Computer Physics Communications 185 (2014) 841–846

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

Optimizing off-lattice Diffusion-Limited Aggregation

Kasper R. Kuijpers, Lilian de Martín *, J. Ruud van Ommen

Delft University of Technology, Department of Chemical Engineering, Product & Process Engineering, Julianalaan 136, 2628 BL Delft, The Netherlands

ARTICLE INFO

Article history:
Received 11 February 2013
Received in revised form
1 December 2013
Accepted 3 December 2013
Available online 13 December 2013

Keywords:
DLA
Agglomerates
Algorithm

ABSTRACT

We present a technique to improve the time scaling of Diffusion-Limited Aggregation simulations. The proposed method reduces the number of calculations by making an extensive use of the RAM memory to store information about the particles' positions and distances. We have simulated clusters up to $5 \cdot 10^6$ particles in 2D and up to $1 \cdot 10^6$ particles in 3D and compared the calculation times with previous algorithms proposed in the literature. Our method scales $t \propto N_p^{1.08}$, outperforming the current optimization techniques.

© 2013 Elsevier B.V. All rights reserved.

1. Introduction

Diffusion-Limited Aggregation (DLA) is a mechanism to describe the irreversible growth of fractal aggregates where diffusion is the dominant transport. It was proposed by Witten and Sander [1] and can be used to simulate the fractal structures of a large variety of processes, such as electrochemical deposition [2,3], viscous fingering [4,5] and nanoparticle agglomeration [6,7].

In a DLA simulation, the cluster is created starting with a static seed particle. Then, another particle is launched from a certain distance and diffuses through the space with Brownian motion. If the walker particle finds a particle in the middle of its trajectory, it will collide and stick to the particle. After sticking, the two particles form a static cluster. Then, a new particle is launched and the process is repeated. The simulation finishes when the cluster has the desired number of particles. The basic algorithm of the off-lattice DLA process is represented in Fig. 1; for more details see [8,9].

There are two reasons to be interested in modeling very large DLA agglomerates. One of them is practical: some agglomerates consist of hundreds of millions of particles, like those formed by nanoparticles in dense suspensions [10]. The other reason is of a more fundamental nature: as recently demonstrated by Menshutin [11], there exists a size effect in DLA agglomerates with less than $\sim 10^7$ particles. Even with the current computers such an amount of particles is difficult to reach in a reasonable amount of time, leading to the need of optimizing the DLA algorithms.

Optimization of the calculation times is thus one of the challenges in creating large DLA clusters. The speed of any simulation decreases with an increase in the number of particles; however, when dealing with fractals such as DLA clusters, this issue is further amplified. In a fractal cluster the volume of empty space inside the cluster grows faster than the volume occupied by the particles when new particles are added to the cluster. The density decreases with the size of the cluster $\rho \sim r^{(D_f - D_s)}$, where D_f is the fractal dimension and D_s is the space dimension [8,9]. There is more space for the particles to diffuse, which combined with the Brownian motion of the particles makes the simulation inherently slow and with a poor time scaling.

Kaufman et al. [12] created a DLA model for parallel computing in 1995, but a downside of this model was that it did not completely follow the DLA scaling behavior due to interference effects. In 2008, Alves et al. [13] reviewed some different optimization techniques used in off-lattice aggregate simulations. The combination of these optimization techniques leads to a time scaling of $t \propto N_p^{1.4}$ for two dimensional DLA up to one million particles. Braga and Ribeiro [14] proposed some further changes, and claim that their approach leads to even better time scaling.

In this paper, we propose a novel approach for DLA simulation that reduces the number of calculations by making an extensive use of the RAM memory – available in large amounts in current computers – to store information about the particles' positions and distances. The time scaling with this method is $t \propto N_p^{1.08}$, outperforming the current optimization techniques [13,14].

2. Collision in the DLA algorithm

As briefly introduced before, a DLA simulation starts with a static seed particle in the middle of the domain. Another particle is then launched from a distance denoted as entrance radius, and diffuses through the space with a random walk. If the walker

* Corresponding author. Tel.: +31 15 27 84753; fax: +31 15 27 88267.
E-mail address: L.DeMartinMonton@tudelft.nl (L. de Martín).

0010-4655/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.cpc.2013.12.003

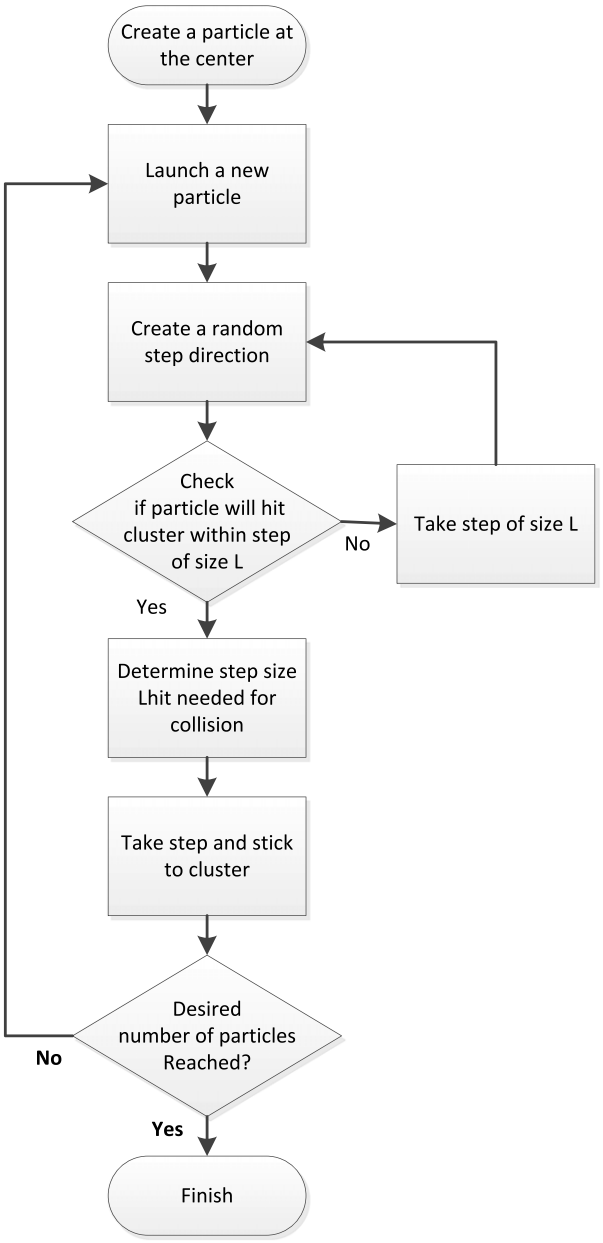

Fig. 1. Flowchart of the off-lattice DLA algorithm. L_{hit} is defined in Fig. 2.

collides with the seed particle it will stick to it, forming a static cluster. Then, another walker is launched from the entrance radius and the process is repeated. If the walker collides with any of the static particles that form the growing cluster it will stick to it, forming part of the cluster.

An important aspect of the simulation is how to define a collision between the walker and any of the particles that form the cluster. The concept is shown in Fig. 2 for a 2D simulation. The walker cannot take the normal step of size L, represented by the red line, because it collides with the particle in the cluster. Instead, it will take a step of size L_{hit}.

To know whether there will be collision between the walker and a particle in the cluster L_{hit} needs to be calculated. According to Fig. 2, a and b are

$$a = x_p - (x_0 + L_{\text{hit}} \cdot \cos(\alpha))$$
$$b = y_p - (y_0 + L_{\text{hit}} \cdot \sin(\alpha)), \tag{1}$$

where α is the angle of the future step. After collision, the distance between the particles is equal to the particle diameter d_p, so $d_p^2 =$

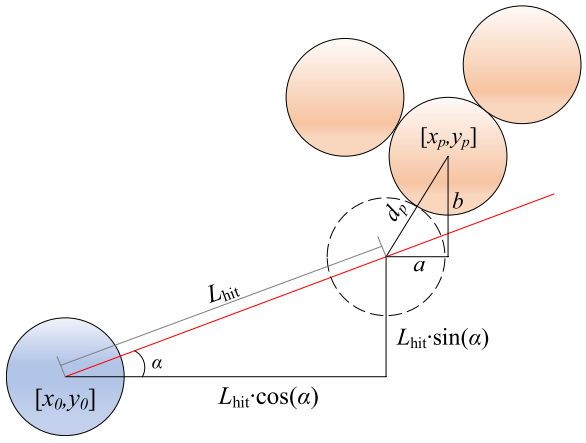

Fig. 2. Sketch of collision between particles in a 2D simulation. Red circles represent static particles forming a cluster. The blue circle represents a walker. The walker cannot take the last step with size L (red line) because it finds a particle in the middle of its trajectory. Instead, it will take a step with size L_{hit}, sticking to the particle. The final position of the walker is represented by a dashed line. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

$a^2 + b^2$, which combined with Eq. (1) gives the following quadratic equation:

$$AL_{\text{hit}}^2 + BL_{\text{hit}} + C = 0 \tag{2}$$

where

$$A = 1$$
$$B = 2(\cos(\alpha)(x_0 - x_p) + \sin(\alpha)(y_0 - y_p))$$
$$C = (x_p - x_0)^2 + (y_p - y_0)^2 - d_p^2. \tag{3}$$

Five hypothetical situations can be found when Eq. (2) is solved:

- No existing solution. The walker does not collide with the particle; the normal step with size L can be taken.
- $L_{\text{hit}} = 0$. The walker was already stuck to the particle, the program returns an error. This situation is prevented by the structure of the algorithm.
- $L_{\text{hit}} < 0$. The walker does not collide with the particle because it needs to move in the opposite direction to collide; the normal step with size L can be taken.
- $L_{\text{hit}} > L$. The walker does not collide with the particle because the normal step is too short; the normal step with size L can be taken.
- $0 < L_{\text{hit}} \leq L$. The walker collides with the particle; a step of size L_{hit} is taken.

If the walker can collide with two or more particles in the cluster, it will collide with the particle that gives the smallest L_{hit}.

3. Optimization of DLA algorithm

The size of the step is an important parameter in this type of simulation. If the step size is too small the walker will need many steps to reach the cluster, leading to a slow simulation. If the step size is too large as compared to the particle size, the motion of the walker is no longer Brownian and the cluster is not formed according to a diffusion limited mechanism. A way of speeding up a simulation without affecting the cluster structure is by using a variable step size. If the walker is far from the cluster the step size is large to approach the cluster quickly. If the walker is close to the cluster, the step size is small to simulate a Brownian motion.

The optimization presented in this paper maximizes the step size by estimating efficiently the distance between the walker and the cluster at each step. Opposite to Alves et al. [13], this

Download English Version:

https://daneshyari.com/en/article/10349753

Download Persian Version:

https://daneshyari.com/article/10349753

Daneshyari.com