2(low\_norm\_curv, av\_norm\_curv) = low\_norm\_curv \cdot av\_norm\_curv, \tag{12}$$

to quantize the local contact with the environment. At each growth step a new layer is added to the coral, the length l between a vertex on the old coral surface and the new layer is

$$l = \begin{cases} s \cdot k \cdot h_2 & \text{for } k \cdot h_2 > tr \\ 0 & \text{for } k \cdot h_2 \leqslant tr \end{cases} \tag{13}$$

where tr is a threshold value and k is the amount of nutrient absorbed. In figure 10 we show some example growth stages taken from [8]. Notice stat that because of the curvature function h_2, branches split in two when they reach a certain maximum curvature. Finally to show the effectiveness of their approach we show in figure 11 some beautiful simulation results (also from Ref. [8]). Notice that as diffusion becomes more important the coral becomes fuller, as to maximize the surface area.

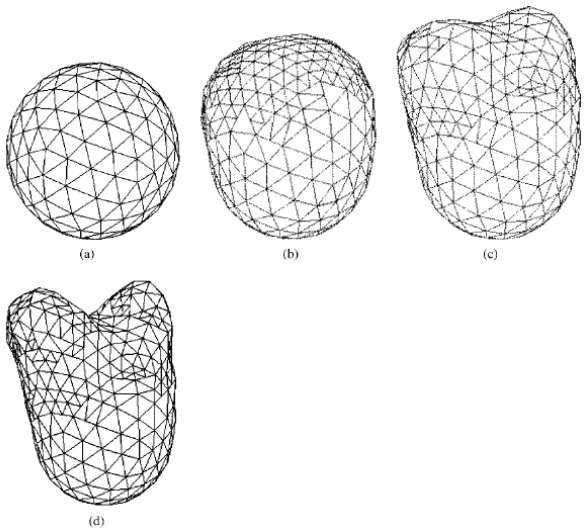

Figure 10: Four stages of the coral growth by Kaandorp and Sloot[8]. In (a) the initial seed is shown, notice that in (c) and (d) the branch splits because the maximum surface curvature was reached.

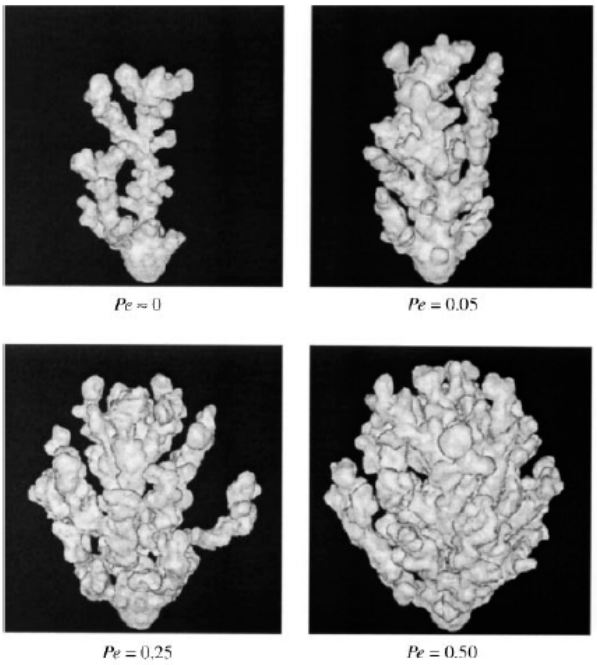

Figure 11: Example simulation results for various Péclet numbers Pe, taken from Ref. [8].

As a last remark we would like to mention the work by Merks, who defended a Ph.D. thesis on coral growth in 2003 [10] at the University of Amsterdam. Merks like Kaandorp and Sloot belongs to the Computational Science group of the University of Amsterdam and his work is based on the work by Kaandorp et al. discussed in this section. As part of his thesis Merks made a number of impressive movies of various coral growth simulations which are available on his website [10].

3.2 Our simulation

In previous section we described a method to simulate coral growth in three dimensions. Because we have only access to standard personal computer hardware we will have to limit our simulation to two dimensions. Thus we have to adapt the method of the previous section to two dimensions. One further difference is that in our simulation each vertex of the coral reef will be at a lattice site, whereas in the 3-dimensional method each vertex could be at an arbitrary coordinate.

We model the food and nutrient mixture as a two fluid mixture which behaves according to the lattice-Boltzmann method. Except that when a nutrient comes into contact with the coral it is absorbed whereas in the case of water the bounce-back rule is applied. The top edge of the simulation area serves as an inlet where a stream of nutrients and water enter the system. The coral seed is put at the bottom of the simulation area. The coral seed incidentally has the shape of a plus sign, where both arms of the '+' are of equal length. We use periodic boundary conditions at the left and the right edge, but not on the top and bottom edge. The fluid density at the top and bottom edge is kept constant. The simulation algorithm can be described as follows:

1. Simulate fluid flow with the lattice-Boltzmann method for Nt time steps. During the fluid flow we keep track of the number of nutrients absorbed by each coral site.
2. Apply growth rules to the coral.
3. Go to 1.

In step one we simply use our implementation of the lattice-Boltzmann method en count the total number of nutrients that are absorbed by the coral. The number of time steps per iteration is a compromise between the computation speed and the accuracy of the growth algorithm. We have used 400 time steps per iteration in our simulations, which we found after experimentation to be quite sufficient. In step two of the algorithm we use an adaptation of the growth rule we presented in the previous section. The growth of the coral depends on two parameters, the total food absorbed and the curvature at each lattice site. For each lattice site at the coral boundary we define the curvature h_2 as follows: Let V_i be the coral site we are currently considering. Imagine that we walk along the coral boundary in clockwise fashion, then the denote its two neighbours on the boundary as V_{i-1} and V_{i+1}. The curvature in the vertex V_i is now defined as the angle between the edges (V_{i-1}, V_i) and (V_i, V_{i+1}). On a hexagonal lattice this is extremely simple to calculate because as can be seen in figure 3 the angles between all succeeding unit vector e_a, e_{a+1} is the same(60°).

At each growth step we grow let every coral site grow l positions. The direction of the growth is the average direction from which the nutrients are absorbed. If k_n is the amount of nutrients absorbed at a particular coral site at time step n and d_n is the direction where the nutrients come from the average direction $\bar{d}$ is given by

$$\bar{d} = \frac{\sum_n k_n \cdot d_n}{\sum_n k_n} \tag{14}$$

Because we can only grow the lattice a discrete number of sites and our lattice is of limited size (200×200 sites) we have to carefully consider the growth rule. One consideration is that we should because of the limited lattice size not grow too many lattice sites at each growth step. But on the other hand we also need to ensure that there is a large variety in growth steps along the coral boundary. Because if some portions of the coral doesn't grow much faster then the rest of the coral we will never get a true coral structure. After some trial and error we found the following l most effective

$$l = \begin{cases} \text{int}(k \cdot h_2/H) & \text{for } k \cdot h_2/H < L \\ L & \text{for } k \cdot h_2/H \geqslant L \end{cases}, \tag{15}$$

where int depicts a conversion to integer, L is the maximum amount of sites the coral can grow and H is a parameter which normalizes the coral growth. The value of H depends on the amount of nutrients that are absorbed per time step by the entire coral.

3.3 Results

We will now demonstrate our coral growth algorithm. In the following we will use a simulation area of 200×200 sites on a hexagonal lattice. We used the following parameters[3] :

- mean collision times $\tau_0 = 10$ and $\tau_1 = 15$,
- masses $m^0 = m^1 = 1$,
- compressibility parameter $d_0 = 0.9$,
- fluid-fluid interaction strength $G_{\sigma\bar{\sigma}}$=-0.2 with $\sigma \neq \bar{\sigma}$,
- solid-fluid interaction strengths $G^0_\alpha = 0.05$ and $G^1_\alpha = -0.05$ with $\alpha = 1, \ldots, 6$ the lattice directions.

The best demonstration is perhaps the time evolution of the coral growth. We have put a number of example movies online [4] demonstrating coral growth for a number of fluid flows.

In figures 12-15 we demonstrate three example simulation runs. In each case we prepare a simulation area of 200×200 sites containing only stationary water with a density of $n_0 = 0.6$. Then we put a downward flow of both water and nutrients in the e_5(south east) and e_6(south west) directions at the top border of the simulation area. This we invoke as a boundary condition during the entire simulation. Before we start the coral growth algorithm we first let the system reach equilibrium. This we simply do by running the simulation for 1600 time steps without applying the coral growth rules. When the system has reached equilibrium we apply the coral growth algorithm as described in section 3.2. We apply $N_t = 400$ time steps for each coral growth step.

In figures 12 and 13 we put a flow of 0.01 of both water and nutrients in the e_5 and e_6 directions. In the second run (figures 14 and 15) we increase the flow to 0.02 and in the third run (figures 16 and 17) we put a flow of 0.06 in the e_5 and e_6 directions. Because of the fact that at higher fluid flows there will automatically be more nutrient absorption we have to scale H (which is defined in equation (15)) proportional to the fluid flow.

The results clearly show a coral like structure, in this sense the model is successful. However, the simulation does not capture all features of realistic coral growth. It is expected that when the current is small the coral will be bushy so that the coral maximizes the nutrients intake. This is shape is optimal for small currents because for these currents the diffusion mode of transport of the liquid becomes important. On the other hand when the current is large, the fluid flow will be the dominant mode of transport. In this case the coral is expected to be compact. Unfortunately this is feature is not present in our simulation. A possible explanation for this is that our simulation in two dimensions is to coarse. Each lattice site is only connected to six neighboring sites, this does not allow for truly realistic fluid dynamics. In the three dimensional D_3Q_{19} lattice for example each lattice site has 18 neighbours. To fully capture the effect of diffusion around the coral a three dimensional model is needed.

[3] Here the water is fluid 0 and the nutrient is fluid 1

[4] `http://members.home.nl/a.keimpema/ics/`

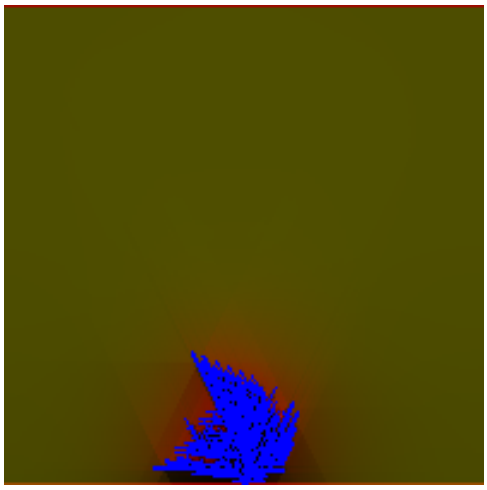

Figure 12: Coral after 15 coral growth steps using a downward flow of density 0.01 and growth speed normalisation $H = 2.5$.

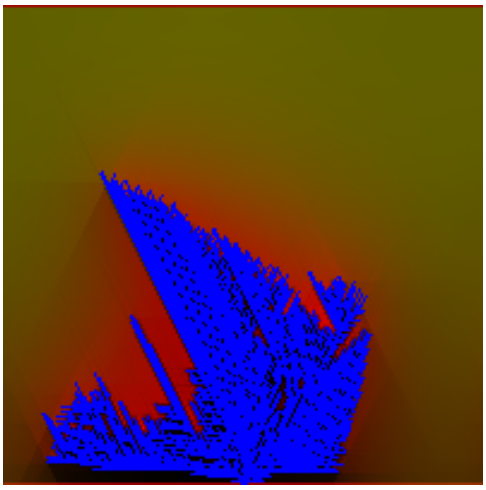

Figure 13: Coral after 30 coral growth steps using a downward flow of density 0.01 and growth speed normalisation $H = 2.5$.

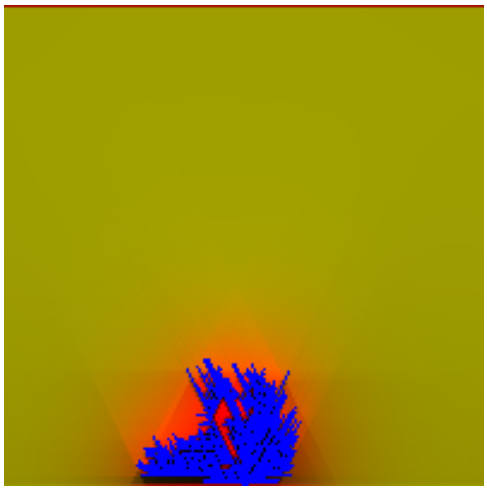

Figure 14: Coral after 15 coral growth steps using a downward flow of density 0.02 and growth speed normalisation $H = 5$.

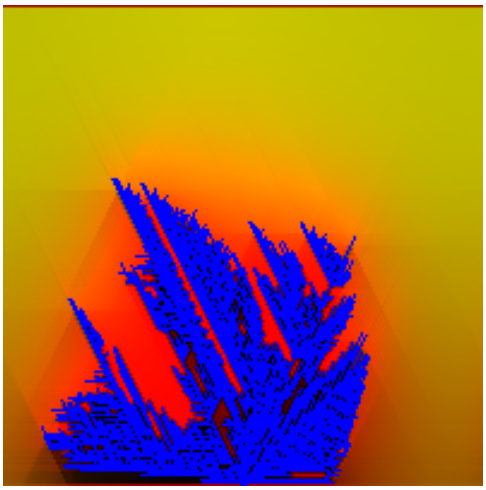

Figure 15: Coral after 30 coral growth steps using a downward flow of density 0.02 and growth speed normalisation $H = 5$.

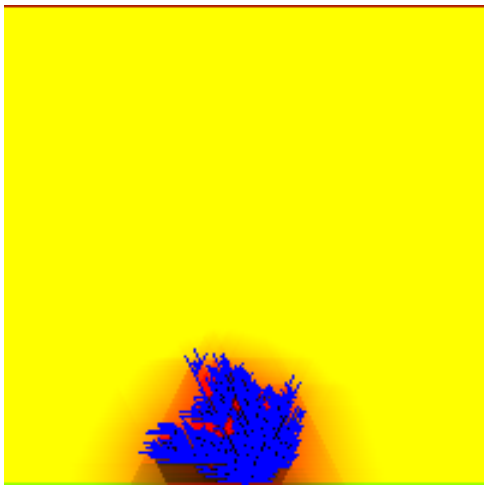

Figure 16: Coral after 15 coral growth steps using a downward flow of density 0.04 and growth speed normalisation H = 10.

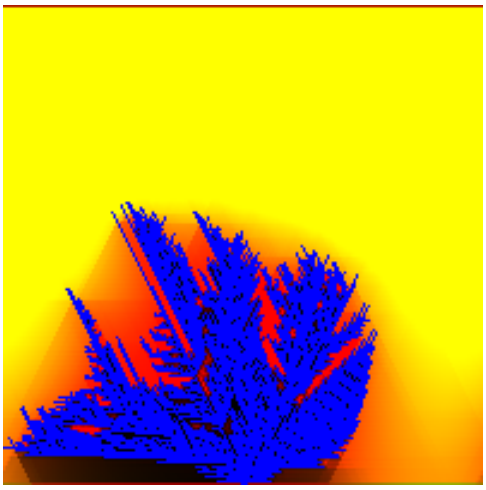

Figure 17: Coral after 30 coral growth steps using a downward flow of density 0.04 and growth speed normalisation H = 10.

4 Conclusion and discussion

In this work we presented a two dimensional simulation of the growth of coral reefs. Instrumental to the coral growth algorithm is a fluid dynamics simulation using the lattice-Boltzmann method. Our implementation is based on the work of Shan and Chen [6]. In section 2.4 we demonstrated that indeed our simulation was a realistic simulation of fluid flows. However, because of the limited resolution of our two dimensional model an unnatural looking highlighting effect took place around solid objects. This highlighting effect as can be clearly seen for example in figure 5. The effect is caused by the fact that fluid colliding with a solid object is scattered in only a small number of directions. In the case of figure 5 for example the fluid is only scattered in two directions. This is an limitation of the two dimensional lattice. Moving too a three dimensional lattice will fix this problem, but this will be to resource intensive for a personal computer.

Our coral growth algorithm is a two dimensional adaptation of the work of Kaandorp et. al [7][8]. The results are presented in section 3.3. It is clear that our simulation produces a coral-like structure but doesn't simulate all aspects of realistic coral growth. The most severe problem is that the shape of the coral is not influenced by the speed of the fluid flow. It is expected that when the flow speed is large the fluid becomes compact whereas at small speeds the coral becomes "bushy". This feature is not present in our simulation, a possible explanation is the limited connectivity of the two dimensional lattice. Each of the lattice site is only connected to six neighbouring lattice sites. Each lattice site is only connected to six other lattice sites. In a typical three dimensional case each lattice site is connected to 18 or more neighbours. Because of this very limited connectivity the fluid flow near the coral boundaries in our simulation is not sufficiently realistic. This is a fundamental problem of a two dimensional simulation and can only be resolved by moving to three dimensions.

5 Acknowledgments

We want to thank our teachers Henk Bekker and Michael Wilkinson for their inspiring course and for the interest in this paper.

References

[1] R.L. Burden and J.D. Faires, *Numerical Analysis-Seventh edition*, Brooks/Cole (2001).

[2] U. Frisch, B. Hasslacher and Y. Pomeau, *Frisch-Lattice-Gas Automata for the Navier-Stokes Equation*, Phys. Rev. Lett. **56**, 1505 (1986).

[3] McNamara and Zanetti, *Use of the Boltzmann Equation to simulate Lattice-Gas Automata*, Phys. Rev. Lett. **61**, 2332 (1988).

[4] N.S. Martys and H. Chen, *Simulation of multicomponent fluids in complex three-dimensional geometries by the lattice Boltzmann method*, Phys. Rev. E **53**, 743 (1996).

[5] H. Chen, S. Chen and W.H. Matthaeus, *Recovery of the Navier-Stokes equations using a lattice-gas Boltzmann method*, Phys. Rev. A **45**, 5339 (1992).

[6] X. Shan and H. Chen, *Lattice Boltzmann model for simulating flows with multiple phases and components*, Phys. Rev. E **47**, 1815 (1999).

[7] J.A. Kaandorp, C.P. Lowe, D. Frenkel, and P.M.A. Sloot, *Effect of Nutrient Diffusion and Flow on Coral Morphology*, Phys. Rev. Lett. **77**, 2328 (1996).

[8] J.A. Kaandorp and P.M.A. Sloot, *Morphological Models of Radiate Accretive Growth and the In6uence of Hydrodynamics*, J. theor. Biol. **209**, 257 (2001).

[9] R. Merks, A. Hoekstra, J. Kaandorp and P. Sloot, *Models of coral growth: spontaneous branching, compactification and the Laplacian growth assumption*, J. Theor. Biol. **224**, 153 (2003).

[10] R. Merks, *Branching Growth in Stony Corals: a modeling approach*, Ph.D. thesis University of Amsterdam, `http://staff.science.uva.nl/~roel/home.html`.