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.

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

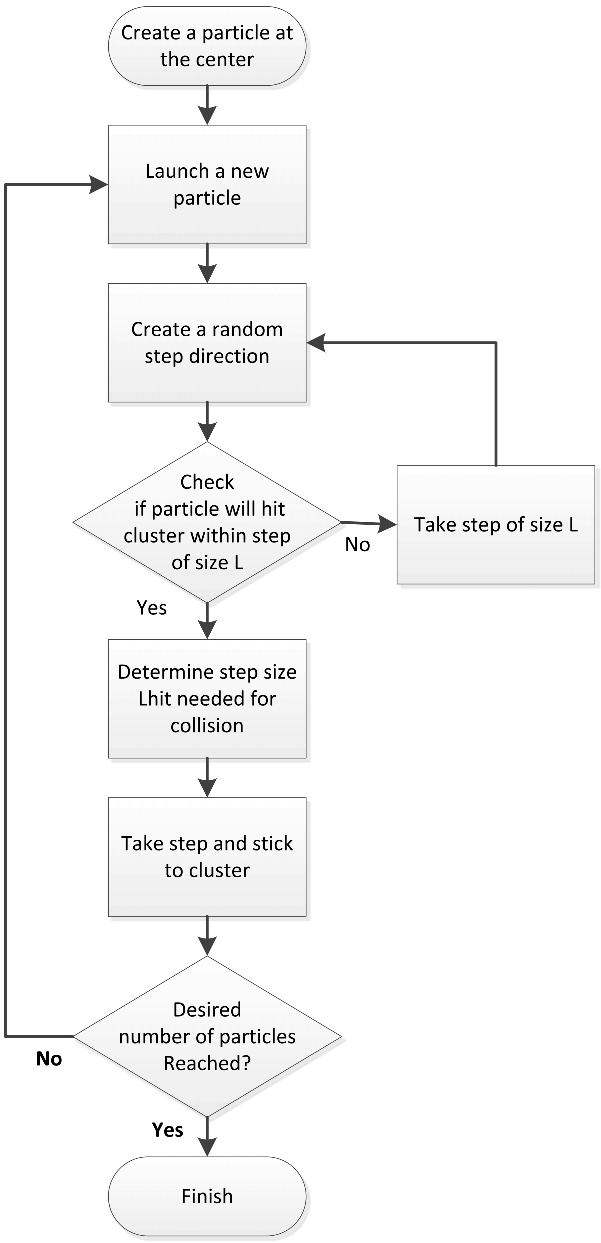

Create a particle at the center
Launch a new particle
Create a random step direction
Check if particle will hit cluster within step of size L
No
Take step of size L
Yes
Determine step size Lhit needed for collision
Take step and stick to cluster
Desired number of particles Reached?
No
Yes
Finish

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

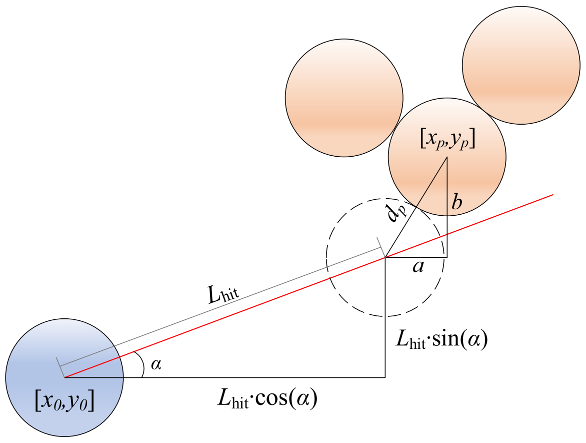

$[x_p,y_p]$
b
d_p
a
L_{hit}
$L_{\text{hit}}\cdot\sin(\alpha)$
α
$[x_0,y_0]$
$L_{\text{hit}}\cdot\cos(\alpha)$

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.)

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

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

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

$$AL_{\text{hit}}^2 + BL_{\text{hit}} + C = 0 \qquad (2)$$

where

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

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

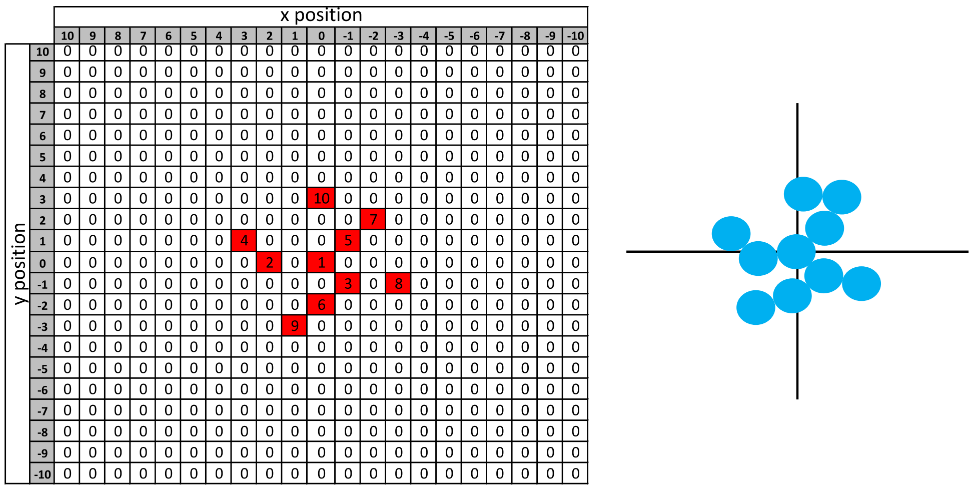
- No existing solution. The walker does not collide with the particle; the normal step with size L can be taken.
- $L_{\text{hit}} = 0$. The walker was already stuck to the particle, the program returns an error. This situation is prevented by the structure of the algorithm.
- $L_{\text{hit}} < 0$. The 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



		x position																				
		10	9	8	7	6	5	4	3	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10
y position	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	4	0	0	0	5	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	2	0	1	0	0	0	0	0	0	0	0	0	0
	-1	0	0	0	0	0	0	0	0	0	0	0	3	0	8	0	0	0	0	0	0	0
	-2	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0
	-3	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0
	-4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	-5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	-6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	-7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	-8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	-9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	-10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig. 3. On-Lattice cluster grid Υ with size $n = 21$. Each particle label is placed on a cell whose position is obtained rounding off the particle coordinates. On the right hand side is shown the real off-lattice agglomerate.

distance is estimated at each step with only one calculation, since the distances between the cluster and all the points in the space are stored in a matrix. This is the way our algorithm translates the problem from speed requirements to RAM memory requirements.

The algorithm uses the following elements.

Off-lattice array Ω

$N \times 3$ matrix in 3D simulations or a $N \times 2$ matrix in 2D simulations containing the *exact* positions (off-lattice) of the particles in the cluster. Each row represents the particle label, which is the chronological order in which the particle entered the cluster. The columns contain the floating point coordinates of the particles.

On-lattice cluster grid Υ

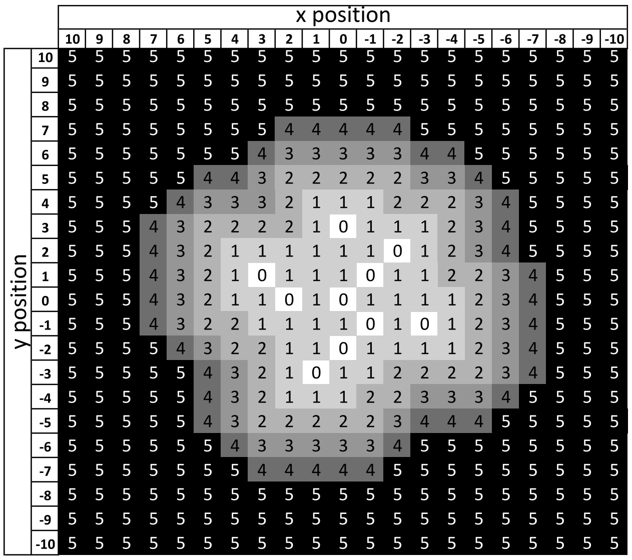
n^3 hyper-matrix in 3D or n^2 matrix in 2D containing the on-lattice version of the cluster, where n is the lattice size. This grid is similar to the matrix Z defined by Alves et al. [13] and it is obtained by mapping the particle positions on a square lattice by rounding off their real coordinates to the nearest integer $i = (i_x, i_y, i_z)$. Each particle is located in the cell $[(n+1)/2 - i_x, (n+1)/2 - i_y, (n+1)/2 - i_z]$ of Υ to get only positive grid positions. In this way, the seed particle with coordinates zero falls in the middle of the grid (Fig. 3). The value of each cell is the particle label; empty cells are zero.

To avoid overlapping between particles when their coordinates are rounded off, the radius of the particles ($r_p = 1$) and the size of the grid cells are the same. Then, the grid cells are 1×1 squares in 2D and $1 \times 1 \times 1$ cubes in 3D. Since the minimum distance between two particles ($2r_p$) is always larger than the distance between the corners of the cells ($\sqrt{2}$ in 2D and $\sqrt{3}$ in 3D), it is impossible that two particles fall in the same cell when their coordinates are rounded off.

On-lattice distance grid Ψ

n^3 hyper-matrix in 3D or n^2 matrix in 2D where each cell contains the rounded down distances (d_{wc}) between itself and the nearest occupied cell, as they appear in Υ. Occupied cells are labeled as zero. The distance grid Ψ of the cluster grid Υ shown in Fig. 3 is illustrated in Fig. 4.

The maximum distance inspected around the cluster is denoted as $D_{\max}$. Cells in which the distance between themselves and the nearest occupied cell is larger than or equal to $D_{\max}$ are labeled as $D_{\max}$, indicating that those cells are *at least* $D_{\max}$ units far from the cluster. $D_{\max}$ is a parameter that is set at the beginning of the simulation and is never recalculated.



		x position																				
		10	9	8	7	6	5	4	3	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10
y position	10	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
	9	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
	8	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
	7	5	5	5	5	5	5	5	5	4	4	4	4	4	5	5	5	5	5	5	5	5
	6	5	5	5	5	5	5	5	4	3	3	3	3	3	4	4	5	5	5	5	5	5
	5	5	5	5	5	5	4	4	3	2	2	2	2	2	3	3	4	5	5	5	5	5
	4	5	5	5	5	4	3	3	3	2	1	1	1	2	2	2	3	4	5	5	5	5
	3	5	5	5	4	3	2	2	2	2	1	0	1	1	1	2	3	4	5	5	5	5
	2	5	5	5	4	3	2	1	1	1	1	1	1	0	1	2	3	4	5	5	5	5
	1	5	5	5	4	3	2	1	0	1	1	1	0	1	1	2	2	3	4	5	5	5
	0	5	5	5	4	3	2	1	1	0	1	0	1	1	1	1	2	3	4	5	5	5
	-1	5	5	5	4	3	2	2	1	1	1	1	0	1	0	1	2	3	4	5	5	5
	-2	5	5	5	5	4	3	2	2	1	1	0	1	1	1	1	2	3	4	5	5	5
	-3	5	5	5	5	5	4	3	2	1	0	1	1	2	2	2	2	3	4	5	5	5
	-4	5	5	5	5	5	4	3	2	1	1	1	2	2	3	3	3	4	5	5	5	5
	-5	5	5	5	5	5	4	3	2	2	2	2	2	3	4	4	4	5	5	5	5	5
	-6	5	5	5	5	5	5	4	3	3	3	3	3	4	5	5	5	5	5	5	5	5
	-7	5	5	5	5	5	5	5	4	4	4	4	4	5	5	5	5	5	5	5	5	5
	-8	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
	-9	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
	-10	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5

Fig. 4. Distance grid Ψ for the cluster grid Υ shown in Fig. 3 for $D_{\max} = 5$. Cells have labels representing the smallest distance from themselves to the closest occupied cell. Occupied cells are zero.

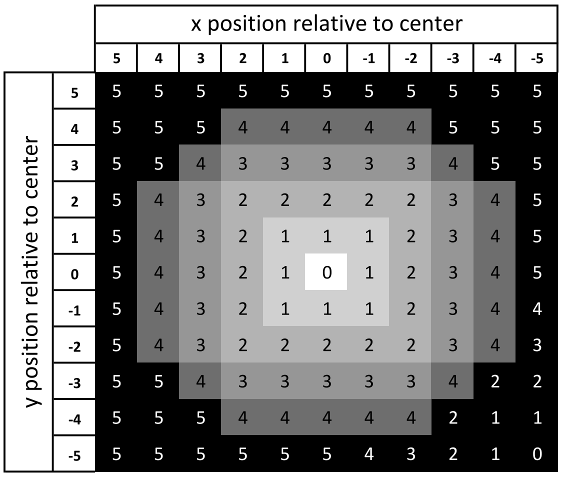
Vicinity grid Θ

$(2D_{\max} + 1)^3$ hyper-matrix in 3D or $(2D_{\max} + 1)^2$ matrix in 2D where the cell in the middle is set to 0 and each cell contains the rounded down distance between itself and the cell in the middle of the grid (Fig. 5). This grid is defined at the beginning of the simulation and is never recalculated.

Procedure

We define a minimum step size $L_{\min}$ for the walker right before collision with the cluster. $L_{\min}$ must be small enough to represent a Brownian motion. Accordingly, there is possibility of collision in the next step only if the exact off-lattice distance between the walker and the closest particle in the cluster is less than or equal to $2r_p + L_{\min}$. Thus, the condition for possible collision in the next step is $d_{wc} \leq 2r_p + L_{\min} + 1$. The one added is to make up for the rounded down distances in Ψ.

Let us take as an example a 2D simulation with $L_{\min} = 1$ and $r_p = 1$. The simulation starts with a seed particle placed at [0, 0] and all the cells of Ψ equal to $D_{\max}$. Then, the cells of Ψ inside a $(2D_{\max} + 1) \times (2D_{\max} + 1)$ square centered around the position of the seed particle are recalculated according to the



y position relative to center \ x position relative to center	5	4	3	2	1	0	-1	-2	-3	-4	-5
5	5	5	5	5	5	5	5	5	5	5	5
4	5	5	5	4	4	4	4	4	5	5	5
3	5	5	4	3	3	3	3	3	4	5	5
2	5	4	3	2	2	2	2	2	3	4	5
1	5	4	3	2	1	1	1	2	3	4	5
0	5	4	3	2	1	0	1	2	3	4	5
-1	5	4	3	2	1	1	1	2	3	4	4
-2	5	4	3	2	2	2	2	2	3	4	3
-3	5	5	4	3	3	3	3	3	4	2	2
-4	5	5	5	4	4	4	4	4	2	1	1
-5	5	5	5	5	5	5	4	3	2	1	0

Fig. 5. Vicinity grid Θ for $D_{\max} = 5$.

instruction min(Ψ, Θ). Now, Ψ is a matrix with a zero in the middle, representing the seed particle, that is surrounded by numbered cells representing increasing distances between the particle and the different points in the space (Fig. 6a).

A walker is launched from the entrance radius with a small step, say $L_{\min}$. After taking the first step, the walker is in the exact coordinates $[x, y]$. The distance d_{wc} between the walker and the closest particle in the cluster can be found in the cell $[(n+1)/2 - i_x, (n+1)/2 - i_y]$ of Ψ, where i is nearest integer to the coordinates.

Once d_{wc} is obtained, three situations can be found:

- $d_{wc} \le 4$: The walker is so close to the nearest particle in the cluster that there is possibility of collision at the next step. We then define a square (a cube in 3D) around the walker with size $2r_p + L_{\min} + 1$ and extract from Υ the labels of the particles that are inside this square, which are candidates for collision. After obtaining the candidates' labels, we extract their exact coordinates from Ω. Now, we have the current position of the walker $[x_0, y_0]$, the positions of all the candidates $[x_p, y_p]$ and the vector of the future step. If there is collision – each candidate is checked individually – L_{hit} (Fig. 2) is calculated with the method previously detailed, obtaining the final coordinates of the walker in the cluster $[x_f, y_f]$.

  Once the walker is added to the cluster, Ω and Υ have to be updated with the coordinates $[x_f, y_f]$ of the new particle. The procedure to recalculate Ψ is identical to the previous case, the only difference is that now the calculation is centered around the position of the new particle. An example is shown in Fig. 6b. Then, a new walker is launched from the entrance radius.
- $4 < d_{wc} < D_{\max}$. The walker is between d_{wc} and $d_{wc} + 1$ units far from the closest particle in the cluster. A step of size $d_{wc} - 4$ can be taken without the need of checking for collision.
- $d_{wc} = D_{\max}$. The walker is at least $D_{\max}$ units far from the closest particle in the cluster. A step of size $\max(d_{w0} - R_{\max} - 4, D_{\max} - 4)$ can be taken without the need of checking for collision. d_{w0} is the distance from the walker to the coordinate [0, 0] and $R_{\max}$ is the maximum radius of the cluster, equal to the largest distance between a particle in the cluster and [0, 0].

The maximizing of the step size with this method is the major improvement in the time scaling calculation. The larger $D_{\max}$ is, the larger the steps can be, making the code faster. The downside is that more cells in Ψ need to be recalculated for every added particle. This maximized step size is comparable to the internal step as implemented by Alves et al. [13]. A difference is that here only one cell of Ψ needs to be checked to calculate the step size.

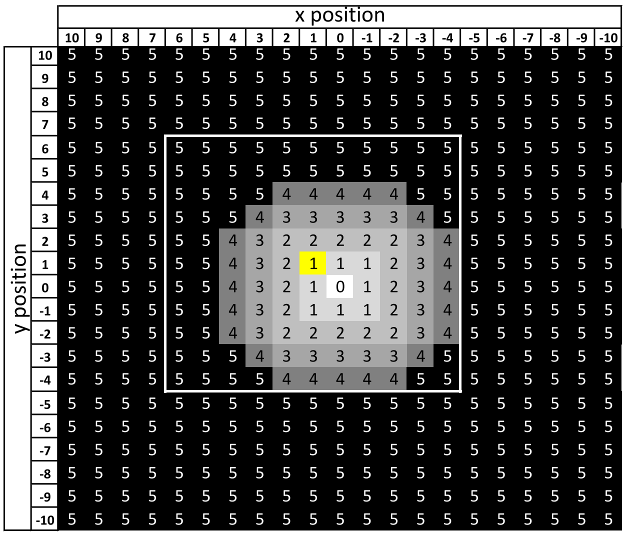
In addition to the optimizations mentioned above, a killing radius as described by Alves et al. [13] is implemented in our model. The killing radius is chosen to be $R_{\text{kill}} = 5R_{\max}$. If the moving particle moves farther away from [0, 0] than R_{kill}, it is removed from the simulation and a new particle is launched from the entrance radius.

An example of a 2D DLA agglomerate obtained with this method is shown in Fig. 7.

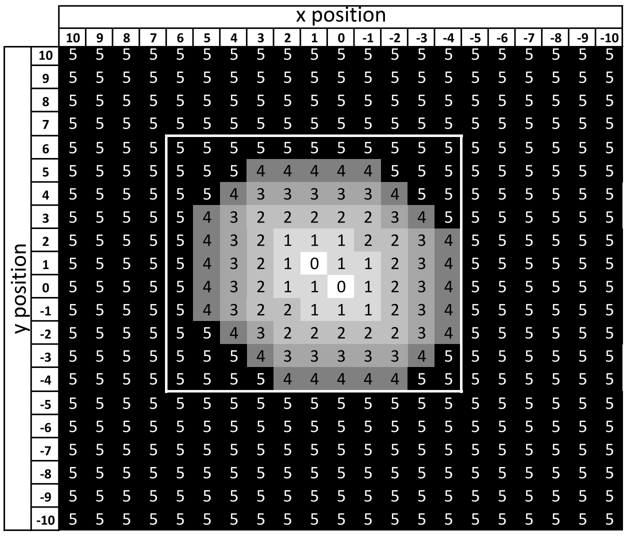
4. Numerical results

The model is implemented in C++, using Microsoft Visual studio 2010 on Windows 7 64-bit. The simulations were performed using an Intel Core i5-2400 3.1 GHz processor and 16 GB of DDR3 RAM memory. The model uses only one of the four CPU cores.

To estimate the time scaling of the algorithm we have simulated clusters up to $5 \cdot 10^6$ particles in 2D and up to $1 \cdot 10^6$ particles in 3D. The allocation and de-allocation of memory before and after the simulation is not taken into account in the simulation times.



y position \ x position	10	9	8	7	6	5	4	3	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10
10	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
9	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
8	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
7	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
6	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
4	5	5	5	5	5	5	5	5	4	4	4	4	4	5	5	5	5	5	5	5	5
3	5	5	5	5	5	5	5	4	3	3	3	3	3	4	5	5	5	5	5	5	5
2	5	5	5	5	5	5	4	3	2	2	2	2	2	3	4	5	5	5	5	5	5
1	5	5	5	5	5	5	4	3	2	1	1	1	2	3	4	5	5	5	5	5	5
0	5	5	5	5	5	5	4	3	2	1	0	1	2	3	4	5	5	5	5	5	5
-1	5	5	5	5	5	5	4	3	2	1	1	1	2	3	4	5	5	5	5	5	5
-2	5	5	5	5	5	5	4	3	2	2	2	2	2	3	4	5	5	5	5	5	5
-3	5	5	5	5	5	5	5	4	3	3	3	3	3	4	5	5	5	5	5	5	5
-4	5	5	5	5	5	5	5	5	4	4	4	4	4	5	5	5	5	5	5	5	5
-5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
-6	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
-7	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
-8	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
-9	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
-10	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5



y position \ x position	10	9	8	7	6	5	4	3	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10
10	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
9	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
8	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
7	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
6	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	4	4	4	4	4	5	5	5	5	5	5	5	5	5
4	5	5	5	5	5	5	4	3	3	3	3	3	4	5	5	5	5	5	5	5	5
3	5	5	5	5	5	4	3	2	2	2	2	2	3	4	5	5	5	5	5	5	5
2	5	5	5	5	5	4	3	2	1	1	1	2	2	3	4	5	5	5	5	5	5
1	5	5	5	5	5	4	3	2	1	0	1	1	2	3	4	5	5	5	5	5	5
0	5	5	5	5	5	4	3	2	1	1	0	1	2	3	4	5	5	5	5	5	5
-1	5	5	5	5	5	4	3	2	2	1	1	1	2	3	4	5	5	5	5	5	5
-2	5	5	5	5	5	5	4	3	2	2	2	2	2	3	4	5	5	5	5	5	5
-3	5	5	5	5	5	5	5	4	3	3	3	3	3	4	5	5	5	5	5	5	5
-4	5	5	5	5	5	5	5	5	4	4	4	4	4	5	5	5	5	5	5	5	5
-5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
-6	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
-7	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
-8	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
-9	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
-10	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5

Fig. 6. Recalculation of the distance grid Ψ with size $n = 21$ and $D_{\max} = 5$. (a) The first particle (white cell) is set at [0, 0] and a second particle is added on the yellow cell at [1, 1]. The thick border centered around the new particle will be recalculated by using the instruction min(Ψ, Θ). (b) Distance grid Ψ after recalculation. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 1
Comparison between the CPU times for 2D DLA obtained in the present work and those reported in [13,14] for different number of particles. Times are expressed in seconds.

N_p	$D_{\max} = 50$	$D_{\max} = 90$	$D_{\max} = 140$	Alves [13]	Braga [14]
1×10^3	0.093	0.17	0.41	0.10	3.7
2×10^3	0.20	0.36	0.81	0.50	18
5×10^3	0.50	0.89	2.0	1.5	92
1×10^4	1.0	1.8	4.0	2.4	175
2×10^4	2.1	3.6	8.1	5.3	832
5×10^4	5.6	9.3	21	14	3600
1×10^5	12	19	42	30	–
2×10^5	28	42	85	93	–
5×10^5	108	127	227	396	–
1×10^6	336	313	485	1560	–
2×10^6	995	777	1113	–	–
5×10^6	6226	3398	3593	–	–
$t \propto$	$N_p^{1.38}$	$N_p^{1.19}$	$N_p^{1.08}$	$N_p^{1.38}$	$N_p^{1.71}$

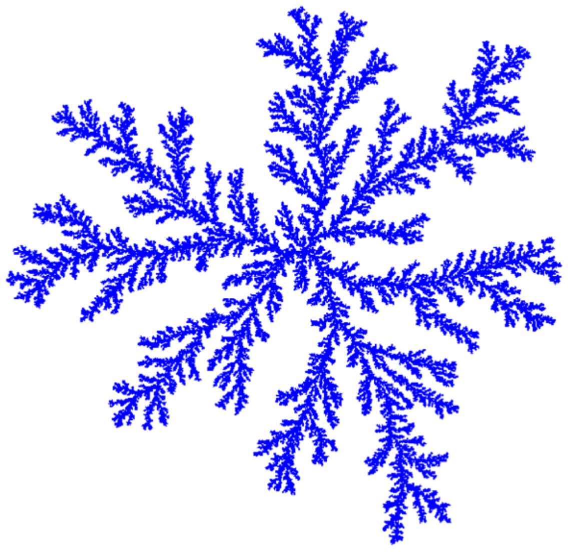


Fig. 7. 2D DLA aggregate with 10^6 particles.

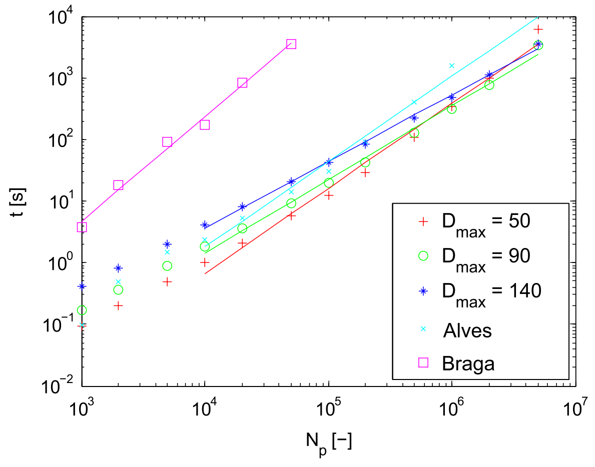


Fig. 8. CPU times of 2D DLA as function of the number of particles. The straight lines give a fit of the data.

Our computation times for 2D DLA have been compared to those for the most optimized calculations of Alves et al. [13] and Braga and Ribeiro [14]. The calculation times for $D_{\max} = 50$, 90, 140 are shown in Table 1 and Fig. 8. The CPU time t vs. the number of particles N_p can be fitted to a function

$$t = AN_p^{\alpha} \tag{4}$$

Table 2
CPU times for 3D DLA obtained in the present work. Times are expressed in seconds.

N_p	$D_{\max} = 10$	$D_{\max} = 15$	$D_{\max} = 20$
1×10^3	0.39	0.50	0.80
2×10^3	0.82	0.98	1.6
5×10^3	2.1	2.5	4.1
1×10^4	4.6	5.3	8.3
2×10^4	10	11	17
5×10^4	28	29	44
1×10^5	60	61	92
2×10^5	133	133	191
5×10^5	471	415	501
1×10^6	1344	1020	1049
$t \propto$	$N_p^{1.22}$	$N_p^{1.14}$	$N_p^{1.05}$

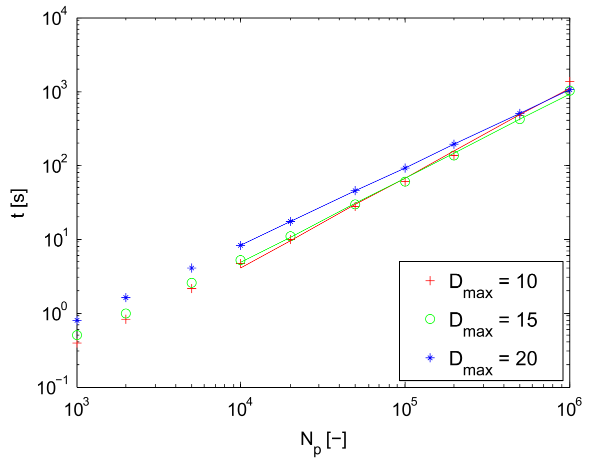


Fig. 9. CPU times of 3D DLA as function of the number of particles. The straight lines give a fit of the data.

where A is prefactor and α is the time scaling. Large α means high sensitivity of the CPU time to the number of particles, thus, poor time scaling. From the data, it is clear that larger $D_{\max}$ gives a lower α and a higher A. Our algorithm has a better time scaling than those presented by Alves et al. [13] and Braga and Ribeiro [14], especially at large $D_{\max}$.

For 3D DLA, we have not found literature data to compare our results with. The CPU times for $D_{\max} = 10, 15, 20$ are shown in Table 2 and Fig. 9. As in the 2D simulations, the time scaling improves with increasing $D_{\max}$.

The fractal dimension D_f of the simulated clusters has been obtained from the fit $N_p = CR_{\text{gyr}}^{D_f}$ [8,9,15], where N_p is the number of particles in the cluster, R_{gyr} is the gyration radius of the cluster and C is a prefactor [16]. Both C and D_f are fitting parameters. An example is shown in Fig. 10.

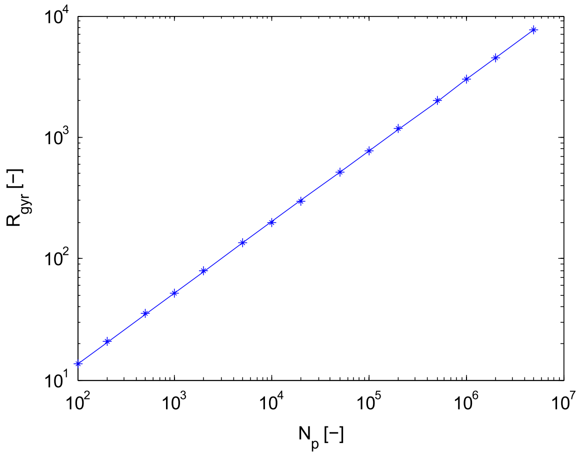


Fig. 10. Example of a fit for determining the fractal dimension based on the gyration radius of 2D DLA. The fractal dimension D_f is 1.71.

The fractal dimension found is $D_f = 1.709 \pm 0.005$ for 2D clusters and $D_f = 2.53 \pm 0.04$ for 3D clusters, in agreement with reported values [17]. The fractal dimensions are independent of $D_{\max}$ in the studied range.

From the results in 2D and 3D it can be concluded that increasing $D_{\max}$ improves the time scaling of the simulation since the particles can take larger internal steps before colliding with the cluster. However, the prefactor A (Eq. (4)) increases with increasing $D_{\max}$ because a larger number of cells have to be recalculated in the matrix Ψ. Therefore, the optimal size of $D_{\max}$ becomes larger the larger the desired cluster is. The downside of our approach is the memory usage by the grids Ψ and Υ. With 16 GB of memory the DLA model is limited to 10^7 particles in 2D and 10^6 particles in 3D. The memory usage can be reduced by increasing the size of the grid cells. If the size of the grid cells is $2r_p$ instead of r_p, the memory usage for the two grids would be decreased by a factor 4 in 2D and 8 in 3D. For Υ this brings in a minor complication, as a grid cell can contain multiple particles. The search for nearby particles will become faster because the number of grid cells that need to be checked is lowered by a factor 4 or 8. For Ψ this will lead to a lower precision, which will cause the step size taken to be a bit lower and will probably increase α (Eq. (4)) somewhat. However, A will be smaller because the number of cells that have to be recalculated is also lowered by a factor 4 or 8.

The optimizations described in this paper can be used in a similar way for ballistic aggregation and DLA with a sticking coefficient, sometimes referred to as reaction-limited aggregation. However, these models form agglomerates with higher fractal dimension, so there is less space within the structure for internal steps. This means that we expect less drastic improvements using the distance grid than in the case of DLA.

5. Conclusions

We have introduced a new approach for optimizing off-lattice Diffusion-Limited Aggregation by making extensive use of the RAM memory to reduce the number of calculations required. The CPU time scaling of the proposed approach is shown to be better than the time scaling of earlier approaches. Our approach is rather memory-intensive, but a way to reduce the memory usage has been discussed.

References

[1] T.A. Witten, L.M. Sander, Phys. Rev. Lett. 47 (1981) 1400–1403.
[2] R.M. Brady, R.C. Ball, Nature 309 (1984) 225–229.
[3] C. Cronemberger, L.C. Sampaio, A.P. Guimarães, P. Molho, Phys. Rev. E 81 (2010) 021403.
[4] J. Nittmann, G. Daccord, H.E. Stanley, Nature 314 (1985) 141–144.
[5] J. Zhang, J. Luo, Z. Liu, 1st International Conference on Consumer Electronics, Communications and Networks, CECNet, pp. 4044–4047.
[6] S.R. Forrest, T.A. Witten Jr., J. Phys. A 12 (1979) L109.
[7] R. Zhang, M. Hummelgard, H. Olin, Phys. Status Solidi. A 209 (2012) 519–523.
[8] T. Vičzek, Fractal Growth Phenomena, World Scientific Pub Co Inc., 1992.
[9] P. Meakin, Fractals, Scaling and Growth Far from Equilibrium, Cambridge University Press, 2011.
[10] C.H. Nam, R. Pfeffer, R.N. Dave, S. Sundaresan, AIChE J. 50 (2004) 1776–1785.
[11] A. Menshutin, Phys. Rev. Lett. 108 (2012) 015501.
[12] H. Kaufman, A. Vespignani, B.B. Mandelbrot, L. Woog, Phys. Rev. E 52 (1995) 5602–5609.
[13] S.G. Alves, S.C. Ferreira Jr., M.L. Martins, Braz. J. Phys. 38 (2008) 81–86.
[14] F. Braga, M. Ribeiro, Comput. Phys. Commun. 182 (2011) 1602–1605.
[15] S.K. Friedlander, Smoke, Dust, and Haze: Fundamentals of Aerosol Dynamics, second ed., Oxford University Press, 2000.
[16] C.M. Sorensen, G.C. Roberts, J. Colloid Interface Sci. 186 (1997) 447–452.
[17] S. Tolman, P. Meakin, Phys. Rev. A 40 (1989) 428–437.