ORNL/TM-10876

ornl

OAK RIDGE
NATIONAL
LABORATORY

MARTIN MARIETTA

SIMDLA: A FORTRAN Program Simulating Diffusion-Limited Aggregation

Lisa Anne Renker
D. G. Wilson

OAK RIDGE NATIONAL LABORATORY
CENTRAL RESEARCH LIBRARY
CIRCULATION SECTION
4500N ROOM 175
LIBRARY LOAN COPY
DO NOT TRANSFER TO ANOTHER PERSON
If you wish someone else to see this report, send in name with report and the library will arrange a loan.

OPERATED BY
MARTIN MARIETTA ENERGY SYSTEMS, INC.
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

Printed in the United States of America. Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road, Springfield, Virginia 22161
NTIS price codes—Printed Copy: A03, Microfiche A01

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

ORNL/TM-10876

Engineering Physics and Mathematics Division

Mathematical Sciences Section

SIMDLA: A FORTRAN Program Simulating Diffusion-Limited Aggregation

Lisa Anne Renker
Baldwin-Wallace College

D. G. Wilson

Date Published - August 1988

Research sponsored by the
Applied Mathematical Sciences Research Program
U.S. Department of Energy
Office of Energy Research

Prepared by the
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831
operated by
MARTIN MARIETTA ENERGY SYSTEMS, INC.
for the
U.S. DEPARTMENT OF ENERGY
under Contract No. DE-AC05-84OR21400

MARTIN MARIETTA ENERGY SYSTEMS LIBRARIES

3 4456 0281457 8

Table of Contents

Abstract 1

Introduction 1

A. Diffusion-Limited Aggregation 2

B. The Program 4

1. Comparison with the Witten-Sander Model 4
2. Input and Output Files 5
   2.1 Input File 5
   2.2 Output File 6
3. Implementation Details 9
   3.1 Data Structure 9
   3.2 Program Organization 10
   3.3 Choosing a Starting Point 13
   3.4 Taking a Step 14
   3.5 Attaching a Particle 14

C. Suggestions for Variations or Further Developments 15

Acknowledgments 16

Bibliography 16

SIMDLA: A FORTRAN PROGRAM SIMULATING DIFFUSION-LIMITED AGGREGATION

Lisa Anne Renker
Baldwin-Wallace College

D. G. Wilson
Mathematical Sciences Section
Engineering Physics and Mathematics Division
Oak Ridge National Laboratory

ABSTRACT

This report documents a FORTRAN program designed to simulate the diffusion-limited aggregation of particles. Particles are released one at a time from a growing internal boundary on a grid of possible sites and allowed to walk randomly until either they stick to the growing cluster or walk off the grid. The user may specify the grid dimension, the number of particles to be added to the cluster, the location and size of the seed cluster, and the possible directions (up to eight) in which particles are allowed to move. The output is a list of the occupied cells after all particles have been added to the cluster. A graphics post processor creates pictures from this list.

INTRODUCTION

The FORTRAN program described in this paper simulates diffusion-limited aggregation (DLA). DLA is thought to form dendritic-like fractals with a fractal dimension of about 1.7. This program was originally designed to experiment with growing aggregates on a grid of hexagonal cells to observe what patterns of growth would form. It has been modified to allow other cell shapes. The user supplies the size of the grid, the number of particles to be added to the cluster, the size of the seed, and the initially occupied seed cells. The program allows the particles to move in up to eight directions that the user specifies (specifying a certain six will result in a grid of hexagonal cells). Particles are released one at a time at a growing internal

boundary of the grid and allowed to walk randomly until either they move to a cell next to an occupied cell or they walk off the grid, at which time a new particle is released. Here, "walk off the grid" means to move out a specified distance beyond the growing internal boundary around the cluster. The output file is a list of occupied cells.

Section A is a summary of some articles on diffusion-limited aggregation and how DLA relates to fractals. Section B deals with the program itself beginning with a subsection that includes a description of the differences between our model and the Witten and Sander model. The second subsection of section B includes sample input and output files. Subsection 3 is further divided into subsections that describe the details of the program. A flowchart is given for the main program, and pseudocodes are given for the subroutine to choose a starting place for a particle and the subroutine to attach a particle to the cluster. The data structure used is explained and a diagram given. The method for taking a step is also described in subsection 3. The paper ends with some suggestions for variations or further development of the program.

A. DIFFUSION-LIMITED AGGREGATION

This paper is concerned with diffusion-limited aggregation (DLA). DLA is thought to produce fractal patterns that resemble dendrites. The motivation for this computer program was the study of dendritic growth, but the aggregation of particles into clusters has many other applications. A detailed literature search on this and similar subjects relating to phase change processes and fractals can be found in Renker and Wilson [8].

The program is based on the DLA model proposed by Witten and Sander [13,14]. Their model, in turn, is based on the Eden model [1]. The Eden model is a simple model for the growth of wispy-looking clusters of particles. It is a lattice model in which particles are added one at a time to random sites adjacent to occupied sites. The result of this aggregation process is a relatively compact cluster with density correlations that are independent of distance as the number of particles increases without bound.

Here we give a brief description of the model proposed by Witten and Sander. Initially a seed particle is located at the origin of a square lattice. Another particle is added at a random spot that is a large distance from the origin. This particle walks randomly on the grid. If the particle walks to a site that is adjacent to

the seed, then that particle is added to the cluster and remains in that spot. The other possibility is that the particle could step off the boundary of the lattice. In either case, the process begins again with a new particle. The process is repeated as many times as specified by the number of particles to be released.

Witten and Sander [13,14] examined the density correlation function to obtain more information about the particle distribution in their model. The density $\rho(r)$ is defined to be 1 for occupied sites and 0 for unoccupied sites. The density correlation function for an aggregate of N-particles in two dimensions is: $C(r)=N^{-1}\sum_{r'}\rho(r')\rho(r'+r)$ where the sum on r' is over all occupied sites and r is the step distance separating two sites averaged over all possible directions (r must be greater than a few lattice spacings, but much less than the size of the aggregate). Suppose, for example, we wanted the density correlation function at a distance of 4. Then for every particle of the aggregate, we would determine how many occupied particles were exactly 4 steps away from that particle in each possible direction. We average the totals obtained over all possible directions and divide by the total number of particles in the aggregate. The result would be C(4).

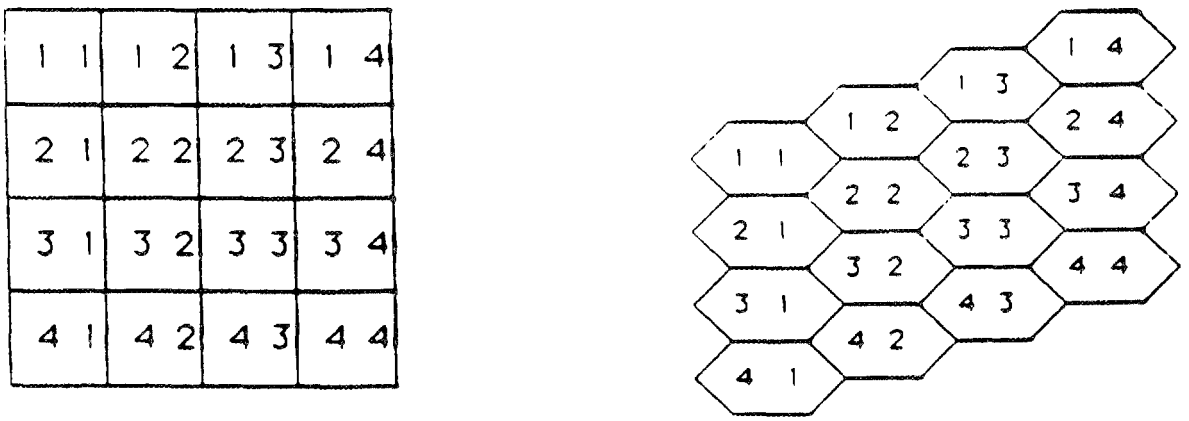
The density correlation function conforms to a power-law relationship: $C(r) \sim r^{-\alpha}$. This power-law form is consistent with a fractal dimension D that characterizes the object by $D = d - \alpha$ (d is the Euclidean dimensionality of the cluster). The radius of gyration (R_g) can also be used to determine the fractal dimension. R_g has a power-law dependence on the number of particles for large N, i.e., $R_g \sim N^{\beta}$. The fractal dimension is then $D = 1/\beta$ [13].

The objects produced by this DLA model were thought to be fractal when the correlations between two particle positions were examined closely by Witten and Sander [14], and Meakin [4,5]. The correlations were found to be typical of a scale-invariant object, i.e., the patterns grown had no natural length scale. The DLA growth formations are tenuous and wispy and do not fill a finite fraction of space. They have large holes owing to the screening effect because the fingers formed make it difficult for particles to walk into the gap between the fingers, so the particles are more likely to stick to the tips of the fingers. According to Sander [9], these holes must be on the order of the size of the object itself.

B. THE PROGRAM

1. COMPARISON WITH THE WITTEN-SANDER MODEL

This program implements a variation of the model proposed by Witten and Sander. One difference is that the program was originally designed with grid cells shaped like hexagons. This allowed six possible directions in which to go at each step of the random walk. The program was later modified in order to incorporate other grid cell shapes and structures so that now there are eight possible directions in which moves can be made. Figure 1 demonstrates the numbering scheme for two different grids. The grid cell shape is determined by directions specified in the input file.



a.) A square close packed grid

b.) A hexagonally close packed grid

Figure 1.

Another difference is that we have incorporated Meakin's [4] idea of having an internal boundary that grows as the cluster size grows. This is done because as the grid size increases, the probability that a

particle released at the edge will make it to the cluster near the center of the grid decreases greatly. A large number of particles must be used to ensure that several of them stick. This results in much time wasted because most of the particles walk off the grid. This tendency is reduced by setting up a second boundary a few lattice spaces larger than the current cluster size. The particle then begins at that internal boundary and is considered finished if it oversteps a somewhat larger boundary. The radius of the internal boundary is increased as the size of the cluster increases, until the internal boundary reaches the edge of the grid. This process could be thought of as an accelerated version of the original model because it is equivalent to assuming that some number of particles would have reached that internally bounded section of the grid after a certain number of particles were released.

In this program, a particle is started at a random spot on the edge of the internal boundary and allowed to perform a random walk on the grid until either it visits a cell adjacent to one that is already occupied or it walks off a boundary with a radius that is ten units larger than the boundary it started on. When one of those two possibilities occurs, a new particle is started at the internal boundary. The size of the internal boundary increases as the radius of the cluster increases. The program is terminated when the specified number of particles has been added to the cluster.

2. INPUT AND OUTPUT FILES

2.1 Input File

The input file contains the desired size of the grid, the number of particles to be added to the cluster, how many cells will initially be occupied, the locations of these initially occupied cells, and a random number to be used as a seed for the random number generator. The desired size of the grid is the dimension of one edge of the grid. The program is set up so that a particle may move in any of up to eight directions. The input file specifies in how many directions the particle may move and allows the user to choose those directions. An example of an input file follows. This example is the data used to simulate a grid of hexagonally close packed locations with a seed of five cells. All the entries are left justified and are explained on the right.

2500	size of the grid	
750	number of particles to be added to the cluster	
5	number of seed particles	
1250 1250	seed locations	
1251 1250		
1250 1251		
1249 1249		
1251 1251		
331740391	random number generator seed	
6	number of directions to be checked	
T	h-up, s-up	*The direction descriptions are
F	h-x, s-up,right	*given for both the hexagonal
T	h-up,right, s-right	*and the square grid.
T	h-down,right, s-down,right	
T	h-down, s-down	
F	h-x, s-down,left	
T	h-down,left, s-left	
T	h-up,left, s-up,left	

* h=hexagon, s=square, x=does not apply

2.2 Output File

The output file contains as its first entry, the dimension of one side of the grid. This is written to the output file as part of the initialization process. After that, the output consists of a list of the occupied cells in order of their addition to the aggregate beginning with the seed cells. Each cell is written to the file as it is attached to the cluster by the *stick* subroutine. The end of file is signalled to the graphics program by a pair of negative numbers that are written by the last executable statement of the program. A series of pictures obtained using the graphics program during two simulations are shown in Figures 2 and 3. The pictures in Figure 2 show the growth of a small aggregate on a grid of hexagonally close packed locations of dimension 40 x 40. The pictures in Figure 3 show the growth of an aggregate on a square grid of dimension 3000 x 3000. A small sample output file obtained from another simulation follows.

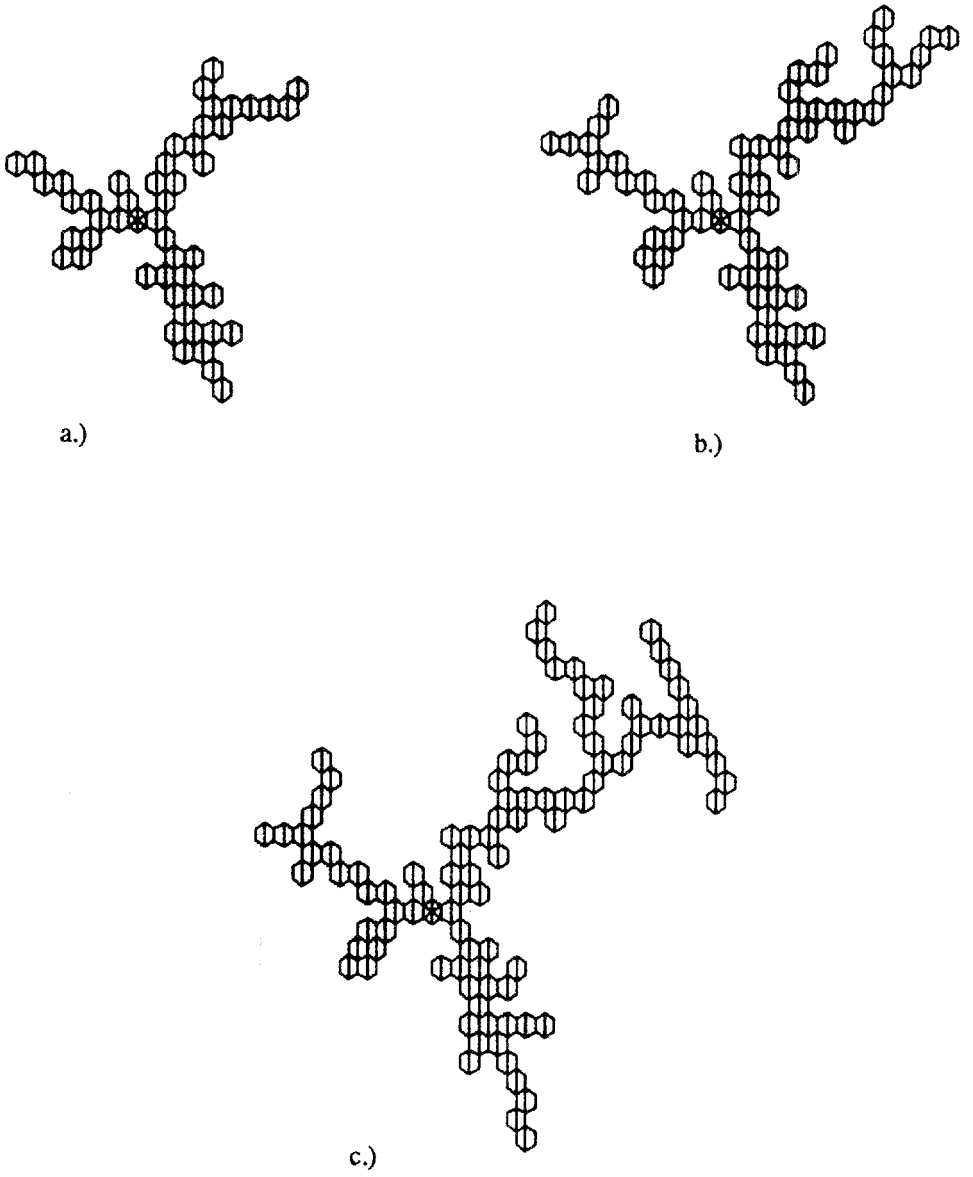


Figure 2. The growth of an aggregate on a hexagonally close packed grid with a.) 50 particles stuck, b.) 70 particles stuck, and c.) 100 particles stuck.

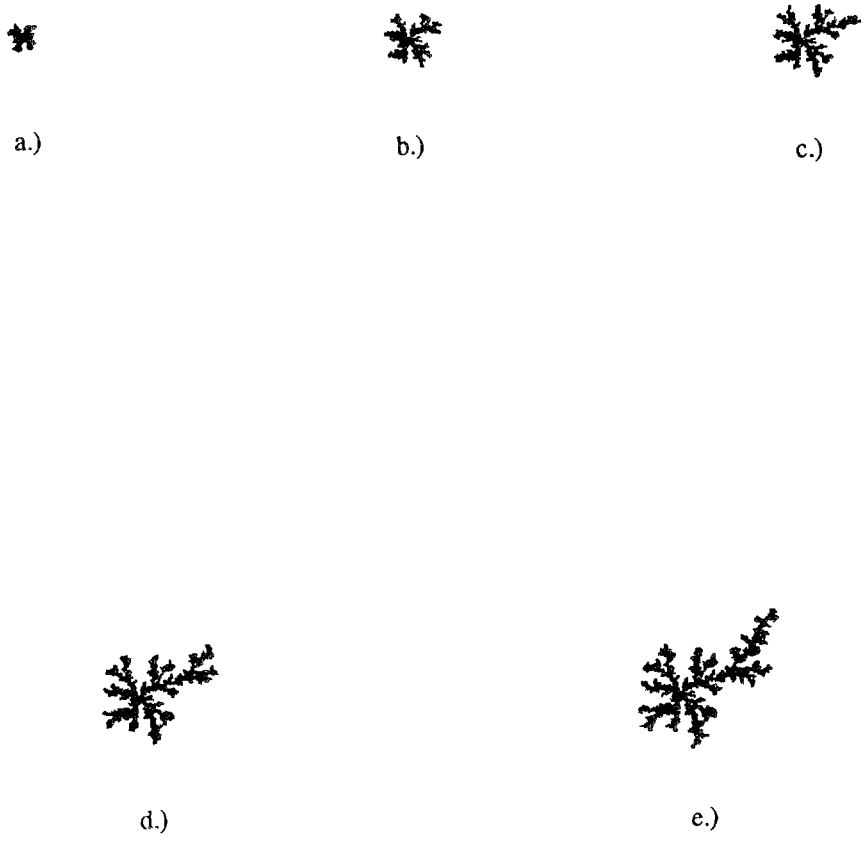


Figure 3. The growth of an aggregate on a square close packed grid with a.) 1000 particles stuck, b.) 3000 particles stuck, c.) 5000 particles stuck, d.) 8000 particles stuck, and e.) 10000 particles stuck.

```
2500
1250 1250
1251 1250
1250 1251
1249 1249
1251 1251
1248 1248
1252 1250
1253 1250
1249 1251
1249 1252
-1 -1
```

3. IMPLEMENTATION DETAILS

3.1 Data Structure

The data structure used is a set of four one-dimensional arrays that form a set of linked lists, one for each row of the grid. This data structure is used to avoid having to store a great many zeroes corresponding to unoccupied cells. The dimension of the array, *irow*, is the size of one edge of the grid itself. It has an entry of zero or an integer for each row on the grid. The zero signifies that none of the cells in that row are occupied. The integer signifies that there is at least one cell occupied and identifies which entry it is in array *ientry*.

The dimension of the arrays *ientry* and *jentry* (and *iptr*) is the number of particles to be added to the cluster. *Ientry* and *jentry* are the first and second index of an occupied cell and *ipir* is the index of the entry in *ientry* (and *jentry*) where the coordinates of the next occupied cell in that row are stored (in ascending order). A particle is added to the end of these arrays when it has stuck to the cluster. *Next* is the pointer to the next available position where an occupied cell can be added to the list.

This data structure saves on storage, but trades that savings for more computation time. Suppose the dimension of a grid is 3000 x 3000 and the number of particles to be added to the cluster is 40,000. Then having an array the size of the grid itself would result in a storage of 9,000,000 cells, and only a small percentage (less than 1%) of those cells would be occupied. 9,000,000 exceeds the available storage on our computer. Using the data structure just described, only 40,000 cells of storage are needed. Locating a specific cell in the list of occupied cells requires one computation in the first case. However, locating a

cell in the second case could require at most 3000 (the grid size) computations. This worst case occurs only when every cell in a row is occupied and we are trying to locate the last cell in that row. Even though there may be a large increase in the computation time required, the linked list data structure is better because it allows for a larger grid size.

A pictorial representation of the data structure is shown in Figure 4. This diagram represents one stage in a simulation with a grid size of 10 x 10 with 20 particles to be added to the cluster. Eleven particles have already stuck to the cluster. In this diagram, (5,5) is the seed particle and therefore the first cell added to the list of occupied cells. To determine whether a certain cell was occupied in row 6, for example, first look at the sixth entry in *irow*. This has a value of 7. This 7 tells us to look in the seventh entries of the other three arrays for the first cell in row 6 that is occupied. In this case, its coordinates are (6,3). *Iptr*(7) is five. This is the pointer to the next set of entries in the linked list. This five means that the next entry in row 6 of the grid is located in the fifth entries of the arrays. The coordinates of this entry are (6,4). The index of the next occupied cell in row 6 is in *iptr*(5). Therefore, the next occupied cell in row 6 is (6,5). The 0 in *iptr*(3) signifies that the end of the list for row 6 has been reached.

3.2 Program Organization

The main processing of the program is done in a subroutine named *controller*. A flow chart for this subroutine is shown in Figure 5. A starting point for the random walk is chosen. If the starting point is already occupied, a new particle is introduced and the process of choosing a starting point begins again. If the starting point is not occupied, then the neighbors are checked for occupation. (The only neighbors checked are those that are in the user specified allowable directions list.) Subroutine *checknghbrs* returns a value of true or false in the form of the variable *found* to the *controller* subroutine. If a neighbor was found to be occupied then the *stick* routine is called and the new cell is added to the list of occupied cells. If none of the neighbors were occupied, then the *mkmove* subroutine is called.

	irow	next		ientry	jentry	iptr
1	0	12	1	5	5	2
2	0		2	5	6	0
3	10		3	6	5	0
4	9		4	4	6	0
5	8		5	6	4	3
6	7		6	7	6	0
7	11		7	6	3	5
8	0		8	5	4	1
9	0		9	4	3	4
10	0		10	3	3	0
			11	7	4	6
			12			
			13			
			14			
			15			
			16			
			17			
			18			
			19			
			20			

Figure 4. The data structure is a set of arrays that form a linked list.

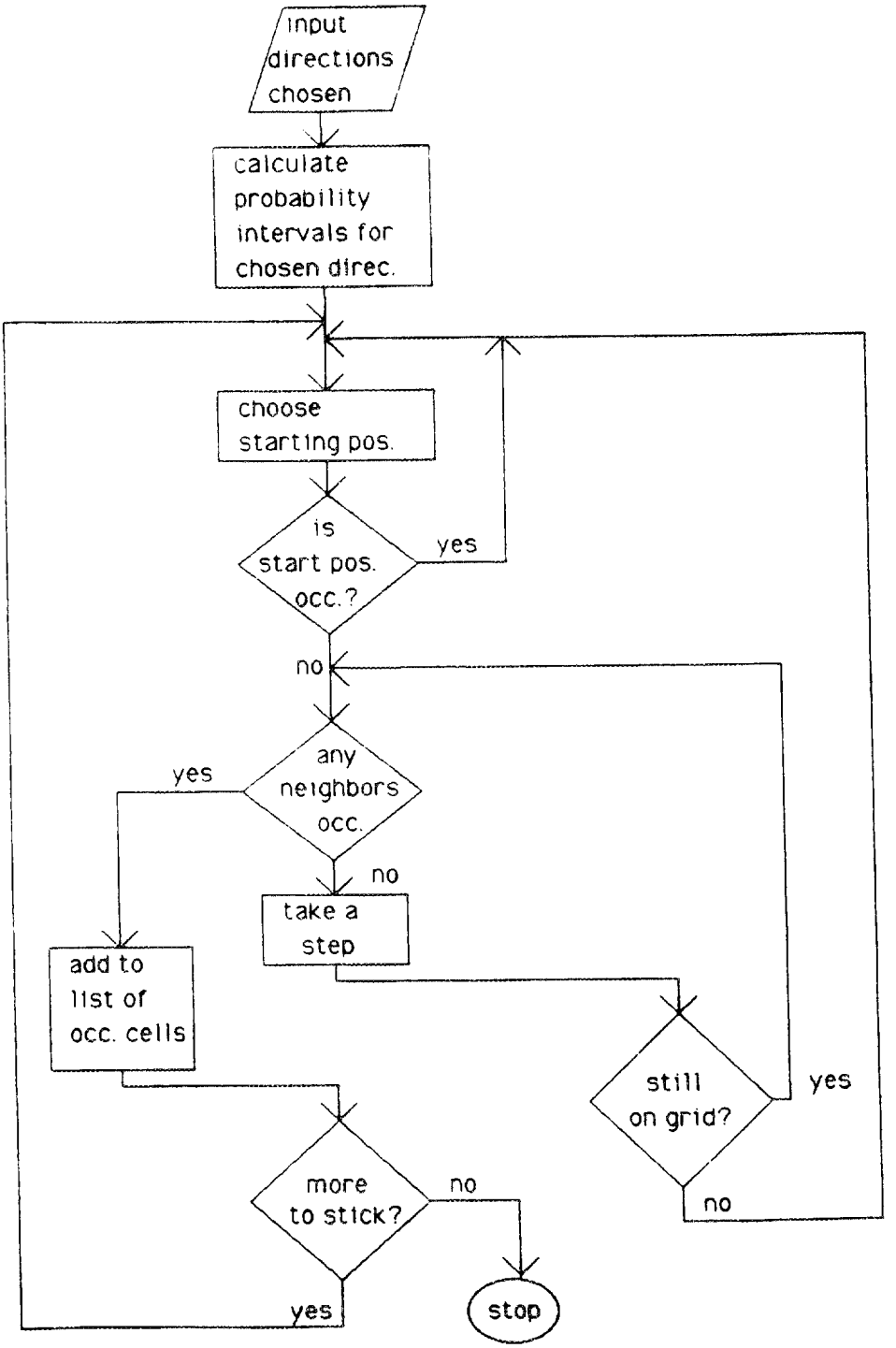


Figure 5. The flow chart for the main loop of the program.

3.3 Choosing a Starting Point

A starting position on the edge of the grid is chosen as follows. Three random numbers in (0,1) are generated. Using the square grid as a simple example, the first random number decides which set of sides the particle will start on, the top and bottom rows or the two sides. This choice determines which index will be set, the row index or the column index. The second random number determines which row or which side out of the two that the starting position will be on. Once the starting edge has been picked, the exact cell must be chosen. The remaining index is equal to the integer portion of two times the product of the third random number and the current radius of the cluster, plus a constant to ensure that the result is in the desired range. This method ensures that all starting locations on the internal boundary are equally likely. The logic of this method is shown in the following pseudocode. This has been simplified by assuming that the particle starts on the edge of the grid rather than on an internal boundary within the grid.

```
 choose three random numbers
    hold=int(3rd number * size of the grid) + 1
    if 1st number < .5
  j index=hold
  if 2nd number < .5
    i index=1
  else
    i index=size of the grid
else
  i index=hold
  if 2nd number < .5
    j index=1
  else
    j index= size of the grid
```

The method used for determining the boundary on a grid of hexagonally close packed locations is similar to the above method, but slightly more complicated.

3.4 Taking a Step

The array named *prob* contains the values of the intervals used in moving the particle. Using the input file given earlier, the intervals of probabilities would be divided into sixths. For example, *prob*(1)=1/6, *prob*(3)=2/6, *prob*(4)=3/6, *prob*(5)=4/6, *prob*(7)=5/6, and *prob*(8)=6/6 because the respective entries in the input file had the value of true. The other two *prob* entries (2 and 6), corresponding to false values in the input file, would be set equal to the value of the previous *prob* entry, so *prob*(2)=1/6 and *prob*(6)=4/6. The reason for this will be seen in the explanation of the *mkmove* subroutine.

The *mkmove* subroutine chooses a random number between zero and one. Then the eight intervals ($0 \rightarrow prob(1), prob(1) \rightarrow prob(2)$, etc.) are checked. The particle moves in the appropriate direction depending on which interval the random number falls in. The reason for setting the *prob* entries that correspond to a value of false in the input file equal to the preceding *prob* entry is that the random number cannot fall in that half closed, half open interval, thus, the particle cannot move in that direction. If, for example, the random number fell between 1/6 and 2/6, the particle should move in the second allowable direction. Setting *prob*(2) equal to *prob*(1) ensures that the direction associated with the $prob(1) \rightarrow prob(2)$ interval, [1/6,1/6), will not be chosen because that interval is empty.

Mkmove returns a value of "true" or "false" in the form of the variable *again*. This variable is set to true if the particle has not moved off the specified boundary edge. If *again* is true, the process of checking neighbors and moving that particle begins again. If *again* is false, that particle is finished and a new particle is released, starting the entire loop again. This loop is executed until the desired number of particles have been added to the cluster.

3.5 Attaching a Particle

The subroutine for adding a particle to the list of occupied cells is called after the particle has moved to a cell adjacent to an occupied cell. This involves updating the appropriate linked list and switching a few pointers. The pseudo-code follows. (Refer to the diagram in section 3.1 for the data structure.) *Spot* and *hold* are pointers.

```
      spot=irow(i index)
      hold=spot
          enter the current i and j indices in the next available position
      if spot = 0        (*no entries in that row*)
        irow(i index)=next
        iptr(next)=0
      else
200     if jentry(spot)>j index  (*attach to beginning of row list*)
        iptr(next)=spot
        if hold=spot  (*attach to beginning of occ. list*)
          irow(i index)=next
        else
          iptr(hold)=next
      else                  (*keep looking for place in list*)
        hold=spot
        spot=iptr(spot)
        goto 200
```

C. SUGGESTIONS FOR VARIATIONS OR FURTHER DEVELOPMENTS

One interesting variation of the program would result from changing the sticking probabilities. Particles could be given a probability of sticking each time they hit the cluster and thus they might bounce around for a while. Also, the program could be modified to make the probability of sticking on the tips lower and increase the probability of sticking in places where many neighbors are occupied. That is, the sticking probability would be smaller or larger on the perimeter sites depending on whether there were fewer or more neighbors occupied.

Another possibility is to model cluster aggregation. More than one particle could be released at a time and if that particle met another one, they would join. Clusters could also be released and allowed to attach to other clusters or single particles. (This model is discussed by Meakin [5].)

One more area of potential interest is to expand the model to three dimensions. Instead of having a hexagon shaped grid cell, we could have a three dimensional figure with twelve neighbors instead of six. This is three dimensional hexagonal close packing, and can be pictured by imagining dodecahedrons set on top of other dodecahedrons. These would rest on every other corner of the lower hexagons, so each cell would have six neighbors in its own plane, three more neighbors above and three more neighbors below. These are just a few suggestions, the possibilities are endless.

ACKNOWLEDGMENTS

We would like to acknowledge Ray Flanery for suggesting the data structure used and also for the graphics programs needed to generate the pictures produced from the output of SIMDLA.

We would also like to acknowledge Colin Sheppard for supplying us with numerous ideas and programming suggestions.

BIBLIOGRAPHY

[1] Eden, Murray, "A Two-Dimensional Growth Process," *Fourth Berkeley Symposium on Mathematical Statistics and Probability*, edited by Jerzy Neyman, **IV**, 223-239 (1961).

[2] Kolb, M., R. Jullien, R. Botet, "Scaling Properties of Cluster and Particle Aggregation," *Scaling Phenomena in Disordered Systems*, edited by Roger Pynn and Arne Skjeltorp, NATO Advanced Study Institute Series, Series B, Physics **133**, 71-78 (1985).

[3] Meakin, Paul, "Diffusion-Controlled Cluster Formation in 2-6-Dimensional Space," *Physical Review A* **27**, No. 3, 1495-1507 (1983).

[4] Meakin, Paul, "Diffusion-Controlled Formation in Two, Three, and Four Dimensions," *Physical Review A* **27**, No. 1, 604-607 (1983).

[5] Meakin, Paul, "Formation of Fractal Clusters and Networks by Irreversible Diffusion-Limited Aggregation," *Physical Review Letters* **51**, No. 13, 1119-1122 (1983).

[6] Nauenberg, M., R. Richter, L. M. Sander, "Crossover in Diffusion-Limited Aggregation," *Physical Review B* **28**, No. 3, 1649-1651 (1983).

[7] Peters, H. P., D. Stauffer, H. P. Hölters, K. Loewenich, "Radius, Perimeter, and Density Profile for Percolation Clusters and Lattice Animals," *Zeitschrift* für *Physik B* **34**, 399-408 (1979).

[8] Renker, Lisa Anne, D. G. Wilson (1988), "Survey of the Literature on Fractals with Possible Applications to Phase Change Processes," *ORNL/TM-10778*, June 1988.

[9] Sander, Leonard M., "Growth by Particle Aggregation," *Scaling Phenomena in Disordered Systems*, edited by Roger Pynn and Arne Skjeltorp, NATO Advanced Study Institute Series, Series B, Physics **133**, 31-47 (1985).

[10] Sander, L. M., P. Ramanlal, E. Ben-Jacob, "Diffusion-Limited Aggregation as a Deterministic Growth Process," *Physical Review A* **32**, No. 5, 3160-3163 (1985).

[11] Stauffer, D., "Scaling Theory of Percolation Clusters," *Physics Reports* **54**, No. 1, 1-74 (1979).

[12] Vicsek, Tamas, "Pattern Formation in Diffusion-Limited Aggregation," *Physical Review Letters* **53**, No. 24, 2281-2284 (1984).

[13] Witten, T. A., L. M. Sander, "Diffusion-Limited Aggregation, A Kinetic Critical Phenomenon," *Physical Review Letters* **47**, No. 19, 1400-1403 (1981).

[14] Witten, T. A., L. M. Sander, "Diffusion-Limited Aggregation," *Physical Review B* **27**, No. 9, 5686-5697 (1983).

INTERNAL DISTRIBUTION

1.	V. Alexiades	22.	J. M. Macdonald
2.	B. R. Appleton	23-27.	F. C. Maienschein
3.	L. C. Cain	28.	M. D. Morris
4.	C. S. Daw	29.	G. Ostrouchov
5.	J. B. Drake	30.	C. J. Remenyik
6.	Y. H. Etheridge	31-35.	R. C. Ward
7.	R. E. Flanery	36-40.	D. G. Wilson
8.	G. A. Geist	41.	P. H. Worley
9.	L. J. Gray	42.	Central Research Library
10-11.	R. F. Harbison	43.	K-25 Plant Library
12.	M. T. Heath	44.	ORNL Patent Office
13.	T. L. Hebble	45.	Y-12 Technical Library /Document Reference Station
14.	T. Kaplan	46.	Laboratory Records - RC
15-19.	J. K. Ingersoll	47-48.	Laboratory Records Department
20.	S. M. Lenhart		
21.	S. H. Liu		

EXTERNAL DISTRIBUTION

49. Dr. Donald M. Austin, ER-7, Applied Mathematical Sciences, Scientific Computing Staff, Office of Energy Research, Office G-437, Germantown, Washington, DC 20545

50. Dr. George I. Bell, T-7 Division, Los Alamos National Laboratory, P.O. Box 1663, Los Alamos, NM 87545

51. Dr. Bill L. Buzbee, C-3, Applications Support & Research, Los Alamos Scientific Laboratory, P. O. Box 1663, Los Alamos, NM 87545

52. Professor Alfredo Bermúdez de Castro, Universidad De Santiago, Dep. de Ecuaciones Funcionales, Facultad de Matematicas, Santiago de Compostela, SPAIN

53. Dr. Sam Coriell, U. S. National Bureau of Standards, Mathematical Analysis Section, Washington, DC 20234

54. Dr. James Corones, Ames Laboratory, Iowa State University, Ames, IA 50011

55. Dr. John J. Dorning, Department of Nuclear Engineering and Engineering Physics, Thornton Hall, University of Virginia, Charlottesville, VA 22901

56. Professor Antonio Fasano, Istituto Matematico Ulisse Dini, V. Le Morgagni 67/A, 50134 Firenze, ITALY

57. Professor Michael Fremond, Laboratoire Central des Pontes et Chaussees, Direction Scientific, 58 Boulevard Lefebvre, F-75732 Paris Cedex 15, FRANCE

58. Dr. Robert M. Haralick, Department of Electrical Engineering, University of Washington, Seattle, WA 98195

59. Professor Dr. Karl-Heinz Hoffman, Angewandte Mathematic I, Universität Augsburg, Memminger Strasse 6, D-8900 Augsburg, WEST GERMANY

60. Dr. Hans G. Kaper, Mathematics & Computer Science, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439

61. Dr. Robert J. Kee, Applied Mathematics Division, 8331, Sandia Laboratories, Livermore, CA 94550

62. Dr. W. Kurz, Laboratoire de metallurgie physique, Ecole polytechnique federale de Lausanne, Ch. de Bellerive 34-1007, Lausanne, SWITZERLAND

63. Dr. Alex Lehoczky, ES72 Space Science Laboratory, Marshall Space Flight Center, Huntsville, AL 35812

64. Dr. Richard A. Little, Mathematics and Computer Science Department, Baldwin-Wallace College, Berea, OH 44017

65. Dr. Basil Nichols, T-7, Mathematical Modeling and Analysis, Los Alamos Scientific Laboratory, P. O. Box 1663, Los Alamos, NM 87545

66. Professor Marek Niezgodka, Systems Research Institute, ul Newelska 6, 01-447 Warszawa, POLAND

67. Professor John R. Ockendon, Oxford University Computing Center, 24 St. Giles, Oxford OX1 3LB, UNITED KINGDOM

68. Dr. Ronald Peierls, Applied Mathematics Department, Brookhaven National Laboratory, Upton, NY 11973

69. Professor Mario Primiccerio, Inst. Matematico Ulisse Dini, Via Morgagni 67/A, I-50134 Firenze, ITALY

70-74. Ms. Lisa Renker, 15161 Dogwood Circle, Middleburg Heights, OH 44130

75. Dr. Timothy Riggle, Mathematics and Computer Science Department, Baldwin-Wallace College, Berea, OH 44017

76. Dr. Raoul Robert, Laboratoire I. M. A. G., B. P. No 68, 38402 Saint Martin d' Heres Cedex, FRANCE

77. Professor L. M. Sander, Physics Department, University of Michigan, Ann Arbor, MI 48109

78. Dr. Lawrence F. Shampine, Numerical Mathematics Division, 5642, Sandia Laboratories, P. O. Box 5800, Albuquerque, NM 87115

79. Professor Dr. Jürgen Sprekels, Angewandte Mathematic I, Universität Augsburg, Memminger Strasse 6, D-8900 Augsburg, WEST GERMANY

80. Dr. Don Steiner, Institute Professor, Department of Nuclear Engineering, Rensselaer Polytechnic Institute, Troy, NY 12181

81. Dr. Frank Szofran, ES72 Space Science Laboratory, Marshall Space Flight Center, Huntsville, AL 35812

82. Dr. Karl Hermann Tacke, Sonnerbergstrasse 7, 8134 Adliswil, SWITZERLAND

83. Professor Domingo Alberto Tarzia, Programme de Mathematical Pura y Applicada, Instituto de Matematica "Beppo Levi," Universidad Nacional de Rosario, Avenida Pelligrini 250, 2000 Rosario, ARGENTINA

84. Professor Rohit Trivedi, Ames Laboratory, Iowa State University, Ames, IA 50011

85. Professor Burton Wendroff, Mathematics Division, Los Alamos Scientific Laboratory, Los Alamos, NM 97544

86. Office of Assistant Manager for Energy Research and Development, U. S. Department of Energy, Oak Ridge Operations Office, P. O. Box 2001, Oak Ridge, TN 37831-8600

87-97. Office of Scientific and Technical Information, P. O. Box 62, Oak Ridge, TN 37830