Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be “used for any purpose other than private study, scholarship, or research.” If a user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of “fair use” that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select “Pages from: first page # to: last page #” on the print dialog screen
The Van Houten library has removed some of the personal information and all signatures from the approval page and biographical sketches of theses and dissertations in order to protect the identity of NJIT graduates and faculty.
Abstract

Title of Thesis: An Analysis of the Dense Packing of Hard Disks
—A Computer Simulated Approach.


Thesis directed by: Dr. Anthony D. Rosato, Assistant Professor

Mechanical Engineering Department.

This thesis is concerned with the analysis of dense packing of hard disks. The Voronoi diagram and the geometric neighbours were first computed. The average number of geometric neighbours of a disk is six. It is thus more efficient to choose structural neighbours from among the geometric neighbours than from among all other disks.

Through the Monte Carlo simulation by Rosato et. al., disk configurations after pouring and subsequent shaking were provided for analysis. The mean number of geometric neighbours and the average coordination number were computed. The angular distribution of the structural neighbours was discussed. The packing fraction increases with number of shakes in a linear relationship. It seems to be packing into an ordered close packing after continued shaking.

A configuration constrained by two rigid vertical walls was analyzed. It was found the packing fraction is smallest in the vicinity of the wall and increased asymptotically to the mean packing fraction when moving away from the wall.
An Analysis of the Dense Packing of Disks
—A Computer Simulated Approach

by
Jir-Yih Tsaur

Thesis submitted to the Faculty of the Graduate School of
the New Jersey Institute of Technology in partial fulfillment of
the requirements for the degree of
Master of Science in Mechanical Engineering
1988
Title of Thesis: An Analysis of the Dense Packing of Disks
— A Computer Simulated Approach

Name of Candidate: Jir-Yih Tsaur

Master of Science in Mechanical Engineering

Thesis and Abstract Approved: ____________________________

Anthony D. Rosato Date

Assistant Professor

Mechanical Engineering Department

Signatures of other members of the thesis committee: ____________________________

Date

Date
Vita

Name: Jir-Yih Tsaur

Permanent address:

Degree and date to be conferred: MSME, 1988

Date of birth:

Place of birth:

Secondary education: Chien Kuo Senior High School, 1974-76

<table>
<thead>
<tr>
<th>Collegiate institutions attended</th>
<th>Dates</th>
<th>Degree</th>
<th>Date of Degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>National Taiwan College of Marine Science and Technology</td>
<td>1977-82</td>
<td>BSME</td>
<td>1982</td>
</tr>
<tr>
<td>New Jersey Institute of Technology</td>
<td>1986-88</td>
<td>MSME</td>
<td>1988</td>
</tr>
</tbody>
</table>

Major: Mechanical Engineering
Acknowledgement

The thesis would not have been completed without help from a lot of kind people. Special thank goes to my advisor Dr. Anthony D. Rosato. He inspired my interest in this research, lead me through while I was confused, with the most enthusiasm I have ever seen. Further, I would like to thank Mr. Y. Lan and Mr. S. Chung, who helped data processing; Mr. K. Reddy, Mr. S. Ishwar, and Mr. S. Dikmenli who advised both on hardwear and softwear problems.
To dear dad and mom.
Contents

1 Introduction ................................................................. 1
  1.1 Motivation and Literary Survey ....................................... 1
    1.1.1 The Packing of Spheres ......................................... 1
    1.1.2 The Packing of Disks ........................................... 3
  1.2 Objectives and Methods ............................................... 6
  1.3 Outline of Chapters .................................................. 7

2 Voronoi Diagram .......................................................... 9
  2.1 Proximity Problems .................................................... 9
  2.2 The B-R Method ....................................................... 10
    2.2.1 Sorting .......................................................... 11
    2.2.2 Fixing the Viewing Box ......................................... 11
    2.2.3 Processing Loop ............................................... 15

3 Random Close Packing .................................................... 30
  3.1 Definition ............................................................ 30
  3.2 Radial Distribution Function Characteristics of A Random Close Packing 33
  3.3 Monte Carlo Simulation ............................................... 34
  3.4 Analysis of the Monte Carlo Simulation Output ....................... 35
    3.4.1 Radial Distribution Function .................................... 36
    3.4.2 Average Number of Geometric Neighbours ....................... 39
3.4.3 Average Number of Structural Neighbours — Mean Coordination Number ........................................ 43
3.4.4 Angular Distribution of Structural Neighbours ............... 54
3.4.5 Packing Fraction ........................................ 55
3.4.6 Rigid Wall Effect ........................................ 61

4 Summary ........................................ 70

A Definitions and Properties Regarding Voronoi Diagram .......... 74
B Algorithm Finding Next Clockwise Geometric Neighbour .......... 91
C Strategy on Completing the Voronoi Diagram ...................... 92
D Program disk.pas ........................................ 93

References ........................................ 152
List of Figures

1  The Voronoi diagram and Delaunay triangulation of 14 disks with a selected value of 1 for the constant $BoxC$. .......................... 13

2  The Voronoi diagram and Delaunay triangulation of 14 disks with a selected value of 10 for the constant $BoxC$.  ...................... 14

3  Geometric neighbours and dummy points associated with point A, the first point processed. ............................................. 16

4  An ensemble of 8 disks. ......................................................... 21

5  Radial distribution function for a 1,000 disk configuration after pouring. 37

6  Radial distribution function for a 1,000 disk configuration after 20 cycles of shakings. .................................................... 38

7  The fixing of a inner box within the viewing box. ................. 41

8  Distribution of geometric neighbour number. ................................. 42

9  Voronoi diagram for an ordered close packing. ............................. 44

10  Delaunay triangulation for an ordered close packing. .................. 45

11  Cumulative probability of the normalized nearest neighbour radius $r/D$ for 1,000 disks after pouring. ............................... 47

12  Cumulative probability of the normalized nearest neighbour radius $r/D$ for 1,000 disks after shaking. ............................... 48

13  Relationship between coordination number and tolerance. ............ 49

14  Distribution of structural neighbour number after pouring. ........... 50
15 Distribution of structural neighbour number after shaking.  
16 Average coordination number versus shaking: 1,000 disks with periodic 
boundary conditions. ............................................. 52 
17 Average coordination number versus shaking: 1,250 disks with non-
periodic boundary conditions. ................................. 53 
18 Angular distribution of each structural neighbour for disks possessing 
two structural neighbours. ..................................... 56 
19 Angular distribution of each structural neighbour for disks possessing 
three structural neighbours. .................................. 57 
20 Angular distribution of each structural neighbour for disks possessing 
four structural neighbours. .................................... 58 
21 Angular distribution of each structural neighbour for disks possessing 
five structural neighbours. .................................... 59 
22 Angular distribution of each structural neighbour for disks possessing 
six structural neighbours. .................................... 60 
23 Energy variation versus shaking: 1,000 disks with periodic boundary 
conditions............................................................ 62 
24 Energy variation versus shaking: 1,250 disks with non-periodic bound-
ary conditions. ................................................... 63 
25 Packing fraction variation versus shaking: 1,000 disks with periodic 
boundary conditions. .......................................... 64
26 Packing fraction variation versus shaking: 1250 disks with non-periodic boundary conditions. .................................. 65

27 Packing fraction versus average coordination number: 1,000 disks with periodic boundary conditions. .................................. 66

28 Packing fraction versus average coordination number: 1,250 disks with non-periodic boundary conditions. .................................. 67

29 Influence of rigid walls on packing fraction for a 1,250 disk configuration. 69

30 Lemma 1. ........................................ 76

31 The extended angle AOB. ........................................ 78

32 Lemma 2. ........................................ 80

33 Delaunay triangulation of 1,000 disks after pouring. .................... 82

34 The configuration of 1,000 disks after pouring —courtesy of A. Rosato, K. Reddy, and Y. Lan. ........................... 83

35 The configuration of 1,000 disks after 20 cycles of shaking —courtesy of A. Rosato, K. Reddy, and Y. Lan. ........................... 84

36 Voronoi diagram of 1,000 disks after pouring. ........................... 85

37 Theorem 1. ........................................ 87

38 Theorem 2. ........................................ 90
List of Tables

1  Normalized median nearest neighbour radii versus cycle number for a 1,000 disk configuration. ........................................... 32
2  Radial distribution peaks in a two-dimensional ordered close packing. 34
3  Cases Studied. .................................................................. 36
4  Mean values for each structural neighbour angle. .................... 55
5  Standard deviation for each structural neighbour angle. .......... 55
1 Introduction

1.1 Motivation and Literary Survey

The study of packing of spheres and disks have been of interest for modelling the structure of granular materials, liquids and glasses. A survey on related research which gives rise to the work in this thesis is presented in this Chapter.

1.1.1 The Packing of Spheres

Dated as early as 1929, the three-dimensional packing of homogeneous spheres was studied by W. O. Smith et. al. [50]. And, as late in 1983, J. G. Berryman [13] presented his definition on random close packing and found the critical packing fraction for random close packing of both spheres and disks.

J. D. Bernal [9] studied the structure of liquids by viewing it as a "heap" of particles of the same size. A liquid in Bernal’s original postulate [10] was homogeneous, coherent and essentially irregular, as contrasted with a crystal, which may be called a "pile" of molecules, regularly arranged. In respect to order, a liquid is different from a crystal and resembles a gas. On the other hand, with respect to density, a liquid would resemble a crystal and differ from a gas.

Through an experiment, Bernal simulated a liquid by a heap of spheres, a close arrangement of particles, where each is in contact with a large number of others but not in any regular way. He calculated the packing density and the average number of geometric neighbour for the model he created. The packing densities were calculated
to be 0.637 for random close packing and 0.60 for random loose packing, as compared to 0.74 for ordered close packing. And, the average number of geometric neighbours was calculated to be 13.6 as compared to 12 for ordered close packing.

Bernal defined the structural neighbours as those having their centroids within 1.05 diameter away from each other. The tolerance 0.05 was experimentally determined in the following procedure by Bernal. A number of steel ball bearings having a light coating of grease were packed in a balloon. Paint was then poured through the bearings. The paint was left, held by surface tension where the balls touched or nearly touched. When the paint was dry and the balloon removed, the spheres within 5% of a diameter away from each other had a dot or a ring formed at the place where they were in contact. By counting number of dots or rings on the surface of a sphere, the number of structural neighbours for the sphere, or the coordination number, was found.


More literature can be found in [3], [37], and [1]. For example, S. Nemat-Nasser et. al. [38] used a sample model in an experiment and found that particle
rolling instead of particle sliding is the major microscopic deformation mechanism.

S. B. Savage et. al. [18] derived the general conservation equations for the rapid flow of a binary mixture of smooth, inelastic, spherical granular particles following the approach of the kinetic theory for mixtures of dense gases.

Other studies of sphere packing exist in the literature, including computer-simulations and theoretical model. They are listed in the Reference and will not be discussed respectively here.

1.1.2 The Packing of Disks

While much work exists on the packing of spheres, the literature is not as extensive with regard to the nature of the structure of the packing of disks. The exact nature of disk packings is not as well understood as its three-dimensional counterpart. There seems to be no analogue to the concept of "loose" and "close" packings with regard to disks as there is for spheres. Disk packings are generally used as a model for monolayer structures.

G. T. Barnes et. al. [6] used packing of disks as a model to study the covering of a liquid surface by surfactant molecules, enabling the rate of evaporation through the "holes" in the surface layer to be deduced. The model of Barnes pictured the monolayer on the surface of water as an array of randomly placed hard disks on a plane surface, and the evaporating water molecules as hard spheres. The model assumed that water molecules could penetrate a monolayer only when void larger than a water
molecule appeared in the monolayer, and that water evaporates through these voids at the usual rate for a free surface. Thus, the reduction of evaporation rate caused by a monolayer is simply a consequence of the reduction of voids in the monolayer. After obtaining the void area, Barnes calculated the evaporation/condensation rate.

Krishna K. Pillai [39] used packed disks as a model to find the cross-sectional voidage variation in the vicinity of a wall for a packed bed of particles. Hard disks were used in an experiment to simulate the particles, which would appear as a monolayer of spheres and have the same front view as that of a disk model. When considering heat conduction between particles along the bed, it is necessary to know the cross-sectional area of solids perpendicular to the direction of heat flow. Though the average void fraction within the bed can be calculated, the existence of a constraining wall will alter the particle packing on the cross-sectional area of the solid in the vicinity of the wall. Through Pillai's model, the packing fraction of a cross-section at some distance \( x \) away from the wall can be calculated. A packing fraction versus \( x \) relationship was derived, which could be described by a curve oscillating in the vicinity of the wall, damped and converging to a mean value at approximately three diameters away from the wall.

Using the fact that the packing density variation near the wall is almost the same as that about a central sphere in the interior, Tory et. al. [23] introduced the molecular pair-correlation function to explain the variation in the local packing density. The results obtained agree with that of Pillai's discussed in the previous paragraph fairly
Quickenden and Tan [41] simulated monolayer structures using a method proposed by Stillinger et. al. [51]. This was done by randomly placing homogeneous disks on an isotropically stretched and subsequently contracted rubber film. The process modeled the compression of a monolayer on a liquid surface. The packing fraction of the disks increased with the contraction of the rubber film rapidly until a value of 0.83 was reached, after which it increased slowly and asymptotically to the value of 0.906 for a ordered close packing. It is interesting that the packing fraction of 0.83 is close to that at the phase transition, which occurs when monolayers of long chain acids or alcohols on a water surface are compressed from the liquid-condensed to the solid phase. Quickenden and Tan thus declared that it is possible to explain and calculate the position of the change from the solid to the liquid-condensed phase, purely on the basis of an array of non-interacting disks.

Following the same idea from Stillinger, Mason [33] modeled a two-dimensional packing of disks through computer simulation. Instead of contracting the coordinate system or moving the disks toward a point inside the packing, Mason kept the disks at the same coordinates and increased their size to densify the packing. Consequently, this technique produced several disks overlapping each other which is mechanically impossible for inelastic hard disks. Two disks in a overlapping pair were then moved equally apart along the line of their centers until they are just touching. This process, called shuffling by Mason, was repeated until no disk remained overlapped. Mason
achieved the same result as Quickenden and Tan, i.e., an ordered close packing would finally be formed.

1.2 Objectives and Methods

The packing structures of assemblies of disks are analyzed. They are created via a Monte Carlo computer simulation developed by Rosato et. al [42]. A locus method which solves many proximity problems in computational geometry, the Voronoi diagram [58], is applied to find geometric neighbours for each disk. An algorithm originated from the one constructed by Kurt E. Brassel and Douglas Reif [14] is chosen to create the Voronoi diagram. The algorithm is implemented in Pascal because the language facilitates the handling of dynamic data structures and recursive subroutine calls. The distribution of the number of geometric neighbours possessed by a disk is computed. The average number is found to be exactly six.

Following the convention of J. D. Bernal, we define disks having their centroids within 1.05 diameter apart from each other as structural neighbours. Structural neighbours are chosen for each disk from among its geometric neighbours, which is more efficient than from among all other disks in the configuration. The distribution of the number of structural neighbours possessed by a disk, the coordination number is calculated.

Included among the geometric neighbours also is a nearest neighbour (Theorem 2, Appendix A), which is nearest to the center disk among all geometric neighbours. In case of a dense packing, which is the topic of this thesis, it is also true that the nearest
neighbour is in touch with the center disk. Therefore, the nearest neighbour of a disk can be found within a even smaller subset composed of the structural neighbours. The cumulative probability of nearest neighbour radii is obtained and gives the median nearest neighbour radius.

A vector is extended from the center disk to each of its structural neighbours. An angle, which is called *structural neighbour angle* is associated with each of the vectors. For a group of disks having M structural neighbours, where M is less than or equal to six, the frequency distribution for each of the M angles is found, from which, the mean and deviation are calculated. Finally, Packing fractions are computed for all assemblies and the variation of the packing fraction in the vicinity of a wall is determined.

1.3 Outline of Chapters

Chapter Two introduces the Voronoi diagram. A variety of proximity related problems which can be solved by Voronoi diagram are discussed. The B-R method is described, which is applied to construct the Voronoi diagram.

Random close packing of hard disks is then discussed in Chapter Three. Definitions and relations are presented. The radial distribution function is introduced, which gives a better understanding of the concept of randomness. The Monte Carlo simulation is described and its output is analyzed.

The Summary and Conclusion are given in Chapter Four. The Appendix contains definitions and properties regarding Voronoi diagram, simplified flow charts for the
algorithm finding next geometric neighbour and completing the Voronoi diagram, and the program *disk.pas*. 
2 Voronoi Diagram

The Voronoi diagram is applied to find geometric neighbours for each disk. From the geometric neighbours, structural neighbours and the nearest neighbour can be selected efficiently. Various algorithms exist for the creation of the Voronoi diagram [15,30,48]. The algorithm by Kurt E. Brassel and Douglas Reif [14] is chosen and will be discussed in section 2.2.

2.1 Proximity Problems

Geometric objects, such as points, circles, and spheres, are sometimes used to model physical entities in the real world. In some cases we would like to have access to a suitable neighbourhood of the objects. For instance, in air traffic control it is important to keep track of the closest two aircrafts. In studying the configuration of an ensemble of granular particles, like liquid, or powdered materials, knowing the local environment of each particle as well as properties is desirable. The above are examples of proximity related problems. Some typical problems in this discipline [29] are listed below:

- *(Closest pair)*: Given \( n \) points in the plane, find two points that are closest.

- *(All nearest neighbours)*: Given \( n \) points in the plane, find for each point a nearest neighbour (other than itself).

- *(Euclidean minimum spanning tree, EMST)*: Given \( n \) points in the plane, find a tree that interconnects all the points with minimum total edge length.
• (Triangulation): Given $n$ points in the plane, construct a planar graph on the set of points such that each bounded polygon within the convex hull of the $n$ points is a triangle.

• (Nearest neighbour search): Given $n$ points in the plane (with preprocessing allowed), find the nearest neighbour of a query point. This problem is also known as the "post office problem".

• ($k$ Nearest neighbours search): The same as the nearest neighbour search problem, except that the $k$ nearest neighbours are sought.

All the above listed kinds of problems can be solved by resorting to the locus method—the Voronoi diagram.

2.2 The B-R Method

The algorithm constructed by Kurt E. Brassel and Douglas Reif, hence termed the B-R Method, is chosen for the creation of the Voronoi diagram. In viewing the Voronoi diagram, a Box was first fixed. The computation of Voronoi diagrams uses an iterative walking process, whereby the processing starts at the lower left corner of the diagram. The use of a sorted array of linked list of points provides for a dynamic data structure and efficient processing.
2.2.1 Sorting

Given a set $S$ of $n$ points $\{p_1, p_2, \cdots, p_n\}$, we first sort it in lexicographical order. A point $p_1(x_1, y_1)$ is lexicographically less than a point point $p_2(x_2, y_2)$ if and only if either one of the following condition is satisfied:

1. $x_1 < x_2$.
2. $x_1 = x_2$ and $y_1 \leq y_2$.

To access a point/points within a range, only this particular range will be searched. All data out of this is neglected.

A merge sort algorithm by Sedgewick [46] was applied in the sorting process. The running time of the merge sort is of order $n \log n$ as opposed to the $n^2$-order of the insertion sort, selection sort, or bubble sort.

2.2.2 Fixing the Viewing Box

In viewing the Voronoi diagram, we fix a Box containing all the centroids of the given set of disks.

**Definition 1** The viewing box for the Voronoi diagram of a disk set $S$ is the region:

$$
Box \equiv \{(x, y) \mid x_{\min} - BoxC \cdot D_{\max} \leq x \leq x_{\max} + BoxC \cdot D_{\max};
\quad y_{\min} - BoxC \cdot D_{\max} \leq y \leq y_{\max} + BoxC \cdot D_{\max}\}
$$

in which $x_{\min}, x_{\max}, y_{\min}$ and $y_{\max}$ are minimum and maximum coordinates of the centroids of the disks in $S, D_{\max}$ is the maximum diameter of the disks, and $BoxC$
is a selected constant.

The choice of the constant \( \text{BoxC} \) introduces slight differences in the number of "unbounded" Voronoi polygons on the boundary of the Voronoi diagram. The larger the constant \( \text{BoxC} \), the smaller will be the number of unbounded Voronoi polygons on the boundary of the Voronoi diagram. A minimum number of unbounded Voronoi polygons on the boundary of the Voronoi diagram, which is exactly the number of disks with centroids located on the convex hull of the disk set, will result when the constant \( \text{BoxC} \) reaches \( \text{BoxC}_{lb} \), its lower bound.

As an example, let a set of 14 disks be given with \( D_{max} = 1 \) unit and \( x_{min} = 1.5, \) \( x_{max} = 9, \) \( y_{min} = 1, \) and \( y_{max} = 9.2. \) If the value of 1 is selected for the constant \( \text{BoxC}, \) according to Definition 1, the viewing box will be defined by the lines: \( x = 0.5, \) \( x = 10, \) \( y = 0, \) and \( y = 10.2 \) as shown in Figure 1. If the value of 10 is selected for the constant \( \text{BoxC}, \) the viewing box will then be defined by: \( x = -8.5, \) \( x = 19, \) \( y = -9, \) and \( y = 19.2 \) as shown in Figure 2. Refer to Figures 1 and 2 the unbounded Voronoi polygons are identified as those having an edge lying on the edges of the viewing box. The diagram in Figure 2 has 9 unbounded Voronoi polygons, while the diagram in Figure 1 has 10 unbounded Voronoi polygons.

It is desirable to generate a Voronoi diagram which contains only Voronoi points at the neighbourhood of the configuration. This is achieved by assigning a small value, a value of 1 for example, for the constant \( \text{BoxC} \) in Definition 1, and by adding some dummy points to the query point in each processing loop discussed in the following
Figure 1: The Voronoi diagram and Delaunay triangulation of 14 disks with a selected value of 1 for the constant BoxC.
Figure 2: The Voronoi diagram and Delaunay triangulation of 14 disks with a selected value of 10 for the constant *BoxC*. 
2.2.3 Processing Loop

Beginning with the set of disk center coordinates, a set of points, the box is fixed as previously described and the processing loop begins. The point nearest to the lower left corner of the box, point A in Figure 3, is selected as the query point in the first loop. Some assignment are executed before the first loop commences.

Four dummy points dummy1, dummy2, dummy3, and dummy4 are introduced so that the perpendicular bisectors between point A and the four dummy points coincide with the four edges of the viewing box. In Figure 3 it can be seen that dummy4 and dummy1 are neighbours of point A, and $V_0$ is a vertex of Voronoi polygon about point A. Further searching for the next clockwise neighbour, i.e., the next neighbour clockwise to dummy1 will be performed in the loop as described in the following.

The procedure to search for the next clockwise neighbour is based on the following properties given in Appendix A.

- **Corollary 1** — Voronoi polygon are always star-shaped.

- **Corollary 2** — For a point having a bounded Voronoi polygon, the next clockwise neighbour is always located in the right half plane of the vector from the query point to the present neighbour. (By the introduction of the four dummy points, Voronoi polygons associated with the given set of points are all bounded.)

- **Lemma 3** — A point $p$ equidistant to three points $p_i$, $p_j$, and $p_k$ is a true Voronoi
Figure 3: Geometric neighbours and dummy points associated with point A, the first point processed.
point if no other point is located within a circle centered at point $p$ with radius the distance from point $p$ to either one of the three points $p_i, p_j,$ and $p_k$.

Based on the three properties, we use the following strategy in the processing loop finding the remaining geometric neighbours of a disk. Take disk $A$ for example, in which dummy4 and dummy1 were two geometric neighbours and $V_0$ was one Voronoi point already known; dummy4 is predefined as $NbrFirst[1]^1$; dummy1 is predefined as $NewNbr^2$:

1. Define the two half planes $E_r(A, NewNbr)$ and $E_l(A, NewNbr)$, such that $E_r(A, NewNbr)$ is the right half plane of vector $\overrightarrow{A NewNbr}$. $E_l(A, NewNbr)$ is the left half plane of vector $\overrightarrow{A NewNbr}$.

2. Assign the first clockwise dummy point in the right half plane $E_r(A, NewNbr)$ as the potential new neighbour, denoted $PNewNbr$, of $A$. Find the point $V$ equidistant to $A$, $NewNbr$, and $PNewNbr$ by intersecting perpendicular bisectors $B(A, NewNbr)$ and $B(NewNbr, PNewNbr)$. $V$ is termed the potential new vertex. $A$, $NewNbr$, and $PNewNbr$ form a circle $(V, R)$ with center $V$ and radius $R$.

3. Search for point $P'$ among the given point set, which is both in the right half plane $E_r(A, NewNbr)$ and inside the circle $(V, R)$. If such a point $P'$ exists,

---

1The symbol $NbrFirst[k]$ will be used for the first geometric neighbour found for the $k$th query disk undergoing processing.

2The symbol $NewNbr$ will be used for the new geometric neighbour found for the disk undergoing processing.
assign $P'$ as the $P_{NewNbr}$, calculate potential new vertex $V'$ and new radius $R'$, where $R' < R$.

4. Repeat Step 3 until there is no point in the given point set which is both in the right half plane $E_r(A, NewNbr)$ and in the circle $(V, R)$, or $(V', R')$ if exists. Then, declare $P_{NewNbr}$ as $NewNbr$ and $V'$ as the new vertex.

5. Redefine half planes $E_r(A, NewNbr)$ and $E_l(A, NewNbr)$. Repeat Steps 2, 3, and 4 until a complete walk around $A$ is completed; i.e., until a $NewNbr$ is identical with $NbrFirst[1]$, the first neighbour of the query point $A$. At this point, all geometric neighbours of point $A$ have been computed.

Following the above strategy and referring to Figure 3, the geometric neighbours of point $A$ are calculated in the following sequence:

1. Dummy4 is known as the first neighbour and dummy1 is known as $NewNbr$. Define half planes $E_r(A, dummy1)$ and $E_l(A, dummy1)$.

2. Select dummy2 as the $P_{NewNbr}$. Compute $V_6$ and $R$ by intersecting bisectors of $Adummy1$ and $Adummy2$. Search for a point among given point set which both lies in $E_r(A, dummy1)$ and in the circle $(V_6, R)$. The search is in the lexicographical order indicated by the dashed line in Figure 3. Since $B$ is both in $E_r(A, dummy1)$ and $(V_6, R)$, it is assigned the $P_{NewNbr}$. Compute $V_1$ and $R'$. As no other point is within $(V_1, R')$, $B$ is declared as the $NewNbr$, and $V_1$ the new vertex.
3. Redefine half planes $E_r(A, B)$ and $E_l(A, B)$. Again, the $PNewNbr$ starts from dummy2. Compute $(V, R)$. Find $C$ within $E_r(A, B)$ and $(V, R)$. Assign $C$ to be the $PNewNbr$, compute $(V_2, R')$. Since there is no point within $(V_2, R')$, declare $C$ as the $NewNbr$, $V_2$ the new vertex.

4. Redefine half planes $E_r(A, C)$ and $E_l(A, C)$. This time the $PNewNbr$ starts from dummy3. Compute $(V, R)$. Find $F$ as the second $PNewNbr$, compute $(V', R')$. Find $D$ within $E_r(A, C)$ and $(V', R')$, assign $D$ the $PNewNbr$, compute $(V_3, R')$. No point exists in $(V_3, R')$. Declare $D$ as the $NewNbr$, $V_3$ the new vertex.

5. Redefine half planes $E_r(A, D)$ and $E_l(A, D)$. Select dummy3 as the first $PNewNbr$, compute $(V, R)$. Find $E$ both lies in $E_r(A, D)$ and $(V, R)$, assign $E$ the $PNewNbr$, compute $(V_4, R')$. No other point lies in $(V_4, R')$, $E$ is declared the $NewNbr$, and $V_4$ the new vertex.

6. Redefine half planes $E_r(A, E)$ and $E_l(A, E)$. Dummy4 is assigned the first $PNewNbr$, $(V_5, R)$ are computed. No point in the given point set lies in $(V_5, R)$, dummy4 is declared the $NewNbr$.

7. Now that the $NewNbr$ is identical with $NbrFirst$, a completed walk around point $A$ has been completed; i.e., the Voronoi polygon about point $A$ has been constructed.

Because each vertex $V_i$ pertains to three Voronoi polygons, all neighbourhood
relationships are mutual. Once a vertex is found, it is stored and the mutual relationships among the centroids involved are recorded in a bookkeeping process. Referring to Figure 3 again, when point $B$ was found as a geometric neighbour of point $A$, with a common Voronoi point $V_1$ among points $A$, dummy1, and $B$, point $A$ was also recorded as a geometric neighbour of point $B$. And, when point $C$ was found as the next geometric neighbour of point $A$, with a common Voronoi point $V_2$ among points $A$, $B$, and $C$, point $C$ is also recorded as a geometric neighbour of point $B$. When the first loop ceases, five records exist in the bookkeeping and record geometric neighbours for points $A$, $B$, $C$, $D$, and $E$.

Once the first loop which processes point $A$ has been completed, point $B$, the first non-dummy geometric neighbour of point $A$ is processed next. The processes continue until all points have been processed and the Voronoi diagram completed. Consider the eight points in Figure 4, the complete process is tabulated in the following. Whenever a new vertex $V_i$ is found, the bookkeeping for the three related centroids will be modified and appear in bold face in the table. Though dummy points are also shown for better understanding of the complete mutual relationship, they will not be recorded in the bookkeeping. Also, a centroid or vertex which has previously been recorded will not be recorded in duplicate and will be put in a pair of parentheses.

$V_0$: $A$ — dummy4, dummy1

dummy4 — dummy1, A
Figure 4: An ensemble of 8 disks.
dummy1 — A, dummy4

\[ V_1: A \rightarrow {\text{dummy4, dummy1, B}} \]

dummy4 — dummy1, A

dummy1 — B, A, dummy4

B — A, dummy1

\[ V_2: A \rightarrow {\text{dummy4, dummy1, B, C}} \]

dummy4 — dummy1, A

dummy1 — B, A, dummy4

B — C, A, dummy1

C — A, B

\[ V_3: A \rightarrow {\text{dummy4, dummy1, B, C, D}} \]

dummy4 — dummy1, A

dummy1 — B, A, dummy4

B — C, A, dummy1

C — D, A, B

D — A, C

\[ V_4: A \rightarrow {\text{dummy4, dummy1, B, C, D, E}} \]
\texttt{dummy4} — \texttt{dummy1}, A
\texttt{dummy1} — B, A, \texttt{dummy4}
B — C, A, \texttt{dummy1}
C — D, A, B
D — E, A, C
E — A, D

\texttt{V_5}: A — \texttt{dummy4}, \texttt{dummy1}, B, C, D, E, (\texttt{dummy4})
\texttt{dummy4} — \texttt{dummy1}, A, E
\texttt{dummy1} — B, A, \texttt{dummy4}
B — C, A, \texttt{dummy1}
C — D, A, B
D — E, A, C
E — \texttt{dummy4}, A, D
(\texttt{dummy4} — A, E)

Now that the \textit{NewNbr} \texttt{dummy4} was the first neighbour, the geometric neighbours of A have all been found and the Voronoi polygon about A has been completed. The first geometric neighbour of point A is now to be processed. By skipping the dummy points \texttt{dummy4} and \texttt{dummy1}, we choose point B with known neighbours C and A, where C is the first neighbour and A is the \textit{NewNbr}. Shift the dummy point to new positions such that the bisectors of the dummy points and B generate the four edges
of the viewing box. To continue:

\[(V_1): \text{B} \rightarrow \text{C, A, dummy1}\]
\[\text{C} \rightarrow \text{D, A, B}\]
\[\text{D} \rightarrow \text{E, A, C}\]
\[\text{E} \rightarrow \text{A, D}\]
\[(\text{A} \rightarrow \text{dummy1, B})\]
\[\text{dummy1} \rightarrow \text{B, A}\]

\[V_6: \text{B} \rightarrow \text{C, A, dummy1, dummy2}\]
\[\text{C} \rightarrow \text{D, A, B}\]
\[\text{D} \rightarrow \text{E, A, C}\]
\[\text{E} \rightarrow \text{A, D}\]
\[\text{dummy1} \rightarrow \text{dummy2, B, A}\]
\[\text{dummy2} \rightarrow \text{B, dummy1}\]

\[V_7: \text{B} \rightarrow \text{C, A, dummy1, dummy2, F}\]
\[\text{C} \rightarrow \text{D, A, B}\]
\[\text{D} \rightarrow \text{E, A, C}\]
\[\text{E} \rightarrow \text{A, D}\]
\[\text{dummy1} \rightarrow \text{dummy2, B, A}\]
\[\text{dummy2} \rightarrow \text{F, B, dummy1}\]
$F \rightarrow B$, dummy2

$V_8$: $B \rightarrow C$, A, dummy1, dummy2, F, (C)

$C \rightarrow D$, A, B, F

$D \rightarrow E$, A, C

$E \rightarrow A$, D

dummy1 $\rightarrow$ dummy2, B, A

dummy2 $\rightarrow$ F, B, dummy1

$F \rightarrow C$, B, dummy2

(C $\rightarrow$ B, F)

The Voronoi polygon about B is now completed. Now, start processing C with known neighbours D, A, B, and F.

$V_9$: $C \rightarrow D$, A, B, F, (D)

$D \rightarrow E$, A, C, F

$E \rightarrow A$, D

$F \rightarrow D$, C, B

(D $\rightarrow$ C, F)

The Voronoi polygon about C is now completed. The next point to be processed will be D, with known neighbours E, A, C, and F.

$V_{10}$: $D \rightarrow E$, A, C, F, G
E — A, D
F — G, D, C, B
G — D, F

$V_{11}$: D — E, A, C, F, G, H
E — A, D
F — G, D, C, B
G — H, D, F
H — D, G

$V_{12}$: D — E, A, C, F, G, H, (E)
E — A, D, H
F — G, D, C, B
G — H, D, F
H — E, D, G
(E — D, H)

Continuing on:

$V_{13}$: E — A, D, H, dummy3
F — G, D, C, B
G — H, D, F
H — dummy3, E, D, G
dummy3 — E, H

$V_{14}$: E — A, D, H, dummy3, dummy4

F — G, D, C, B

G — H, D, F

H — dummy3, E, D, G

dummy3 — dummy4, E, H

dummy4 — E, dummy3

$(V_5)$: E — A, D, H, dummy3, dummy4, (A)

F — G, D, C, B

G — H, D, F

H — dummy3, E, D, G

dummy3 — dummy4, E, H

dummy4 — A, E, dummy3

(A — E, dummy4)

To keep on:

$(V_7)$: F — G, D, C, B, dummy2

G — H, D, F

H — E, D, G

(B — dummy2, F)
dummy2 — F, B

$V_{15}$: F — G, D, C, B dummy2, (G)

G — H, D, F, dummy2

H — E, D, G

dummy2 — G, F, B

(G — F, dummy2)

For point G:

$(V_{15})$: G — H, D, F, dummy2

H — E, D, G

(F — dummy2, G)

dummy2 — G, F

$V_{16}$: G — H, D, F, dummy2, (H)

H — E, D, G, dummy2

dummy2 — H, G, F

(H — G, dummy2)

Finally, for the last centroid $H$:

$(V_{16})$: H — E, D, G, dummy2

(G — dummy2, H)
After a NewNbr $E$ is found which is the same as $NbrFirst$ of the query point $H$, the last Voronoi polygon is completed. The Voronoi diagram is also finished because all points have been processed. The complete frameworks finding the next geometric neighbour and completing the Voronoi diagram are shown in Appendices B and C.

Once the entire diagram is completed, an array of linked list of geometric neighbours and another array of linked list of vertices are created and stored in two different files. Each element of these arrays is associated with a point of the set. Using the array in the two files the Voronoi diagram and the Delaunay triangulation [16] can be drawn. The Voronoi diagrams associated with the set are shown in solid lines in Figures 1 and 2, and the Delaunay triangulations are shown in dashed lines.
3 Random Close Packing

3.1 Definition

Given an ensemble of disks, first, is it an ordered packing or a random packing? Second, is it a close packing or not?

A random packing differs from an ordered packing in that the latter has a regular periodic structure with a fixed lattice spacing. Berryman [13] had given the following geometric interpretation of random packing:

1. A random packing is a packing containing no statistically significant short- or long-range order.

2. A random close packing is a random packing so dense such that the density of it will not be increased without significant increase in short-range order.

3. From (1) and (2) above, it is derived that a disk in a random close packing must be in contact with several others.

4. Any decrease in density makes a random close packing not close, though still random, anymore. It is because a given disk will not necessarily be in contact with another disk when the density is decreased.

Obviously, Berryman’s description was based on geometry only and was independent of thermodynamic interpretation.

Based on the previous concepts, Berryman had made the following definitions:
Definition 2 Let $D$ be the diameter of the disks in the random packing, $P(r \leq R)$ be the cumulative probability that the nearest neighbour of a disk at the origin is at some radius $r$ in the range $D < r < R$. Then, for a fixed packing fraction $\eta$, the median nearest neighbour radius $R_{MNN}(\eta)$ is defined implicitly by

$$P(r \leq R_{MNN}) = \frac{1}{2}. \quad (1)$$

For a dilute packing of disks, the median nearest neighbour radius will be large, tending to infinity as the packing density $\eta$ approaches 0. As the density increases toward random close packing, the median nearest neighbour radius will approach $D$. Furthermore, it follows from our previous discussion that

$$R_{MNN} = D \quad (2)$$

is characteristic of random close packing since each disk must be touching its nearest neighbour in such a packing. Thus, the problem of determining whether a disk configuration forms a close packing can be answered by checking if Eq. (2) is satisfied. However, it has been found that $R_{MNN}$ never reaches $D$ if a digital computer with some fixed precision is used. The median nearest neighbour radius calculated for a 1,000 disk configuration after different number of shakes are listed in Table 1. To conquer this problem, a tolerance is introduced and Eq. (2) is modified as follows while using a computer:

$$R_{MNN} = (1 + \text{tolerance}) \cdot D \quad (3)$$

The choice of the tolerance will be discussed in Section 3.4.3.
Table 1: Normalized median nearest neighbour radii versus cycle number for a 1,000 disk configuration.

<table>
<thead>
<tr>
<th>cycle no.</th>
<th>$R_{MNN}/D$</th>
<th>cycle no.</th>
<th>$R_{MNN}/D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.00181</td>
<td>8</td>
<td>1.00201</td>
</tr>
<tr>
<td>1</td>
<td>1.00195</td>
<td>9</td>
<td>1.00194</td>
</tr>
<tr>
<td>2</td>
<td>1.00182</td>
<td>10</td>
<td>1.00179</td>
</tr>
<tr>
<td>3</td>
<td>1.00217</td>
<td>11</td>
<td>1.00182</td>
</tr>
<tr>
<td>4</td>
<td>1.00192</td>
<td>12</td>
<td>1.00201</td>
</tr>
<tr>
<td>5</td>
<td>1.00202</td>
<td>13</td>
<td>1.00203</td>
</tr>
<tr>
<td>6</td>
<td>1.00202</td>
<td>14</td>
<td>1.00221</td>
</tr>
<tr>
<td>7</td>
<td>1.00204</td>
<td>15</td>
<td>1.00175</td>
</tr>
</tbody>
</table>

Eqs. (2) and 3 are also characteristic of all configurations with a packing fraction greater than the packing fraction of a random close packing. Thus, a definition is given below which defines the packing fraction of a random close packing, denoted $\eta_{RCP}$:

**Definition 3** The packing fraction of a random close packing is the minimum packing fraction, such that $R_{MNN}(\eta) = D$. Explicitly,

$$
\eta_{RCP} \equiv \min\{\eta | R_{MNN}(\eta) = D\} 
$$

(4)

The problem of determining $\eta_{RCP}$ can now be reduced to one of estimating $R_{MNN}(\eta)$ for dilute packings of disks and then extrapolating to higher densities to find the point at which Eq. (4) is satisfied. This is where the Voronoi diagram comes into play. However, this will require more configurations with much more distributed packing fractions. And, it is not the topic of this thesis. So, no more discussion will be focussed on it.
3.2 Radial Distribution Function Characteristics of A Random Close Packing

The concept of randomness can be understood by introducing RDF, the radial distribution function [36]. This function, denoted by \( g \), gives the number of disk centers, \( N_c \), within an annulus (or shell in three dimensions) located at a radial distance \( r \) from a central disk of diameter \( D \), in which \( N_c \) is normalized by \( 2\pi \frac{D}{2} \). Hence, in two dimensions,

\[
g(r/D) = \frac{N_c}{2\pi(r/D)}
\]  

Clearly, \( g(r/D) = 0 \) if \( r/D < 1 \). At \( r/D = 1 \), an estimate of the average coordination number can be calculated from Eq. (5). Also,

\[
\frac{g(r/D)}{\Delta(r/D)} = \frac{N_c}{2\pi(r/D)\Delta(r/D)}
\]

tends to the number density \( \rho \), the number of disks per unit area, as \( r/D \) becomes large. If

\[
\rho = \lim_{r/D \to \infty} \frac{g(r/D)}{\Delta(r/D)}
\]

then the packing fraction, \( PF \), defined as the fraction of a unit area which is occupied by disks, can be expressed as:

\[
PF = \frac{\pi}{4} \rho D^2
\]  

In the case of a two-dimensional ordered close packing, also termed HCP (hexagonal close packing), or crystallographic packing, the RDF has peaks at some discrete values of \( r/D \) and zeroes in between. These locations can be easily determined and are listed in Table 2.
Table 2: Radial distribution peaks in a two-dimensional ordered close packing.

This feature possessed by the ordered close packing does not occur for a random close packing. Therefore a comparison of the RDF of a random packing with that of an HCP will give a measure of the "randomness" of the configuration.

In this work, attention is not focused on the RDF and plots of this function are given only to demonstrate that the generated packings are indeed random. Rather, the main thrust here is on the analysis of the packing through the use of Voronoi diagram.

3.3 Monte Carlo Simulation

The work by Rosato et. al. [42,43] has been discussed in Chapter One. The "pouring" and "shaking" of hard disks were simulated based on a Monte Carlo method. It was able to model the dynamical many-particle process, which occurs during the shaking of disks. Horizontal boundary conditions could be either non-periodic or periodic, i.e., with or without hard vertical walls. The one with periodic condition simulates closely the behaviour of an infinite system, while the non-periodic one gives a model which shows the effects of walls.

The initial coordinates of the disk centers, which form a configuration, are created by a random number generator. For any possible variation, a trial configuration is created by moving one disk at a time within a small predefined neighbourhood,
usually some fraction of the disk diameter and denoted $\delta$. A trial configuration is accepted as the new configuration based on the change in the system—inter-particle and gravitational potential energy variation $\delta E$, using the Boltzmann distribution. The details of the algorithm will not be repeated here, and may be found in [42]. An attempt to move all of the disks once is defined as a pass. In general, many passes are necessary to attain the equilibrium configuration. The simulation is run so as to allow no upward movements of disks, so that the gravitational potential is permitted only to decrease. This simulates a process whereby the initially randomly placed disks slowly settle to the bottom of the container.

To simulate a shaking, the disks are first lifted uniformly by a specific amount, called amplitude. Then, using the Monte Carlo method again, they are allowed to settle to an equilibrium state. This process is called a cycle. A cycle ceases when the change of the average system energy is less than a preselected tolerance. The disks are now packed at the bottom and a new cycle may begin. Figure 34 shows an assembly of 1,000 disks after a pouring of 60,000 passes and Figure 35 shows the same assembly after 20 cycles of shaking.

### 3.4 Analysis of the Monte Carlo Simulation Output

Some analysis of the output given by the code briefly described in the previous section will be discussed in this section. This includes (1) the radial distribution function, (2) the average count of geometric neighbours, (3) the average count of structural neighbours, (4) the angular distribution of structural neighbours, (5) the packing
fraction, and (6) the rigid wall effect. The case studies include: (1) 1,000 disks in a cell with horizontal periodic boundary conditions and (2) 1,250 disks in a cell with impenetrable vertical walls. The maximum allowed movement, \( \delta \), for each disk during a pass was equal to one sixth of the disk diameter, while the shaking amplitude for both cases was one third of the disk diameter. The details are shown in Table 3.

<table>
<thead>
<tr>
<th>case study</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>disk number</td>
<td>1,000</td>
<td>1,250</td>
</tr>
<tr>
<td>cell width</td>
<td>8 in. with periodic b.c.</td>
<td>9 in. with non-periodic b.c.</td>
</tr>
<tr>
<td>disk dia. (D)</td>
<td>0.3 in.</td>
<td>0.25 in.</td>
</tr>
<tr>
<td>pouring passes</td>
<td>60,000</td>
<td>60,000</td>
</tr>
<tr>
<td>( \delta ) in each pass</td>
<td>( 1 / 3 ) D</td>
<td>( 1 / 3 ) D</td>
</tr>
<tr>
<td>shaking amplitude</td>
<td>( 1 / 3 ) D</td>
<td>( 1 / 3 ) D</td>
</tr>
<tr>
<td>passes per shake</td>
<td>20,000</td>
<td>20,000</td>
</tr>
<tr>
<td>( \delta ) in each pass</td>
<td>( 1 / 6 ) D</td>
<td>( 1 / 6 ) D</td>
</tr>
<tr>
<td>no. of shakes</td>
<td>20</td>
<td>19</td>
</tr>
</tbody>
</table>

Table 3: Cases Studied.

3.4.1 Radial Distribution Function

The radial distribution functions, denoted by \( g \), for the case study (1), 1,000 disks with periodic boundary condition are shown in Figure 5 and 6 for the poured configuration and the one after 20 cycles of shake respectively. After shaking, the peaks become more pronounced since the packing has become more dense and less random, as depicted in Figures 34 and 35.

The value of the first peak at \( r/D = 1 \) has also increased after shaking. It is calculated as 0.654809 for the poured case and 0.761822 for the shaken one. It yields an estimate of the average coordination number as mentioned in Section 3.2. Using
Figure 5: Radial distribution function for a 1,000 disk configuration after pouring.

1,000 Disks
0 Cycle
60,000 Passes
First Peak:
r/D = 1, g(r/D) = 0.654809
Figure 6: Radial distribution function of a 1,000 disk configuration after 20 cycles of shakings.

1,000 Disks
20 Cycles
460,000 Passes
First Peak:
$r/D = 1$, $g(r/D) = 0.761822$
Eq. (5), the average coordination number estimated are:

- $2\pi g(r/D) \mid_{r/D=1} = 4.11429$, for the poured case.
- $2\pi g(r/D) \mid_{r/D=1} = 4.78667$, for the shaken case.

### 3.4.2 Average Number of Geometric Neighbours

It is well known that the average number of edges per Voronoi polygon is six [49]. That is, the average count of geometric neighbours of a disk is six, independent of the randomness and the density of the packing.

To obtain an acceptable statistical result, the number of disk in the configuration to be analyzed is supposed to be great enough so that the finite size effect, sometimes called edge effect, would be eliminated. However, to utilize the minimum space and time on computer for each study, the size of the system to be processed is restricted.

To minimize the edge effects rooted from a small configuration, an inner box adjusted within the viewing box introduced in Definition 1, was fixed. The method of fixing the inner box is described below with reference to Figure 7.

1. Define disks having an unbounded Voronoi polygon as the boundary disks. All boundary disks form the boundary of a disk configuration.

2. Find Corner1, Corner2, Corner3, and Corner4, the disks closest to the lower left, upper left, upper right, and lower right corners of the viewing box respectively. The four corners divide the boundary of the disk configuration into four divisions, namely left, top, right, and bottom boundaries.
3. Find the most inner disk on each of the four boundaries; i.e., \((x_l, y_l)\)–the right most disk on the left boundary, \((x_t, y_t)\)–the bottom most disk on the top boundary, \((x_r, y_r)\)–the left most disk on the right boundary, and \((x_b, y_b)\)–the top most disk on the bottom boundary.

4. Walk twice the maximum disk diameter further toward the center to get the constraint on the four edges of the inner box:

\[
x \geq x_l + 2D_{\text{max}}; \quad y \leq y_t - 2D_{\text{max}};
\]
\[
x \leq x_r - 2D_{\text{max}}; \quad y \geq y_b + 2D_{\text{max}}.
\]

Each and every disk in the ensemble is then checked and discarded from the analysis if it lies completely outside of the inner box. The analysis is then be performed on disks possessing no edge disk characteristics.

What is the edge disk characteristics? For instance, a disk on the edge of an ensemble would possess significantly fewer geometric neighbours on the average than the inner disks. Through the introduction of the inner box, the edge disks are taken out and the result will be closer to that of an infinite ensemble.

A geometric neighbour number distribution for the inner disks of a finite disk configuration composed of 1,000 disks, after 15 cycles of shaking, is presented in figure 8. The average count of it is calculated to be 5.99875. The small deviation from six is due to the remaining finite size effect and the machine error.
Figure 7: The fixing of an inner box within the viewing box.
Figure 8: Distribution of geometric neighbour number.

1,000 Disks (802 Inner Disks)
15 Cycles
360,000 Passes
(20,000 Pass/Cycle)
Disk Dia. = 0.5 in.
Shaking Amplitude = 0.1 in.
δ = 0.1 in.
Mean = 5.99875
3.4.3 Average Number of Structural Neighbours — Mean Coordination Number

The structural neighbours of a disk are contained in its geometric neighbours and are those disks in contact with it. The number of structural neighbours is called coordination number. It is well known that all disks in a ordered close packing have six geometric neighbours and the geometric neighbour set for each disk in such a packing is also the structural neighbour set. Therefore, the average coordination number of a ordered close packing is six, the same as the average number of geometric neighbour.

A Voronoi diagram and its Delaunay triangulation for an ordered close packing are shown in Figure 9 and 10. It can be seen that each inner Voronoi polygon has exactly six edges and each vertex in the Delaunay triangulation has six lines incident upon it.

Unlike the ordered packing the coordination number may vary from disk to disk. Thus, we can only speak of coordination numbers in a statistical mean sense for a random close packing.

For this work, the accuracy of the digital computer must be considered. Let the distance between one geometric neighbour and its center disk be $r$, and let their radii be $r_1$ and $r_2$ respectively. For the case in which the geometric neighbour is in contact with the center disk, the following is true:

$$r = r_1 + r_2$$
Figure 9: Voronoi diagram for an ordered close packing.
Figure 10: Delaunay triangulation for an ordered close packing.
However, since the equality can never be exactly obtained, a tolerance is introduced:

\[ r \leq (1 + \text{tolerance}) \cdot (r_1 + r_2) \]  \hspace{1cm} (7)

A disk is classified as a structural neighbour if Eq. 7 is satisfied.

Figure 11 and 12 give the cumulative probability of the normalized nearest neighbour radius, \( r/D \), that is, \( P(r/D \leq R) \), for the poured and shaken (with 15 cycles) configurations. In Figure 12,

\[ \lim_{x \to 1.03} P(r/D \leq R) = 1 \]

It seems to indicate that a tolerance of 0.03 would be appropriate. This value is regarded as a lower bound on the tolerance. The upper bound is chosen to be 0.05 following Bernal's [9] convention. In all of the analysis, the upper bound value was used. Figure 13 indicates that the average coordination number increases with tolerance.

Also, the structural neighbour number distributions for the 1,000 disk configuration after pouring and after 15 cycles of shaking respectively are shown in Figures 14 and 15. It is observed that that shaking increases the average coordination number. Figures 16 and 17 clearly exhibit this trend for case studies 1 and 2. This is physically reasonable since it is expected more disks would be in contact with each other after shaking.
Figure 11: Cumulative probability of the normalized nearest neighbour radius $r/D$ for 1,000 disks after pouring.
Figure 12: Cumulative probability of the normalized nearest neighbour radius $r/D$ for 1,000 disks after shaking.

1,000 Disks
15 Cycle
360,000 Passes
Disk Dia. = 0.3 in.
Shaking Amplitude = 0.1 in.
$\delta = 0.1$ In.
$R_{\text{NN}} = 0.300526$ In.
$R_{\text{NN}} = 1.001753$ Disk Dia.
Figure 13: Relationship between coordination number and tolerance.
Figure 14: Distribution of structural neighbour number after pouring.

1,000 Disks
(761 Inner Disks)
0 Cycle
60,000 Passes
Disk Dia. = 0.3 in.
δ = 0.1 in.
Mean = 4.01445
Figure 15: Distribution of structural neighbour number after shaking.
Figure 16: Average coordination number versus shaking: 1,000 disks with periodic boundary conditions.

Linear Fit Eq.: \( y = 0.04279 \times x + 3.98037 \)

Correlation = 0.9315
Figure 17: Average coordination number versus shaking: 1,250 disks with non-periodic boundary conditions.

Linear Fit Eq. \( y = 0.05005 \times + 3.93557 \)

Correlation = 0.9760
3.4.4 Angular Distribution of Structural Neighbours

As discussed in the previous section, structural neighbours are those geometric neighbours which satisfy Eq. (7). Under a selected tolerance, a disk will have a number of structural neighbours, less than or equal to six. The structural neighbours "park" on the circumference of the center disk and each one has an associated structural neighbour angle. The structural neighbour angle is defined as follows:

Definition 4 Let \( p \) and \( q \) be the centroids of a disk and one of its structural neighbour respectively. The structural neighbour angle associated with this structural neighbour is the angle measured from the positive x-direction to vector \( \overrightarrow{pq} \) in counterclockwise direction.

A disk in a random close packing, possessing \( M \) structural neighbours, will have \( M \) structural neighbour angles. It is interesting to determine the distribution of structural neighbour angles for those disks having \( M = 1, 2, 3, 4, 5, \) or 6 structural neighbours.

To accomplish this task, the disks were first categorized into groups with the same number, \( M \), of structural neighbours. The \( M \) structural neighbour angles for each disk possessing \( M \) structural neighbour were then sorted in descending order such that the first structural neighbour had the largest structural neighbour angle and the \( M^{th} \) one had the smallest structural neighbour angle.

The distributions of the first through the \( M^{th} \) angles were computed, resulting in \( M \) distributions. These distributions are presented in Figures 18 to 22 for the 1,000
Table 4: Mean values for each structural neighbour angle.

<table>
<thead>
<tr>
<th>number of structural neighbour</th>
<th>mean value for structural neighbour in sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>292.3</td>
</tr>
<tr>
<td>3</td>
<td>293.0</td>
</tr>
<tr>
<td>4</td>
<td>312.3</td>
</tr>
<tr>
<td>5</td>
<td>326.4</td>
</tr>
<tr>
<td>6</td>
<td>349.6</td>
</tr>
</tbody>
</table>

Table 5: Standard deviation for each structural neighbour angle.

<table>
<thead>
<tr>
<th>number of structural neighbour</th>
<th>standard deviation for structural neighbour in sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>40.53</td>
</tr>
<tr>
<td>3</td>
<td>45.69</td>
</tr>
<tr>
<td>4</td>
<td>24.36</td>
</tr>
<tr>
<td>5</td>
<td>28.18</td>
</tr>
<tr>
<td>6</td>
<td>20.28</td>
</tr>
</tbody>
</table>

disk configuration after 15 cycles of shakings. The mean value and standard deviation were derived from the distribution and were presented in Tables 4 and 5.

3.4.5 Packing Fraction

Knowledge of the packing fraction gives an estimate of the density of the granular mass. In addition, the use of Voronoi diagram permits the determination of the distribution of void fraction for each Voronoi polygon. Such information is of use in modelling the behavior of powdered materials. However, calculations of the void fraction distributions has not been done in this study. Rather, the packing fraction for the whole configuration has been determined for pouring and subsequent shaking of disk assemblies. The purpose of the shaking is to densify the assembly.
Figure 18: Angular distribution of each structural neighbour for disks possessing two structural neighbours.
Figure 19: Angular distribution of each structural neighbour for disks possessing three structural neighbours.
Figure 20: Angular distribution of each structural neighbour for disks possessing four structural neighbours.
Figure 21: Angular distribution of each structural neighbour for disks possessing five structural neighbours.
Figure 22: Angular distribution of each structural neighbour for disks possessing six structural neighbours.
Figures 23 and 24 show the decrease of the system energy versus shaking, while Figures 25 and 26 show the increase of the packing fraction versus shaking. In case study (1), the linear least square fit of the data in Figures 23 and 25 show that a 6.17% decrease in the system energy over 20 cycles resulted in a 6.96% increase in the packing fraction. Similarly, in case study (2), the linear least square fit of the data in Figures 24 and 26 show that a 6.13% decrease in the system energy over 20 cycles of shaking resulted in a 6.39% increase in the packing fraction. As the assemblies become more dense, the average coordination number also increases as discussed in Section 3.4.3. Figures 27 and 28 further illustrate the relationship between the packing fraction and the average coordination number. The fact that the packing fraction increases with average coordination number is in agreement with the results of D. N. Sutherland [52] although the coordination numbers presented here are greater. This may have been caused by the selection of Bernal's tolerance of 0.05 applied in three-dimensional spheres, which is an upper bound we selected in our study.

3.4.6 Rigid Wall Effect

Up to this point, the analyses were aimed at the overall characteristics of a configuration. The configuration with non-periodic boundaries, a finite configuration with constraining walls at both sides, has some interesting local characteristics in the vicinity of the wall, which are now discussed. Specifically, the variation of the packing
Figure 23: Energy variation versus shaking: 1,000 disks with periodic boundary conditions.

Linear Fit Eq. \( y = -0.00530 \times x + 1.63155 \)

Correlation = 0.9584
Figure 24: Energy variation versus shaking: 1,250 disks with non-periodic boundary conditions.

Linear Fit Eq.: $y = -0.00169 \times x + 0.52368$

Correlation = 0.9594
Figure 25: Packing fraction variation versus shaking: 1,000 disks with periodic boundary conditions.

Linear Fit Eq. \( y = 0.00292 \times x + 0.79671 \)

Correlation = 0.9713
Figure 26: Packing fraction variation versus shaking: 1250 disks with non-periodic boundary conditions.

Linear Fit Eq. $y = 0.00266 \times + 0.79071$
Correlation = 0.9790
Figure 27: Packing fraction versus average coordination number: 1,000 disks with periodic boundary conditions.
Figure 28: Packing fraction versus average coordination number: 1,250 disks with non-periodic boundary conditions.
fraction affected by walls is discussed.

Starting with an collection of 1,250 disks in a cell with hard vertical walls the poured configuration was obtained, and subsequently it was shaken for 19 cycles. To find the packing fraction variation near the wall, different PF boxes were selected and the corresponding packing fractions were calculated.

The left edge of the PF boxes in which the packing fractions were determined, denoted $PF_l$, was fixed at the left wall of the configuration cell $x = 0$. The bottom edge, denoted $PF_b$, was fixed at the bottom of the cell $y = 0$. The top edge, denoted $PF_t$, was selected so that no top boundary disk would be included. Initially, the PF box had a width equal to one disk diameter. The right edge, denoted $PF_r$, was gradually moved until it coincided with the right wall of the cell. For any PF box, the packing fraction, $\eta$, is calculated as:

$$\eta = \frac{\text{Sum of disk areas within the PF box}}{\text{PF box area}}$$  \hspace{1cm} (8)

Figure 29 shows the variation of $\eta$ with respect to $PF_r$ (normalized by the disk diameter). The packing fraction is smallest at the location of the wall and increases with the size of the PF box. It reaches the average packing fraction discussed in Section 3.4.5 while $PF_r$ moves to the right wall, which is far away from the left wall. The trend here is similar to that studied by Pillai [39] and Tory [23], in which the variation of cross sectional packing fraction of a monolayer with respect to the distance from the wall was calculated.
Figure 29: Influence of rigid walls on packing fraction for a 1,250 disk configuration.
4 Summary

The purpose of this work was to study the disk packing structure generated using a Monte Carlo technique, whereby disks slowly settle under gravity to the bottom of a container. This container of disks was then shaken, using a modification of the Monte Carlo code [42,43], and the resulting configurations were analyzed.

Two different types of boundary conditions were used: horizontal, periodic boundary conditions and rigid vertical walls. The former condition simulates a very large system by removing the restrictions imposed by hard vertical walls. In this way, the wall effects are eliminated. Imposition of hard vertical walls was done to study the effects of this type of boundary on the packing fraction. In order to analyze the configurations, an algorithm by Kurt E. Brassel and Douglas Reif [14] was chosen, modified and implemented to construct the Voronoi diagram. This permitted measurements of the distribution of coordination number, geometric and structural neighbors, distribution of structural neighbour angles, packing fraction and rigid wall effects. The Voronoi code was verified by checking that the average number of geometric neighbors, or the average number of sides of a Voronoi polygon, was equal to the mathematically proven value of six. In addition, the packing fraction for a hexagonally close-packed structure was computed as 0.906901, which agreed very well with the theoretically-computed value of 0.9068997. The structural neighbours of a disk (of diameter $D$) were defined as those disks whose centers were $(1 + \epsilon) \cdot D$ away from the specific disk. The tolerance, $\epsilon$, was chosen to be equal to Bernal’s
experimentally-determined value of 0.05. From the list of structural neighbours, the median nearest neighbor radius was computed. The cumulative probability of the median nearest neighbour radius was also computed. The upper bound value of \( r/D \), for which the probability equaled unity, was 1.03. Therefore, the tolerance value of 0.03 may be considered a lower bound. Analyses on the sensitivity of the algorithm to the tolerance was not done. However, the variation of the coordination number versus tolerance gave an indication of this sensitivity over the range of tolerance values, from 0.0 to 0.2. A continuation of this curve past a tolerance of 0.2 would have resulted in an average coordination number of six, and hence all of the geometric neighbours would be captured.

After the pouring was completed, the assembly was shaken for a number of cycles. For both the periodic and non-periodic boundary conditions, this resulted in an increase of the coordination number after 19 cycles. Linear regressions on the coordination number versus cycle number resulted in a high correlation. The values predicted from the regression equation (in the case of periodic boundary conditions) agreed well with those calculated from the first peak of the radial distribution function for the poured and shaken assemblies. The increase of coordination number as shaking proceeds is physically reasonable and is a consequence of the densification of the assembly and an increase in the average number of contacts per disk. In fact, the structure was quite hexagonally close-packed and the packing fraction increased from its poured value of approximately 0.79 to 0.85. Linear regressions on the packing
fraction versus cycle number yielded a high value of the correlation coefficient, indicating a good linear fit. A linear regression on the packing fraction against average coordination number also gave a very good linear fit.

The influence of the wall on the packing fraction was determined by using the hard vertical wall boundary conditions. A plot of the packing fraction versus normalized distance from the wall indicated that the wall had a strong influence up to approximately nine diameters away from the wall for the poured configuration. This influence extended somewhat further for the shaken assembly. The trend for both the poured and shaken assemblies were the same.

Two observations are made concerning the result of shaking the poured assembly of disks. The first of these deals with the coordination number, or the average number of structural neighbours. The simulated shaking has the effect of changing the poured, random configuration to a somewhat ordered, dense structure. The average coordination number increased linearly with the cycle and this is physically realistic since shaking should cause an increase in the disk contacts. Secondly the packing fraction showed a six percent increase over its poured value. The structure became less random and appeared to more like an hexagonal close-packed crystal. This result is apparent from the actual disk configuration as well as from the fact that the peaks of the radial distribution function became more pronounced. The location of these peaks were in very good agreement with those of a hexagonal close-packed structure. In fact, the first three peaks are indicative of short-range order.
Extensions—the following is a listing of possible extensions to this research:

1. Study of the effect of the tolerance on the average coordination number.

2. Determination of the distribution of void space within the assembly.

3. Experimentation to verify the data with regard to the poured configuration.

4. Study of the angular distribution of geometric neighbours as the assembly densifies with shaking.

5. Extensions to three-dimensional systems of spheres.
Appendix:

A Definitions and Properties Regarding Voronoi Diagram

The Voronoi polygon\(^3\), also termed *Thiessen polygon* [54] or *Dirichlet region* [17], is used to define the geometric neighbour. In any collection of points there is, about each point, a region containing all points of the plane closer to that point than to any other point. Dirichlet was first to describe these regions, hence the name Dirichlet region. It has been proven [49] that for any collection of points in the plane the Voronoi polygon has an average of six sides. If the perpendicular bisectors of all the lines connecting the point in question and all other points of the set are constructed, a set or polygons will result. The "closest" of these polygon, the one which is not truncated by any other line, is the one which satisfies the condition that it shall contain all points of plane closest to the point in question. With this understanding in mind, the following definitions quoted from D. T. Lee [30] are easier to understand:

**Definition 5** *Given an ensemble \( S \) of points \( \{p_1, p_2, \ldots, p_n\} \), the locus of points closer to \( p_i \) than to any other point is called the Voronoi polygon, denoted \( V(p_i) \), associated with \( p_i \).*

For an ensemble of \( n \) points, there are \( n \) Voronoi polygon associated with each and every point. A side of the Voronoi polygon is called a Voronoi Edge. A collection

---

\(^3\)A Voronoi polygon corresponds to a two-dimensional point. However, a three-dimensional point will have a Voronoi polyhedron correspond to it.
of the \( n \) Voronoi polygon is called the Voronoi diagram \( V(S) \).

**Definition 6** The points of the ensemble which contribute edges to the Voronoi polygon of a point are defined as the geometric neighbours of that point.

Before applying the B-R Method, some properties of the Voronoi diagram are also needed to be discussed.

**Definition 7** A region \( R \) is star-shaped with nucleus \( p \), where \( p \) is a point in \( R \), if the line segment of \( p \) and any other point in \( R \) lies completely in \( R \).

**Lemma 1** Given an ensemble of points \( \{p_1, p_2, \ldots, p_n\} \) and a point \( p \), if the nearest neighbour of \( p \) is \( p_i \), i.e., \( p \in V(p_i) \), the Voronoi polygon associated with \( p_i \), then the line segment \( \overline{pp_i} \) lies entirely in \( V(p_i) \).

Proof: Referring to Figure 30 it is easy to demonstrate that the nearest neighbour of any point on the line segment \( \overline{pp_i} \) is \( p_i \) if \( p_i \) is the nearest neighbour of \( p \). Let \( d(p, q) \) denote the distance between two points \( p \) and \( q \). Suppose there exists a point \( q, q \in \overline{pp_i} \) whose nearest neighbour is \( p_j, j \neq i \); i.e.,

\[
d(q, p_j) < d(q, p_i)
\]  

(9)

By the triangular inequality we have

\[
d(p, p_j) \leq d(p, q) + d(q, p_j)
\]  

(10)

Further,

\[
d(p, p_i) = d(p, q) + d(q, p_i)
\]  

(11)
Figure 30: Lemma 1.
We thus have the following inequality from Eq. (9) – (11) above:

\[ d(p, p_j) < d(p, p_i) \]  

(12)

which contradicts the assertion that \( p_i \) is the nearest neighbour of \( p \). Thus, for all points \( q \in \overline{pp_i} \), \( p_i \) must be the one and only nearest neighbour of it, i.e., the line segment \( \overline{pp_i} \) lies entirely in \( V(p_i) \). 

From Definition 7 and Lemma 1, the following corollary is given:

**Corollary 1** Given an ensemble of points \( \{p_1, p_2, \ldots, p_n\} \), the Voronoi polygon \( V(p_i) \) is a star-shaped polygon with nucleus \( p_i \).

The angle extended from the nucleus to any two vertices in sequence of a bounded star-shaped polygon, \( \angle AOB \) in Figure 31 for instance, must be less than \( \pi \). The Voronoi polygon is a star-shaped polygon as of Corollary 1, it is thus followed:

**Corollary 2** Let point \( O \) and point \( A \) be the nucleus and one of the vertices of a bounded Voronoi polygon. Let \( E_r(O, A) \) and \( E_l(O, A) \) be the right and left half planes divided by vector \( \overrightarrow{OA} \). Then, the next sequential vertex in clockwise direction to vertex \( A \) will lie in \( E_r(O, A) \) and the next sequential vertex in counterclockwise direction to vertex \( A \) will lie in \( E_l(O, A) \).

This is a simple, while important property of Voronoi polygon.

**Lemma 2** Given an ensemble \( S \) of points \( \{p_1, p_2, \ldots, p_n\} \), the Voronoi polygons \( V(p_i) \) and \( V(p_j) \) share an edge if and only if there exists a point \( q \) such that the circle
Figure 31: The extended angle AOB.
centered at q with radius d(q, p_i) passing through points p_i and p_j does not include any other point of S in its interior or on its boundary.

Proof: Refer to Figure 32. Suppose there exists a point q satisfying this condition; the circle Q centered at q with radius d(q, p_i) passing through points p_i and p_j does not include any other point of S. Then the point q must lie on the bisector B(p_i, p_j) perpendicularly bisecting points p_i and p_j, and both p_i and p_j are its nearest neighbours. Since q is shared by V(p_i) and V(p_j), by continuity of the distance metric we can always find a portion of B(p_i, p_j), denoted \( \overline{B(p_i, p_j)} \), containing q such that every point on the line segment \( \overline{B(p_i, p_j)} \) is shared only by V(p_i) and V(p_j). That is, some portion of B(p_i, p_j) must be shared by V(p_i) and V(p_j).

To prove the converse, let us consider any point q on the edge shared by V(p_i) and V(p_j), but q is not an endpoint of the edge. Since both p_i and p_j are the nearest neighbours of q, the circle Q centered at q with radius d(q, p_i) passes through p_i and p_j and will not contain any other point of S in its interior or on its boundary. \( \Box \)

The two endpoints of \( \overline{B(p_i, p_j)} \), if \( \overline{B(p_i, p_j)} \) is bounded, are called Voronoi points. Each Voronoi point is equidistant from the points whose associated polygon have the Voronoi point in common. Thus, if \( \overline{B(p_i, p_j)}, \overline{B(p_j, p_k)}, \) and \( \overline{B(p_k, p_i)} \) meet at a common point p, the point p is the circumcenter of the triangle \( \triangle p_i p_j p_k \). If we assume that no more than three points are cocircular, then for each Voronoi point there are three Voronoi edges incident with it.
Figure 32: Lemma 2.
Lemma 3 A point \( p \) equidistant to three points \( p_i, p_j, \) and \( p_k \) is a true Voronoi point if no other point is located within a circle centered at point \( p \) with radius the distance from point \( p \) to either one of the three points \( p_i, p_j, \) or \( p_k. \)

Now, let us consider the straight-line dual of the Voronoi diagram \( V(S) \), the Delaunay triangulation, denoted \( D(S) \). \( D(S) \) is a planar graph on the given set of points in which two points are connected if and only if their associated polygon share a Voronoi edge. \( D(S) \) partitions the plane into a number of finite triangular regions and its complementary infinite region. The edges in \( D(S) \) are called Delaunay edges. Since each Voronoi polygon \( V(p_i) \) is star-shaped with nucleus \( p_i \), the edges of the Voronoi polygon must occur in sequence, and the two corresponding Delaunay edges must appear in sorted angular order. In particular, the point \( p_i \) whose associated Voronoi polygon is unbounded will be on the boundary of \( D(S) \), and the two boundary edges incident with \( p_i \) correspond to the two unbounded Voronoi edges. The Delaunay triangulations for a set of 1,000 points is shown in Figure 33. The corresponding configuration plot and Voronoi diagram are presented in Figures 34 to 36. The Delaunay triangulation can be said to be a byproduct of the Voronoi diagram. With some modifications, it can be applied for triangular mesh generation in finite element method. Knowledge on the Delaunay triangulation helps better understanding of the Voronoi diagram.

Definition 8 Given two points \( p_i \) and \( p_j \) and a point \( p \) which lies to the left of the vector \( p_i \overrightarrow{p_j} \) and on the bisector \( B(p_i, p_j) \), the circle centered at \( p \) with radius \( d(p, p_i) \)
Figure 33: Delaunay triangulation of 1,000 disks after pouring.
MONTE CARLO SIMULATION

Periodic Boundary Conditions.
Shaking - Segregation.
CYCLE NUMBER = 0
AMPLITUDE = 0.000
Pass Number = 60000
Number of Discs = 1000
Diameter = 0.300
Diameter Ratio = 1.000
Energy = 1.647315

Figure 34: The configuration of 1,000 disks after pouring — courtesy of A. Rosato, K. Reddy, and Y. Lan.
Figure 35: The configuration of 1,000 disks after 20 cycles of shaking — courtesy of A. Rosato, K. Reddy, and Y. Lan.
Figure 36: Voronoi diagram of 1,000 disks after pouring.
passes through \( p_i \) and \( p_j \) and is called the limiting circle \( LC(p_i, p_j) \) of \( p_i \) and \( p_j \) when the radius approaches infinity. If \( p \) lies to the right of the vector \( \overrightarrow{p_i p_j} \) (to the left of \( \overrightarrow{p_j p_i} \)), its corresponding limiting circle is denoted by \( LC(p_j, p_i) \).

In \( L_2 \)-Metric [30], the limiting circle \( LC(p_i, p_j) \) and \( LC(p_j, p_i) \) coincides and is the line \( \overline{p_i p_j} \).

**Theorem 1** The boundary of the dual graph \( D(S) \) of the Voronoi diagram \( V(S) \) is the convex hull \( CH(S) \) of the set \( S \) of points.

Proof: Referring to Figure 37. Suppose \( \overline{p_i p_j} \) is an edge of the convex hull \( CH(S) \), shown in dashed line, of the set \( S \) of points. All the points of \( S \) must lie on one side of the line \( \overline{p_i p_j} \) containing \( \overline{p_i p_j} \). Suppose \( S \subset E_r(p_i, p_j) \cup \overline{p_i p_j} \), i.e., the points of \( S \) lie to the right of \( \overline{p_i p_j} \). Consider the limiting circle \( LC(p_i, p_j) \), which is the straight line \( \overline{p_i p_j} \) and hence does not contain any other point of \( S \) in its interior. From Lemma 2, the perpendicular bisector \( B(p_i, p_j) \) must contain an unbounded Voronoi edge of \( V(S) \), and so must \( \overline{p_i p_j} \) be a Delaunay edge.

Now, suppose \( \overline{p_i p_j} \) is a Delaunay edge on the boundary of \( D(S) \). Since it corresponds to an unbounded Voronoi edge on \( B(p_i, p_j) \), the limiting circle centered at the unbounded end of \( B(p_i, p_j) \) does not contain any other point in its interior. That is, all the points lie on one side of the line containing \( \overline{p_i p_j} \). Therefore \( \overline{p_i p_j} \) is an edge on the convex hull \( CH(S) \). This completes the proof. \( \ddagger \)

In other words, \( \overline{p_i p_j} \) is a boundary edge of \( D(S) \) if and only if one of the limiting circles of \( p_i \) and \( p_j \) does not contain any other point of \( S \) in its interior. Each boundary
Figure 37: Theorem 1.
edge of \( D(S) \) has an unbounded Voronoi edge corresponding to it. This can be extended to another statement:

The number of points on the convex hull of the ensemble \( S \) is the same as the number of unbounded Voronoi polygon in the Voronoi diagram \( V(S) \).

**Theorem 2** Given an set \( S \) of points \( \{p_1, p_2, \cdots, p_n\} \), let \( D_i(S) \) denote the subset of points of \( S \) which are connected to \( p_i \) in the Delaunay triangulation. Then \( D_i(S) \) contains the nearest neighbour of \( p_i \), i.e., there exists a point \( p_j \in D_i(S) \) such that

\[
d(p_i, p_j) = \min_{q \in S-p_i} d(q, p_i)
\]

Proof: Referring to Figure 38, \( D_i(S) \) is composed of \( \{p_r, p_s, p_t, p_u, p_v\} \). Suppose the nearest neighbours of \( p_i \) are not contained in \( D_i(S) \). Let \( p_k \) be the nearest neighbour of \( p_i \). Then \( d(p_k, p_i) < d(q, p_i) \) for all \( q \in D_i(S) \).

Consider the line segment \( \overline{p_k p_i} \); it must intersect some Voronoi edge of the Voronoi polygon, say point \( z \) on \( \overline{B(p_i, p_s)} \). By definition, \( p_s \in D_i(S) \) and

\[
d(z, p_s) = d(z, p_i) \tag{13}
\]

Also, the circle centered at \( z \) with radius \( d(z, p_i) \) does not contain any other point of \( S \). So,

\[
d(z, p_i) < d(z, p_k) \tag{14}
\]

From Eq. (13) and (14) and the triangular inequality:

\[
d(p_s, p_i) \leq d(z, p_s) + d(z, p_i) < d(z, p_k) + d(z, p_i) = d(p_k, p_i) \tag{15}
\]
This is a contradiction of the assumption. Therefore $D_i(S)$ contains the nearest neighbour of $p_i$. \[ \]
Figure 38: Theorem 2.
B Algorithm Finding Next Clockwise Geometric Neighbour

start

1. Find first potential neighbour $P$;
   Compute vertex $V$ and radius $R$;
   $V, R = f(A, X, P)$
2. Find first point $P'$ to be subjected to circle test

$P'$ within abscissa range of circle $(V, R)$

Yes

End; $P$ is the new neighbour
$V$ is the new vertex

No

Find next point $P'$ in sequence

$P'$ in right half plane $E_R (A, X)$

Yes

No

$P'$ location within circle $(V, R)$

Yes

Declare $P'$ as potential neighbour
$P$; recompute $V, R$
C Strategy on Completing the Voronoi Diagram

Start

Sort point set along x-axis

Select a first centroid and its first neighbour

Add dummy points to point set

Find next neighbour clockwise; compute new vertex

Bookkeeping: store new vertex coordinate and record neighbourhood relationship

Voronoi polygon complete?

No

Select next centroid to be processed

Yes

All Voronoi polygons complete?

No

Write output files

End
**D Program disk.pas**

```pascal
(* *********************************************************************** *)
(*                                                                         *)
(*                                                                           *)
(*      INTRODUCTION                                                        *)
(*                                                                           *)
(*                                                                         *)
(*   THE CODE ANALYZES AN ENSEMBLE OF DISKS, EITHER IN THE SAME SIZE OR *)
(*   IN DIFFERENT SIZES, WITH OR WITHOUT PERIODICAL BOUNDARY CONDITION IN *)
(*   THE HORIZONTAL DIRECTION.                                              *)
(*                                                                           *)
(*   FIRST, A LINKED LIST 'RANDOM' IS CREATED. IT CONTAINS THE INDEX AND *)
(*   THE DIAMETER OF THE DISK, AND THE COORDINATES OF THE CENTROID. BEFORE *)
(*   PROCESSING, WE SORT THE LINKED LIST IN LEXICOGRAPHICAL ORDER AND NAME *)
(*   THE SORTED LINKED LIST 'SORT'. A SORTED TWO-DIMENSIONAL COORDINATES IN *)
(*   LEXICOGRAPHICAL ORDER IS INTERPRETED AS: P1 (X1, Y1) IS SAID TO BE *)
(*   LEXICOGRAPHICALLY LESS THAN P2 (X2, Y2), IF AND ONLY IF X1 < X2, OR *)
(*   X1 = X2 AND Y1 < Y2.                                                   *)
(*                                                                           *)
(*   SECOND, THE VORONOI DIAGRAM IS GENERATED ACCORDING TO THE B-R *)
(*   METHOD, WHICH IS CONSTRUCTED BY KURT E. BRASSEL AND DOUGLAS REIF. THE *)
(*   GEOMETRIC NEIGHBOURS ARE THUS FOUND. IN VIEWING THE VORONOI DIAGRAM, WE *)
(*   FIX A BOX WHICH IS EXPRESSED EXPLICITLY:                              *)
(*                                                                           *)
(*   BOX = (X, Y) | XMIN - BOX_C * DMAX <= X <= XMAX + BOX_C * DMAX;       *)
(*     YMIN - BOX_C * DMAX <= Y <= YMAX + BOX_C * DMAX                    *)
(*                                                                           *)
(*   WHERE XMIN, XMAX, YMIN AND YMAX ARE THE EXTREMUM COORDINATES OF *)
(*   CENTROIDS OF THE DISKS, DMAX IS THE MAXIMUM DIAMETER OF THE DISKS, AND *)
(*   BOX_C IS A CONSTANT WHICH DETERMINES THE SIZE OF THE BOX.             *)
(*                                                                           *)
(*   FROM THE GEOMETRIC NEIGHBOURS WE CHOOSE THE NEAREST NEIGHBOUR AND *)
(*   COMPUTE MEDIAN NEAREST NEIGHBOUR RADIUS. FOR DISKS OF SAME SIZE, IF THE *)
(*   MEDIAN NEAREST NEIGHBOUR RADIUS IS GREATER THEN THE DIAMETER OF THE *)
(*   DISKS (OR (1 + TOLERANCE) * DIAMETER), THE ENSEMBLE IS TOO THIN TO BE A *)
(*   RANDOM CLOSE PACKING. NO FURTHER ANALYSIS IS GOING TO BE MADE IN THIS *)
(*   CASE OTHERWISE, WE PROCEED AND PERFORM THE FOLLOWING ANALYSIS.         *)
(*                                                                           *)
(*   TO BEGIN WITH, WE FIX ANOTHER INNER BOX, AND DELETE ALL THOSE DISKS *)
(*   COMPLETELY OUT OF THE INNER BOX. THE REMAINING LINKED LIST, WHICH IS *)
(*   CALLED 'INNER', CONTAINS ONLY DISKS IN OR ON THE BOUNDARY OF THE INNER *)
(*   BOX AND WITH BOUNDED VORONOI POLYGONS. THE STATICS WILL BE MADE ON *)
(*   THESE DISKS.                                                          *)
(*                                                                           *)
(*   THE STRUCTURAL NEIGHBOURS ARE PICKED OUT FROM THE GEOMETRIC *)
(*   NEIGHBOURS. AGAIN, THE SAME TOLERANCE FOR DETERMINEING THE RANDOM CLOSE *)
(*   PACKING IS APPLIED TO DECIDE WHETHER A GEOMETRIC NEIGHBOUR IS A *)
(*   STRUCTURAL NEIGHBOUR OR NOT. A TOLERANCE 0.05 IS SELECTED IN OUR CASE, *)
(*   WHICH WAS THE TOLERANCE J. D. BERNAL USED IN HIS 3-D MODEL.            *)
(*                                                                           *)
(*   ANGLES FROM THE POSITIVE X-DIRECTION TO A LINE FROM THE CENTROID OF *)
```
THE CENTER DISK TO THE CENTROID OF EACH STRUCTURAL NEIGHBOUR ARE CALCULATED. THE STATISTICS IS THEN CARRIED OUT, WHICH INCLUDES:

1. THE MEAN VALUE AND DISTRIBUTION OF THE NUMBER OF GEOMETRIC NEIGHBOUR.
2. THE MEAN VALUE AND DISTRIBUTION OF THE NUMBER OF STRUCTURAL NEIGHBOUR.
3. CATEGORISE INNER DISKS WITH THE SAME NUMBER OF STRUCTURAL NEIGHBOURS IN GROUPS. THEN, FOR EACH GROUP, FIND:
4. PACKING FRACTION, WHICH IS DEFINED AS:

   \[
   \text{DISK AREA WITHIN THE INNER BOX} \quad \frac{\text{\eta}}{\text{INNER BOX AREA}}
   \]

IF THE CONFIGURATION IS CONSTRAINED BY TWO VERTICAL WALLS, THE RIGID WALL EFFECT MAY BE CALCULATED IF DESIRED. THIS CONCLUDES THE CODE.

INPUT AND OUTPUT

TO RUN THE CODE, A INPUT FILE DISKIN.DAT IS FIRST CREATED. THE DATA PROVIDED IN DISKIN.DAT ARE LISTED BELOW:

LINE # DATA

1 # OF DISKS
2 BOOLEAN VARIABLE IF ALL DISKS ARE OF SAME SIZE
3 BOOLEAN VARIABLE IF GIVEN ENSEMBLE IS KNOWN TO BE A RANDOM CLOSE PACKING
4 DISTANCE BETWEEN VERTICAL PERIODICAL BOUNDARIES OR WALLS
5 BOOLEAN VARIABLE IF THE PERIODICAL BOUNDARY CONDITION IS TO BE IMPLEMENTED
6 BOOLEAN VARIABLE IF THE WALL EFFECT IS TO BE CALCULATED
7 HEIGHT OF THE INNER BOX AND WIDTH OF THE INNER BOX
8 BOOLEAN VARIABLE IF THE BOX FOR CALCULATING PACKING FRACTION IS TO BE DEFINED DIFFERENT FROM THE INNER BOX
8' IF B IS TRUE, GIVE PF_L, PF_R, PF_B, PF_T, SUCH THAT X = PF_L, X = PF_R, Y = PF_B, AND Y = PF_T CONSTITUTE THE BOX FOR CALCULATING PACKING FRACTION
9 TOLERANCE FOR DETERMINING STRUCTURAL NEIGHBOUR
10 NUMBER OF ANGULAR DIVISION
10 TO EOF INDEX, X-COORDINATE, Y-COORDINATE, DIAMETER OF A DISK

THE CODE THEN GIVES SIX OUTPUT FILES: NBRFILE, VTXFILE, BOX, SNBRFILE, DELAUNAY, AND TABLE AFTER BEING RUN. EACH OUTPUT FILE CONTAINS THE FOLLOWING INFORMATIONS:
(*) 1) NBRFILE: 'NUM_LOOP' (NUMBER OF DISKS TO BE PROCESSED AFTER THE *)
     IMPLEMENTATION OF PERIODICAL BOUNDARY CONDITION; SAME AS *
     NUMBER OF DISK IF PERIODICAL BOUNDARY CONDITION NOT *
     REQUIRED) OF LINKED LIST OF NEIGHBOURS; THE FIRST NODE *
     IN THE LIST BE THE CENTER DISK. *

(*) 2) VTXFILE: 'NUM_LOOP' OF LINKED LIST OF VERTICES. THIS FILE WILL *
     BE USED AS THE INPUT FILE OF THE PROGRAM *
     'VORONOIFORME.FOR', WHICH TRANSFORM 'VTXFILE' INTO THE *
     FORMAT HP7580_A PLOTTER CAN READ. *

(*) 3) BOX: GIVES THE COORDINATES OF THE VIEWING BOX AND THE INNER BOX, *
     'NUM_LOOP', AND 'NUM_IN' (NUMBER OF DISKS IN OR ON THE INNER *
     BOX), AND ERROR MESSAGES IF ANY, ETC. *

(*) 4) SNBRFILE: 'NUM_IN' OF LINKED LIST OF STRUCTURAL NEIGHBOURS FOR *
     INNER DISKS; THE FIRST NODE BE THE CENTER DISK. *

(*) 5) DELAUNAY: END POINTS OF EDGES OF DELAUNAY TRINAGULATION. THIS *
     FILE WILL BE USED AS THE INPUT FILE OF THE PROGRAM *
     'DELAUNAYFILE.FOR', WHICH TRANSFORM 'DELAUNAY.DAT' INTO *
     THE FORMAT HP7580_A PLOTTER CAN READ. *

(*) 6) TABLE: GIVES THE FOLLOWING INFORMATION FOR EACH INNER DISK: *
     A) THE NUMBER OF GEOMETRIC NEIGHBOUR AND STRUCTURAL NEIGHBOUR. *
     B) THE INDEX OF EACH GEOMETRIC NEIGHBOUR AND STRUCTURAL NEIGHBOUR. *
     C) THE ANGLE ASSOCIATED WITH EACH STRUCTURAL NEIGHBOUR. *
     D) THE MEAN VALUE OF THE NUMBER OF GEOMETRIC NEIGHBOUR AND *
        STRUCTURAL NEIGHBOUR. *
     E) THE CUMULATIVE PROBABILITY OF THE NEAREST NEIGHBOUR RADIUS. *
     F) THE DISTRIBUTION OF THE NUMBER OF GEOMETRIC NEIGHBOUR AND *
        STRUCTURAL NEIGHBOUR. *
     G) FOR EACH CATEGORIZED GROUP WITH THE SAME NUMBER OF STRUCTURAL *
        NEIGHBOURS, *
        a) THE MEAN VALUE OF THE ANGLE ASSOCIATED WITH THE FIRST, THE *
        SECOND, ..., DOWN TO THE LAST STRUCTURAL NEIGHBOUR. *
        b) THE STANDARD DEVIATION OF THE ANGLE ASSOCIATED WITH THE *
        FIRST, THE SECOND, ..., DOWN TO THE LAST STRUCTURAL *
        NEIGHBOUR. *
        c) THE DISTRIBUTION OF THE ANGLE ASSOCIATED WITH THE FIRST, THE *
        SECOND, ..., DOWN TO THE LAST STRUCTURAL NEIGHBOUR. *
     H) THE PACKING FRACTION. *

(*) AND IF WALL EFFECT IS TO BE FOUND: *
(*) I) PACKING FRACTION VARIATION. *

************************************************************************************************************
(*)
(*) Nomenclature
(*)
(*)
(*)

NUM_DISK : NUMBER OF DISKS TO BE HANDELED.
NUM_LOOP : NUMBER OF DISKS TO BE HANDELED AFTER THE IMPLEMENTATION
OF THE PERIODICAL BOUNDARY CONDITION.
NUM_IN : NUMBER OF DISKS IN OR ON THE BOUNDARY OF THE INNER BOX;
NUMBER OF INNER DISKS.
NUM_ANG_DIV : NUMBER OF ANGULAR DIVISION FOR THE STATICS ON THE ANGULAR
DISTRIBUTION.
2 : DUMMY TAIL OF A LINKED LIST.
(* RANDOM : RANDOM LINKED LIST OF DISKS. *)
(* SORT : SORTED LINKED LIST OF DISKS. *)
(* INNER : LINKED LIST OF DISKS IN OR ON THE INNER BOX. *)
(* LAST : MOVING POINTER IN THE LINKED LIST. *)
(* NBR [K] : ELEMENT OF ARRAY OF LINKED LIST OF GEOMETRIC NEIGHBOURS, *)
  (* WITH THE CENTER DISK AS THE LIST HEAD. *)
(* NBR_LAST [K] : MOVING POINTER IN THE LINKED LIST 'NBR [K]'. *)
(* GNBR [K] : ELEMENT OF SORTED ARRAY OF LINKED LIST OF GEOMETRIC *)
  (* NEIGHBOURS. *)
(* INNERG [K] : ELEMENT OF ARRAY OF LINKED LIST OF GEOMETRICAL NEIGHBOURS *)
  (* FOR INNER DISKS. *)
(* SNBR [K] : ELEMENT OF SORTED ARRAY OF LINKED LIST OF STRUCTURAL *)
  (* NEIGHBOURS. *)
(* SNBR_LAST [K] : MOVING POINTER IN THE LINKED LIST 'SNBR [K]'. *)
(* INNERS [K] : ELEMENT OF ARRAY OF LINKED LIST OF STRUCTRUAL NEIGHBOURS *)
  (* FOR INNER DISKS. *)
(* NEW_NBR : NEWLY FOUND (POTENTIAL) NEIGHBOUR. *)
(* NBR_TEMP : TEMPERARY MEMORY LOCATION OF 'NEW_NBR' TO BE LINKED TO *)
  (* THE LINKED LIST NBR [K]’s.*)
(* VTX [K] : ARRAY OF LINKED LIST OF VORONOI POINTS WITH THE CENTER *)
  (* DISK AS THE LIST HEAD. *)
(* VTX_LAST [K] : MOVING POINTER IN THE LINKED LIST 'VTX [K]'. *)
(* NEW_VTX : NEWLY FOUND (POTENTIAL) VORONOI POINT. *)
(* VTX_TEMP : TEMPERARY MEMORY LOCATION OF 'NEW_VTX' TO BE LINKED TO *)
  (* THE LINKED LIST VTX [K]’s.*)
(* NVTX [K] : ELEMENT OF SORTED ARRAY OF LINKED LIST OF VERTICES. *)
(* INNERV [K] : ELEMENT OF ARRAY OF LINKED LIST OF VERTICES FOR INNER *)
  (* DISKS. *)
(* INDEX : THE UNIQUE INTEGER REPRESENTING A CERTAIN DISK. *)
(* X, Y : THE COORDINATE OF A CENTROID OF A DISK. *)
(* DIA : THE DIAMETER OF A DISK. *)
(* ANGLE : THE ANGLE ASSOCIATED WITH A STRUCTURAL NEIGHBOUR. *)
(* XMAX : MAXIMUM X COORDINATE OF THE CENTROIDS OF THE DISKS HANDLED.*)
(* XMIN : MINIMUM X COORDINATE OF THE CENTROIDS OF THE DISKS HANDLED.*)
(* YMAX : MAXIMUM Y COORDINATE OF THE CENTROIDS OF THE DISKS HANDLED.*)
(* YMIN : MINIMUM Y COORDINATE OF THE CENTROIDS OF THE DISKS HANDLED.*)
(* DMAX : THE MAXIMUM DIAMETER OF THE DISKS. *)
(* WIDE : THE DISTANCE BETWEEN TWO VERTICAL PERIODICAL BOUNDARIES. *)
(* HEIGHT : THE HEIGHT OF THE INNER BOX. *)
(* WIDTH : THE WIDTH OF THE INNER BOX. *)
(* DUMMY1, 2 : COORDINATES OF THE DUMMY POINTS WITH RESPECT TO THE *)
  (* 3, 4 CENTROID BEING PROCESSED. *)
(* CORNER1, 2 : DISKS NEAREST TO FOUR CORNERS OF THE BOX RESPECTIVELY. *)
(* 3, 4 *)
(* FIRST, SECOND: THE FIRST AND THE SECOND DUMMY POINTS IN CLOCKWISE *)
  (* DIRECTION WITH RESPECT TO THE LINE FROM THE CENTROID TO *)
  (* THE LAST NBR FOUND. *)
(* NBR_FIRST : THE FIRST GEOMETRIC NEIGHBOUR IN A LINKED LIST NBR [K]. *)
(* AVG_GN : THE MEAN VALUE OF THE NUMBER OF GEOMETRIC NEIGHBOUR. *)
(* AVG_SN : THE MEAN VALUE OF THE NUMBER OF STRUCTURAL NEIGHBOUR. *)
(* NUM_GN [K] : ELEMENT OF ARRAY OF INTEGER OF NUMBER OF GEOMETRIC *)
  (* NEIGHBOUR FOR THE Kth INNER DISK. *)
(* NUM_SN [K] : ELEMENT OF ARRAY OF INTEGER OF NUMBER OF STRUCTURAL *)
NEIGHBOUR FOR THE Kth INNER DISK.

NUM_DST_GN(K): ELEMENT OF ARRAY OF INTEGER OF NUMBER OF DISKS WITH K
GEOMETRICAL NEIGHBOURS.

NUM_DST_SN(K): ELEMENT OF ARRAY OF INTEGER OF NUMBER OF DISKS WITH K
STRUCTURAL NEIGHBOURS.

NRR [K]: ELEMENT OF ARRAY OF REAL OF NEAREST NEIGHBOUR RADIUS OF
THE Kth DISK.

GROUP_SN [K, J]: THE Jth ELEMENT OF ARRAY OF LINKED LIST OF STRUCTURAL
NEIGHBOUR FOR DISKS WITH K STRUCTURAL NEIGHBOURS.

GROUP_SUM [K]: ELEMENT OF ARRAY OF INTEGER OF NUMBER OF DISKS WITH K
STRUCTURAL NEIGHBOURS.

NMIN, NMAX: MINIMUM AND MAXIMUM NUMBER OF GEOMETRIC NEIGHBOUR A DISK
IN THE GIVEN ENSEMBLE HAS.

SMIN, SMAX: MINIMUM AND MAXIMUM NUMBER OF STRUCTURAL NEIGHBOUR A DISK
IN THE GIVEN ENSEMBLE HAS.

ANG_DST_SN [K, J, I]: ELEMENT OF ARRAY OF INTEGER OF NUMBER OF DISKS
WITH THE Jth STRUCTURAL NEIGHBOUR OF IT LYING IN
THE Ith ANGULAR DIVISION. THE STATICS IS
PERFORMED FOR EACH CATAGORISED GROUP OF DISKS
CONTAINING THE SAME NUMBER, K, OF STRUCTURAL
NEIGHBOURS.

MEAN_NRR: MEAN NEAREST NEIGHBOUR RADIUS.

MEDIAN_NRR: MEDIAN NEAREST NEIGHBOUR RADIUS.

CP_NRR [K]: ELEMENT OF ARRAY OF REAL OF CUMULATIVE PROBABILITY FOR
THE RADIAL DISTRIBUTION OF NEAREST NEIGHBOUR RADIUS.

ETA: THE PACKING FRACTION.

PLUS: BOOLEAN VARIABLE, WHICH IS TRUE IF
1) THE NEWLY FOUND NBR IS NOT A DUMMY POINT.
2) THE NEWLY FOUND NBR IS NOT THE FIRST NBR.
3) THERE IS STILL NOT A LINKED LIST OF NBR'S AND VTX'S
FORMED CORRESPONDING TO THE NEWLY FOUND NBR.
AND, WHICH IS UNCHANGED IF THE NEWLY FOUND NBR IS THE
FIRST NBR.

ADD: BOOLEAN VARIABLE, WHICH IS TRUE IF THE PREVIOUS NEWLY
FOUND NBR (TO THE CENTROID THEN UNDER PROCESS) PRODUCES A
TRUE 'PLUS'.

EQ_DISKS: BOOLEAN VARIABLE, WHICH IS TRUE IF ALL DISKS IN THE GIVEN
ENSEMBLE ARE OF EQUAL SIZE.

PERIODICAL_REQUIRED: BOOLEAN VARIABLE, WHICH IS TRUE IF THE PERIODICAL
BOUNDARY CONDITION IS TO BE IMPLEMENTED.

KNOWN_TO_BE_RCP: BOOLEAN VARIABLE, WHICH IS TRUE IF IT IS KNOWN THE
GIVEN DISK ENSEMBLE IS A RANDOM CLOSE PACKING.

DEFINE_PF_BOX: BOOLEAN VARIABLE, WHICH IS TRUE IF THE BOX FOR
CALCULATING THE PACKING IS TO BE DEFINED DIFFERENT FROM
THE INNER BOX.

FIND_WALL_EFFECT: BOOLEAN VARIABLE, WHICH IS TRUE IF THE WALL EFFECT IS
TO BE FOUND.

AUTHOR: JIRYIH TSAUR
MECHANICAL ENGINEERING DEPARTMENT
NEW JERSEY INSTITUTE OF TECHNOLOGY

********************************************************************
PROGRAM DISK (INPUT, OUTPUT, DISKIN, NBRFILE, vtxfile, BOX, SNBRFILE, TABLE, DELAUNAY);

LABEL 100;
CONST PI = 3.141592654;
N = 1250;
SUB_L = ROUND (0.3 * N);
SUB_R = ROUND (1.3 * N);
BOX_C = 1;

TYPE INFO = RECORD
  INDEX: INTEGER;
  X, Y, DIA, ANGLE: REAL
END;

LIST = *NODE;
NODE = RECORD
  DATA: INFO;
  NEXT: LIST
END;

DOUBLE_ARRAY = ARRAY [SUB_L..SUB_R] OF LIST;
LIST_ARRAY = ARRAY [1..N] OF LIST;
GROUP = ARRAY [1..6,1..N] OF LIST;
SMALL_ARRAY = ARRAY [0..10] OF INTEGER;
SMALL_REAL_ARRAY = ARRAY [0..100] OF REAL;
NUM_ARRAY = ARRAY [0..SUB_R] OF INTEGER;
REAL_ARRAY = ARRAY [SUB_L..SUB_R] OF REAL;
ARRAY_3D = ARRAY [1..6,1..6,0..SUB_R] OF INTEGER;

VAR STRING30: PACKED ARRAY [1..30] OF CHAR;

Z, RANDOM, SORT, SORT_COPY, LAST, NEW_NBR, NBR_TEMP, NEW_VTX, VTX_TEMP,
INNER, PF_INNER: LIST;

NBR, NBR_LAST, GNBR, VTX, VTX_LAST: DOUBLE_ARRAY;

SNBR, SNBR_LAST, INNERG, INNERV, INNERS: LIST_ARRAY;

GROUP_SN: GROUP;

GROUP_SUM, NUM_DST_GN, NUM_DST_SN, DST: SMALL_ARRAY;

NNR: REAL_ARRAY;

CP_NNR: SMALL_REAL_ARRAY;

EXPECT_ANGLE, STD_DVA: ARRAY [1..6,1..6] OF REAL;

ANG_DST_SN, DST_SN: ARRAY_3D;

DUMMY1, DUMMY2, DUMMY3, DUMMY4,
NBR_FIRST, FIRST, SECOND, VERTEX_TEMP,
CORNER1, CORNER2, CORNER3, CORNER4: INFO;

INDEX, NUM_DISK, NUM_IN, NUM_LOC1,
NMIN, NMAX, SMIN, SMAX, NUM_RAD_DIV, NUM ANG_DIV,
I, J, K, L: INTEGER;

XC, YC, XC_TEMP, YC_TEMP, X1, Y1, X2, Y2, X3, Y3, X4, Y4, XL, XR,
XMAX, XMIN, YMAX, YMIN,
R, NEW_R, R_TEMP, DMA, AVG_GN, AVG_SN, DIA1, DIA2, MIN, DIVISION,
ALFA, BETA, GAMMA, AREA, ETA, SUM, SUMS, MEAN_NNR, MEDIAN_NNR,
LEFT, RIGHT, BOTTOM, TOP, PF_L, PF_R, PF_B, PF_T,
TOLERANCE, WIDE, HEIGHT, WIDTH: REAL;

NUM_GN, NUM_SN: NUM_ARRAY;

STOP, ADD, PLUS, EQ_DISKS, PERIODICAL_REQUIRED, KNOWN_TO_BE_RCP,
DEFINE_PF_BOX, FIND_WALL_EFFECT: BOOLEAN;

DISKIN, NBRFILE, VTXFILE, BOX, SNBRFILE, TABLE, DELAUNAY: TEXT;

 DISCLAIMS Fowler's FORTRAN code. The text is a part of a FORTRAN program.

FUNCTION NODE_CREATE: LIST;
VAR CENTROID: LIST;
BEGIN
  NEW (CENTROID);
  WITH CENTROID^.DATA DO 
    BEGIN
    READLN (DISKIN, INDEX, X, Y, DIA);
    CENTROID^.NEXT := Z
    END;  (END WITH CENTROID^.DATA)
  NODE_CREATE := CENTROID
END;  (END FUNCTION NODE_CREATE)

FUNCTION LIST_CREATE: LIST;
VAR LISTHEAD, NEWNODE: LIST;
BEGIN
  NEW (LISTHEAD);
  LISTHEAD^.NEXT := Z;
  LAST := LISTHEAD;
  WHILE NOT EOF (DISKIN) DO
    BEGIN
    NEWNODE := NODE_CREATE;
    LAST^.NEXT := NEWNODE;
    LAST := LAST^.NEXT
    END;  (END WHILE NOT EOF (DISKIN))
  LIST_CREATE := LISTHEAD^.NEXT
END;  (END FUNCTION LIST_CREATE)
(* MERGE TWO SORTED LIST TO FORM ONE SORTED LIST, SUCH THAT *)
(* EITHER 1) LIST^.DATA.X < LIST^.NEXT.DATA.X *)
(* OR 2) LIST^.DATA.X = LIST^.NEXT.DATA.X AND *)
(* LIST^.DATA.Y < LIST^.NEXT.DATA.Y *)
(*----------------------------------------------------------------*)

FUNCTION LEXICO_MERGE (A, B: LIST): LIST;
VAR LISTHEAD: LIST;
BEGIN
  NEW (LISTHEAD);
  LISTHEAD^.NEXT := Z;
  LAST := LISTHEAD;
  WHILE LAST <> Z DO
    IF A^.DATA.X < B^.DATA.X THEN
      BEGIN LAST^.NEXT := A; LAST := LAST^.NEXT; A := A^.NEXT END
    ELSE IF B^.DATA.X < A^.DATA.X THEN
      BEGIN LAST^.NEXT := B; LAST := LAST^.NEXT; B := B^.NEXT END
    ELSE IF A^.DATA.Y < B^.DATA.Y THEN
      BEGIN LAST^.NEXT := A; LAST := LAST^.NEXT; A := A^.NEXT END
    ELSE
      BEGIN LAST^.NEXT := B; LAST := LAST^.NEXT; B := B^.NEXT END;
    LEXICO_MERGE := LISTHEAD^.NEXT
  END; (* END FUNCTION LEXICO_MERGE *)

(*----------------------------------------------------------------*)
(* SORT A LIST BY MERGE *)
(*----------------------------------------------------------------*)

FUNCTION MERGESORT (A: LIST;
  NUM_DISK: INTEGER): LIST;
VAR B, C, LEXICO_B, LEXICO_C: LIST;
BEGIN
  IF A^.NEXT = Z THEN
    MERGESORT := A
  ELSE
    BEGIN
      B := A;
      FOR K := Z TO ( NUM_DISK DIV 2 ) DO A := A^.NEXT;
      C := A^.NEXT; A^.NEXT := Z;
      IF B^.NEXT <> Z THEN
        LEXICO_B := MERGESORT ( B, ( NUM_DISK DIV 2 ) )
      ELSE
        LEXICO_B := B;
      IF C^.NEXT <> Z THEN
        LEXICO_C := MERGESORT ( C, NUM_DISK - ( NUM_DISK DIV 2 ) )
      ELSE
        LEXICO_C := C;
      MERGESORT := LEXICO_MERGE (LEXICO_B, LEXICO_C)
    END; (* END ELSE A^.NEXT <> Z *)
  END; (* END FUNCTION MERGESORT *)
FUNCTION NEAREST (XC, YC: REAL;
  SORT: LIST): INFO;
BEGIN
  LAST := SORT;
  NEW_NBR^.DATA := LAST^.DATA;
  WITH NEW_NBR^.DATA DO
  BEGIN
    X1 := X; Y1 := Y
    END;  (END WITH NEW_NBR^.DATA)
  R := SORT ( SQRT ( X1 - XC ) + SQRT ( Y1 - YC ) );
  REPEAT
    STOP := TRUE;
    XL := XC - R;
    XR := XC + R;
    LAST := LAST^.NEXT;
    WITH LAST^.DATA DO
    BEGIN
      X2 := X; Y2 := Y
      END;  (END WITH LAST^.DATA)
    NEW_R := SORT ( SQRT ( X2 - XC ) + SQRT ( Y2 - YC ) );
    IF ( XL <= X2 ) AND ( X2 <= XR ) THEN
      BEGIN
        STOP := FALSE;
        IF NEW_R < R THEN
          BEGIN
            NEW_NBR^.DATA := LAST^.DATA;
            X1 := X2;
            Y1 := Y2;
            R := NEW_R
            END  (END IF NEW_R <R)
          END  (END IF (XL <= X2) AND (X2 <= XR))
  UNTIL
  STOP;
  NEAREST := NEW_NBR^.DATA
  END;  (END FUNCTION NEAREST)
PROCEDURE DUMMY_CREATE (XC, YC, XMAX, XMIN, YMAX, YMIN: REAL;
  VAR DUMMY1, DUMMY2, DUMMY3, DUMMY4: INFO);
BEGIN
  WITH DUMMY1 DO
    BEGIN INDEX := 0; X := 2 * XMIN - XC; Y := YC END;
  WITH DUMMY2 DO
    BEGIN INDEX := 0; X := XC; Y := 2 * YMAX - YC END;
  WITH DUMMY3 DO
    BEGIN INDEX := 0; X := 2 * XMAX - XC; Y := YC END;
  WITH DUMMY4 DO
    BEGIN INDEX := 0; X := XC; Y := 2 * YMIN - YC END
END;  (END PROCEDURE DUMMY_CREATE)
(%*******************************************************************************
(* FIND 'FIRST' AND 'SECOND', WHICH ARE THE FIRST AND THE SECOND *)
(* DUMMY POINTS IN CLOCKWISE DIRECTION WITH RESPECT TO THE LINE *)
(* FROM THE CENTROID BEING PROCESSED TO THE PREVIOUS NBR *)
*******************************************************************************%)

PROCEDURE DUMMY_FND (X1, Y1, X2, Y2: REAL;
                        DUMMY1, DUMMY2, DUMMY3, DUMMY4: INFO;
                        VAR FIRST, SECOND: INFO); BEGIN
  IF (X2 <= X1) AND (Y2 < Y1) THEN
    BEGIN FIRST := DUMMY1; SECOND := DUMMY2 END;
  IF (X2 < X1) AND (Y2 >= Y1) THEN
    BEGIN FIRST := DUMMY2; SECOND := DUMMY3 END;
  IF (X2 >= X1) AND (Y2 > Y1) THEN
    BEGIN FIRST := DUMMY3; SECOND := DUMMY4 END;
  IF (X2 > X1) AND (Y2 <= Y1) THEN
    BEGIN FIRST := DUMMY4; SECOND := DUMMY1 END
END;  (END PROCEDURE DUMMY_FND)

(********************************************************************************
(* FIND THE POINT EQUIDISTANT TO (X1, Y1), (X2, Y2), AND (X3, Y3) *)
*******************************************************************************%)

FUNCTION VX_FND (X1, Y1, X2, Y2, X3, Y3: REAL): INFO;
  VAR XC, YC, WC, A1, B1, C1, A2, B2, C2: REAL;
  VERTEX: INFO;
BEGIN
  XC := (X1 + X2) / 2;
  YC := (Y1 + Y2) / 2;

  IF X1 = X2 THEN
    BEGIN
      A1 := 0;
      B1 := 1;
      C1 := YC
    END  (END IF X1 = X2)
  ELSE
    BEGIN
      MC := (Y1 - Y2) / (X1 - X2);
      A1 := 1;
      B1 := MC;
      C1 := XC + MC * YC
    END  (END ELSE - X1 <> X2)

  XC := (X1 + X3) / 2;
  YC := (Y1 + Y3) / 2;

  IF X1 = X3 THEN
    BEGIN
      A2 := 0;
      B2 := 1;
      C2 := YC
    END  (END IF X1 = X3)
ELSE
BEGIN
MC := ( Y1 - Y3 ) / ( X1 -X3 );
A2 := 1;
B2 := MC;
C2 := XC + MC * YC
END; (END ELSE - X1 <> X3)
WITH VERTEX DO
BEGIN
X := ( C1 * B2 - C2 * B1 ) / ( A1 * B2 - A2 * B1);
END; (END WITH VERTEX)
VTX_FIND := VERTEX;
END; (END FUNCTION VTX_FIND)

******************************************************************************
(* IMPLEMENT THE PERIODICAL BOUNDARY CONDITION. *)
(* 1) LINK NODES IN THE ORIGINAL LINKED LIST 'SORT' AND WITHIN 'DMAX'. *)
(* FROM THE LEFT MARGIN OF THE BOX TO THE REAR OF 'SORT'. *)
(* 2) LINK NODES IN THE ORIGINAL LINKED LIST 'SORT' AND WITHIN 'DMAX'. *)
(* FROM THE RIGHT MARGIN OF THE BOX IN FRONT OF 'SORT'. *)
******************************************************************************

PROCEDURE PERIODICAL (VAR SORT: LIST;
NUM_DISK: INTEGER;
VAR NUM_LOOP: INTEGER;
DMAX, WIDE: REAL);
VAR TAIL, SORT_HEAD, HEAD: LIST;
BEGIN
(*
(* LINK NODES IN FRONT OF 'SORT'*)
(*
NEW (SORT_HEAD);
SORT_HEAD^.NEXT := Z;
HEAD := SORT_HEAD;
LAST := SORT;
REPEAT
LAST := LAST^.NEXT
UNTIL
LAST^.DATA.X >= WIDE - 4 * DMAX;
K := 0;
WHILE LAST <> Z DO
BEGIN
NEW (NBR_TEMP);
NBR_TEMP^.NEXT := Z;
WITH NBR_TEMP^.DATA DO
BEGIN
INDEX := LAST^.DATA.INDEX - ( NUM_DISK + 1 );
X := LAST^.DATA.X - WIDE;
Y := LAST^.DATA.Y;
DIA := LAST^.DATA.DIA
END; (END WITH NBR_TEMP^.DATA)
104

HEAD^.NEXT := NBR_TEMP;
READ := HEAD^.NEXT;
LAST := LAST^.NEXT;
K := K + 1
END; (END WHILE LAST <> 2)
HEAD^.NEXT := SORT;
(*
  *)
(*LINK NODES TO THE REAR OF 'SORT'*
  *)
(*
  *)
LAST := SORT;
FOR I := 1 TO NUM_DISK - 1 DO
  LAST := LAST^.NEXT;
TAIL := SORT;
J := 0;
WHILE TAIL^.DATA.X <= 4 * DMAX DO
BEGIN
  NEW (NBR_TEMP);
  NBR_TEMP^.NEXT := 2;
  WITH NBR_TEMP^.DATA DO
  BEGIN
    INDEX := TAIL^.DATA_INDEX + NUM_DISK;
    X := TAIL^.DATA.X + WIDE;
    Y := TAIL^.DATA.Y;
    DIA := TAIL^.DATA_DIA
  END; (END WITH NBR_TEMP^.DATA)
  LAST^.NEXT := NBR_TEMP;
  LAST := LAST^.NEXT;
  TAIL := TAIL^.NEXT;
  J := J + 1
END; (END WHILE TAIL^.DATA.X <= 3 * DMAX)
SORT := SORT_HEAD^.NEXT;
NUM_LOOP := NUM_DISK + K + J
END; (END PROCEDURE PERIODICAL)

(* *******************************************************
(* FIND THE ANGLE SWEEPING FROM THE POSITIVE X-DIRECTION TO *)
(* THE LINE EXTENDED FROM (X1, Y1) TO (X2, Y2) *)
(* *******************************************************
FUNCTION THETA (X1, Y1, X2, Y2: REAL): REAL;
BEGIN
  IF X2 = X1 THEN
    BEGIN
      IF Y2 > Y1 THEN THETA := 90
      ELSE THETA := 270
    END (END IF X2 = X1)
  ELSE
    BEGIN
      ALFA := 180 * ARCTAN (( Y2 - Y1 ) / ( X2 - X1 )) / PI;
      IF X2 < X1 THEN
        BEGIN
          ALFA := ALFA + 180;
          THETA := ALFA
        END (END IF X2 < X1)
ELSE IF Y2 < Y1 THEN
  BEGIN
    ALFA := ALFA + 360;
    THETA := ALFA
  END {END IF Y2 < Y1}
ELSE
  THETA := ALFA
END {END ELSE - X2 <> X1}
END; {END FUNCTION THETA }

******************************************************************************
(* 
MAKE A COPY OF A LINKED LIST 
*)
******************************************************************************
PROCEDURE MAKE_COPY (SORT: LIST; VAR SORT_COPY: LIST);
  VAR COPY_LAST: LIST;
BEGIN
  NEW (SORT_COPY);
  SORT_COPY^.NEXT := Z;
  COPY_LAST := SORT_COPY;

  LAST := SORT;
  WHILE LAST <> Z DO
    BEGIN
      NEW (NBR_TEMP);
      NBR_TEMP^.NEXT := Z;
      NBR_TEMP^.DATA := LAST^.DATA;
      COPY_LAST^.NEXT := NBR_TEMP;
      LAST := LAST^.NEXT;
      COPY_LAST := COPY_LAST^.NEXT
    END; {END WHILE LAST <> Z}
  SORT_COPY := SORT_COPY^.NEXT
END; {END PROCEDURE MAKE_COPY}

******************************************************************************
(* 
GIVEN A BOX DEFINED BY: 
*)
(* 
X = LEFT, X = RIGHT, Y = BOTTOM, AND Y = TOP 
*)
(* 
AND A SORTED LINKED LIST WITH EACH NODE A TWO DIMENSIONAL 
POINT. ELIMINATE OUTER DISKS FROM THE LINKED LIST 'SORT' 
*)
(* 
NAME THE NEW LINKED LIST 'INNER' 
*)
******************************************************************************
PROCEDURE FIND_INNER (SORT_COPY: LIST;
  LEFT, RIGHT, BOTTOM, TOP: REAL;
  VAR INNER: LIST);

  VAR X1, Y1, R: REAL;
  STOP: BOOLEAN;
BEGIN
  LAST := SORT_COPY;
  REPEAT
    STOP := TRUE;
    */
WHILE (LAST^.DATA.X <= LEFT - LAST^.DATA.DIA / 2) OR
     (LAST^.DATA.Y <= BOTTOM - LAST^.DATA.DIA / 2) OR
     (LAST^.DATA.Y >= TOP + LAST^.DATA.DIA / 2) DO
BEGIN
  NBR_TEMP := LAST;
  LAST := LAST^.NEXT;
  DISPOSE (NBR_TEMP)
END;  (END WHILE)
WITH LAST^.DATA DO
BEGIN
  X1 := X;
  Y1 := Y;
  R := DIA / 2
END;  (END WITH LAST^.DATA)

IF X1 < LEFT THEN
BEGIN
IF Y1 < BOTTOM THEN
BEGIN
  IF SQR (R) <= SQR (X1 - LEFT) + SQR (Y1 - BOTTOM) THEN
  BEGIN
    STOP := FALSE;
    NBR_TEMP := LAST;
    LAST := LAST^.NEXT;
    DISPOSE (NBR_TEMP)
  END
ENDIF Y1 < BOTTOM
ELSE IF Y1 > TOP THEN
BEGIN
  IF SQR (R) <= SQR (X1 - LEFT) + SQR (Y1 - TOP) THEN
  BEGIN
    STOP := FALSE;
    NBR_TEMP := LAST;
    LAST := LAST^.NEXT;
    DISPOSE (NBR_TEMP)
  END
END ELSE IF Y1 > TOP
END IF X1 < LEFT
UNTIL
STOP;

INNER := LAST;
WHILE LAST^.NEXT^.DATA.X < LEFT DO
BEGIN
REPEAT
  STOP := TRUE;
  WITH LAST^.NEXT^.DATA DO
  BEGIN
    X1 := X;
    Y1 := Y;
    R := DIA / 2
  END;  (END WITH LAST^.NEXT^.DATA)

END;  (END REPEAT)
INNER := LAST^.NEXT;
WHILE INNER^.DATA.Y <= TOP + INNER^.DATA.DIA / 2 DO
BEGIN
  NBR_TEMP := INNER;
  INNER := INNER^.NEXT;
  DISPOSE (NBR_TEMP)
END;  (END WHILE)
IF X1 <= LEFT - R THEN
BEGIN
    STOP := FALSE;
    NBR_TEMP := LAST^.NEXT;
    LAST^.NEXT := LAST^.NEXT^.NEXT;
    DISPOSE (NBR_TEMP)
END (END IF X1 <= LEFT - R)
ELSE IF ( Y1 <= BOTTOM - R ) OR ( Y1 >= TOP + R ) THEN
BEGIN
    STOP := FALSE;
    NBR_TEMP := LAST^.NEXT;
    LAST^.NEXT := LAST^.NEXT^.NEXT;
    DISPOSE (NBR_TEMP)
END
ELSE IF Y1 < BOTTOM THEN
BEGIN
    IF SQR (R) <= SQR (X1 - LEFT) + SQR (Y1 - BOTTOM) THEN
    BEGIN
        STOP := FALSE;
        NBR_TEMP := LAST^.NEXT;
        LAST^.NEXT := LAST^.NEXT^.NEXT;
        DISPOSE (NBR_TEMP)
    END (END IF Y1 < BOTTOM)
ELSE IF Y1 > TOP THEN
BEGIN
    IF SQR (R) <= SQR (X1 - LEFT) + SQR (Y1 - TOP) THEN
    BEGIN
        STOP := FALSE;
        NBR_TEMP := LAST^.NEXT;
        LAST^.NEXT := LAST^.NEXT^.NEXT;
        DISPOSE (NBR_TEMP)
    END (END ELSE IF Y1 > TOP)
UNTIL
( STOP ) OR ( LAST^.NEXT^.DATA.X >= LEFT );
IF STOP THEN
LAST := LAST^.NEXT
END; (END WHILE LAST^.NEXT^.DATA.X < LEFT)
WHILE LAST^.NEXT^.DATA.X <= RIGHT DO
BEGIN
    REPEAT
    STOP := TRUE;
    IF (LAST^.NEXT^.DATA.Y<=BOTTOM - LAST^.NEXT^.DATA.DIA / 2) OR
       (LAST^.NEXT^.DATA.Y=TOP + LAST^.NEXT^.DATA.DIA / 2) THEN
    BEGIN
        STOP := FALSE;
        NBR_TEMP := LAST^.NEXT;
        LAST^.NEXT := LAST^.NEXT^.NEXT;
        DISPOSE (NBR_TEMP)
    END
UNTIL
( STOP ) OR ( LAST^.NEXT^.DATA.X > RIGHT );
IF STOP THEN
    LAST := LAST^.NEXT
END;  {END WHILE LAST^.NEXT^.DATA.X <= RIGHT}

WHILE LAST^.NEXT <> Z DO
    BEGIN
        REPEAT
            STOP := TRUE;
            WITH LAST^.NEXT^.DATA DO
                BEGIN
                    X1 := X;
                    Y1 := Y;
                    R := DIA / 2
                END;  {END WITH LAST^.NEXT^.DATA}
            IF X1 >= RIGHT + R THEN
                BEGIN
                    STOP := FALSE;
                    NBR_TEMP := LAST^.NEXT;
                    LAST^.NEXT := LAST^.NEXT^.NEXT;
                    DISPOSE (NBR_TEMP)
                END  {END IF X1 >= RIGHT + R}
            ELSE IF ( Y1 <= BOTTOM - R ) OR ( Y1 >= TOP + R ) THEN
                BEGIN
                    STOP := FALSE;
                    NBR_TEMP := LAST^.NEXT;
                    LAST^.NEXT := LAST^.NEXT^.NEXT;
                    DISPOSE (NBR_TEMP)
                END
            ELSE IF Y1 < BOTTOM THEN
                BEGIN
                    IF SQR (R) <= SQR (X1 - RIGHT) + SQR (Y1 - BOTTOM) THEN
                        BEGIN
                            STOP := FALSE;
                            NBR_TEMP := LAST^.NEXT;
                            LAST^.NEXT := LAST^.NEXT^.NEXT;
                            DISPOSE (NBR_TEMP)
                        END
                    END  {END IF Y1 < BOTTOM}
                ELSE IF Y1 > TOP THEN
                    BEGIN
                        IF SQR (R) <= SQR (X1 - RIGHT) + SQR (Y1 - TOP) THEN
                            BEGIN
                                STOP := FALSE;
                                NBR_TEMP := LAST^.NEXT;
                                LAST^.NEXT := LAST^.NEXT^.NEXT;
                                DISPOSE (NBR_TEMP)
                            END
                        END  {END ELSE IF Y1 > TOP}
                    UNTIL
                        ( STOP ) OR ( LAST^.NEXT = Z );
                LAST := LAST^.NEXT
            END  {END WHILE LAST^.NEXT <> Z}
    END;  {END PROCEDURE FIND_INNER}
(* FIND THE NUMBER DISTRIBUTION *)

PROCEDURE NUM_DST (NUM: NUM_ARRAY;
    NUM_IN: INTEGER;
    VAR NMIN, NMAX: INTEGER;
    VAR DST: SMALL_ARRAY);

VAR J, K: INTEGER;
BEGIN
    NMIN := 100;
    NMAX := 0;
    FOR J := 0 TO 10
        DO DST [J] := 0;
    FOR K := 1 TO NUM_IN DO
        BEGIN
            J := NUM [K];
            IF J < NMIN THEN NMIN := J;
            IF J > NMAX THEN NMAX := J;
            DST [J] := DST [J] + 1
        END (END FOR K := 1 TO NUM_IN)
    END; (END PROCEDURE NUM_DST)

(* FIND THE ANGULAR DISTRIBUTION OF STRUCTURAL NEIGHBOURS *)

PROCEDURE ANG_DST (GROUP_SN: GROUP;
    GROUP_SUM: SMALL_ARRAY;
    SMIN, SMAX, NUM_ang_DIV: INTEGER;
    VAR DST: ARRAY_3D);

VAR I, J, K, N, Q: INTEGER;
    GAMMA: REAL;
BEGIN
    GAMMA := 360 / NUM_ang_DIV;
    FOR K := SMIN TO SMAX DO
        BEGIN
            IF K > 0 THEN
                BEGIN
                    FOR N := 1 TO GROUP_SUM [K] DO
                        NBR_LAST [N] := GROUP_SN [K, N].NEXT;
                    FOR J := 1 TO K DO
                        BEGIN
                            FOR I := 0 TO NUM_ang_DIV - 1 DO DST [K, J, I] := 0;
                            FOR N := 1 TO GROUP_SUM [K] DO
                                BEGIN
                                    Q := TRUNC ( NBR_LAST [N].DATA_ANGLE / GAMMA );
                                    DST [K, J, Q] := DST [K, J, Q] + 1;
                                    NBR_LAST [N] := NBR_LAST [N].NEXT
                                END (END FOR N := 1 TO GROUP_SUM [K])
                            END (END FOR J := 1 TO K)
                        END (END IF K > 0)
                    END (END FOR K := SMIN TO SMAX)
                END; (END PROCEDURE ANG_DST)
PROCEDURE FIND_NNR (INNERS: LIST_ARRAY;
    GNBRI: DOUBLE_ARRAY;
    NUM_IN: INTEGER;
    VAR NNR: REAL_ARRAY);

VAR NBR_TEMP: LIST;
    K, J: INTEGER;
    X1, Y1, X2, Y2: REAL;

BEGIN
    K := 1 TO NUM_IN DO
    BEGIN
        (*
        (* IF INNERS[K] is not empty, there is some structural *)
        (* NEIGHBOURS in the linked list INNERS[K]. Then, search *)
        (* FOR THE NEAREST NEIGHBOUR AMONG THE STRUCTURAL *)
        (* NEIGHBOURS. *)
        (*)
        IF INNERS[K]^<.NEXT <> Z THEN
            NBR_TEMP := INNERS[K]
            (*
            (*)
            (* IF INNERS[K] is empty, then search for the nearest *)
            (* NEIGHBOUR AMONG THE GEOMETRIC NEIGHBOURS. *)
            (*)
            ELSE
                J := INNERS[K]^<.DATA_INDEX;
                NBR_TEMP := GNBRI[J]
                END;  (END ELSE - INNERS[K] IS NOT EMPTY)
        WITH NBR_TEMP^<.DATA DO
            BEGIN
                X1 := X; Y1 := Y
            END;  (END WITH NBR_TEMP^<.DATA)
        NBR_TEMP := NBR_TEMP^<.NEXT;
        WITH NBR_TEMP^<.DATA DO
            BEGIN
                X2 := X; Y2 := Y
            END;  (END WITH NBR_TEMP^<.DATA)
        NNR [K] := SQR (SQR (X1 - X2) + SQR (Y1 - Y2));
    END;  (END WITH NBR_TEMP^<.DATA)
    WHILE NBR_TEMP <> Z DO
        BEGIN
            WITH NBR_TEMP^<.DATA DO
                BEGIN
                    X2 := X; Y2 := Y
                END;  (END WITH NBR_TEMP^<.DATA)
            R := SQR (SQR (X1 - X2) + SQR (Y1 - Y2));
            IF R < NNR [K] THEN
                NNR [K] := R;
        END;  (END WHILE)
    END;  (END FOR)
END;  (END FIND_NNR)
NBR_TEMP := NBR_TEMP^..NEXT
END;  {END WHILE NBR_TEMP <> Z}
END  {END FOR K := 1 TO NUM_LOOP}
END;  {END Procedure FIND_NNR}

(*---------------------------------------------------------------------*)
(*
SORT A ONE-DIMENSIONAL ARRAY
*)
(*---------------------------------------------------------------------*)

PROCEDURE ARRAY_SORT (VAR NNR: REAL_ARRAY; NUM_IN: INTEGER);
VAR I, J: INTEGER;
TEMP: REAL;
BEGIN
FOR I := 1 TO NUM_IN - 1 DO
  FOR J := I + 1 TO NUM_IN DO
    IF NNR[J] < NNR[I] THEN
      BEGIN
        TEMP := NNR[I];
        NNR[I] := NNR[J];
        NNR[J] := TEMP
      END  {END IF}
  END  {END FOR}
END;  {END Procedure ARRAY_SORT}

(*---------------------------------------------------------------------*)
(*
GIVEN A BOX DEFINED BY:
(*)
(*)  X = LEFT, X = RIGHT, Y = BOTTOM, AND Y = TOP
(*)
(*)  AND A SORT LINKED LIST 'INNER' WITH EACH NODE A POINT WITHIN
(*)
(*)  OR ON THE BOUNDARY OF THE BOX. FIND THE PACKING FRACTION:
(*)
(*)  SUM OF DISK AREA INSIDE THE BOX
(*)
(*)  P.F. = -------------------------------------
(*)  BOX AREA
(*)
(*)
*)
(*---------------------------------------------------------------------*)

PROCEDURE FIND_PF (INNER: LIST;
                     LEFT, RIGHT, BOTTOM, TOP: REAL;
                     VAR ETA, SUM: REAL);
VAR AREA: REAL;
ALFA, BETA, GAMMA: REAL;
X1, X2, Y1, Y2, R: REAL;
LAST: LIST;
BEGIN
  SUM := 0;
  LAST := INNER;
  WHILE (LAST^.DATA.X < LEFT + DMAX / 2) AND (LAST <> Z) DO
    BEGIN
      WITH LAST^.DATA DO
        BEGIN
          X1 := X - LEFT;
          X2 := RIGHT - X;
          Y1 := Y - BOTTOM;
          Y2 := TOP - Y;
          R := DIA / 2
        END;  {END WITH LAST^.DATA}
    END;  {END WHILE}
  END;  {END PROCEDURE FIND_PF}
IF X1 < 0 THEN
BEGIN
IF Y1 < 0 THEN
BEGIN
ALFA := 2 * ARCTAN ( SQRT ( R * R - X1 * X1 ) / ( - X1 ) );
BETA := 2 * ARCTAN ( SQRT ( R * R - Y1 * Y1 ) / ( - Y1 ) );
GAMMA := 0.5 * PI + ( ALFA + BETA ) / 2;
AREA := X1 * Y1 + R * R * ( ALFA - SIN ( ALFA ) / 2 +
                                  BETA - SIN ( BETA ) / 2 -
                                  GAMMA ) / 2
END (END IF Y1 < 0)
ELSE IF Y2 < 0 THEN
BEGIN
ALFA := 2 * ARCTAN ( SQRT ( R * R - X1 * X1 ) / ( - X1 ) );
BETA := 2 * ARCTAN ( SQRT ( R * R - Y2 * Y2 ) / ( - Y2 ) );
GAMMA := 0.5 * PI + ( ALFA + BETA ) / 2;
AREA := X1 * Y2 + R * R * ( ALFA - SIN ( ALFA ) / 2 +
                                  BETA - SIN ( BETA ) / 2 -
                                  GAMMA ) / 2
END (END ELSE IF Y2 < 0)
ELSE IF ( Y1 = 0 ) OR ( Y2 = 0 ) THEN
BEGIN
ALFA := 2 * ARCTAN ( SQRT ( R * R - X1 * X1 ) / ( - X1 ) );
AREA := R * R * ( ALFA - SIN ( ALFA ) ) / 4
END (END ELSE IF ( Y1 = 0 ) OR ( Y2 = 0 ))
ELSE IF Y1 < R THEN
BEGIN
IF R * R > X1 * X1 + Y1 * Y1 THEN
BEGIN
ALFA := 2 * ARCTAN ( SQRT ( R * R - X1 * X1 ) / ( - X1 ) );
BETA := 2 * ARCTAN ( SQRT ( R * R - Y1 * Y1 ) / Y1 );
GAMMA := 0.5 * PI + ( ALFA + BETA ) / 2;
AREA := R * R * ( GAMMA - BETA + SIN ( BETA ) / 2 -
                                  SIN ( ALFA ) / 2 ) / 2 + X1 * Y1
END (END IF R * R > X1 * X1 + Y1 * Y1)
ELSE
BEGIN
ALFA := 2 * ARCTAN ( SQRT ( R * R - X1 * X1 ) / ( - X1 ) );
AREA := R * R * ( ALFA - SIN ( ALFA ) ) / 2
END (END ELSE - R * R <= X1 * X1 + Y1 * Y1)
END (END ELSE IF Y1 < R)
ELSE IF Y2 < R THEN
BEGIN
IF R * R > X1 * X1 + Y2 * Y2 THEN
BEGIN
ALFA := 2 * ARCTAN ( SQRT ( R * R - X1 * X1 ) / ( - X1 ) );
BETA := 2 * ARCTAN ( SQRT ( R * R - Y2 * Y2 ) / Y2 );
GAMMA := 0.5 * PI + ( ALFA + BETA ) / 2;
AREA := R * R * ( GAMMA - BETA + SIN ( BETA ) / 2 -
                                  SIN ( ALFA ) / 2 ) / 2 + X1 * Y2
END (END IF R * R > X1 * X1 + Y2 * Y2)
END
112
ELSE
BEGIN
  ALFA := 2 * ARCTAN ( SQRT ( R * R - X1 * X1 ) / ( - X1 ) );
  AREA := R * R * ( ALFA - SIN ( ALFA ) ) / 2
END END ELSE - R * R <= X1 * X1 + Y2 * Y2
END END ELSE IF Y2 < R
ELSE
BEGIN
  ALFA := 2 * ARCTAN ( SQRT ( R * R - X1 * X1 ) / ( - X1 ) );
  AREA := R * R * ( ALFA - SIN ( ALFA ) ) / 2
END END ELSE
END IF X1 < 0
ELSE IF X1 = 0 THEN
BEGIN
IF Y1 < 0 THEN
BEGIN
  BETA := 2 * ARCTAN ( SQRT ( R * R - Y1 * Y1 ) / ( - Y1 ) );
  AREA := R * R * ( BETA - SIN ( BETA ) ) / 4
END END IF Y1 < 0
ELSE IF Y2 < 0 THEN
BEGIN
  ALFA := 2 * ARCTAN ( SQRT ( R * R - Y2 * Y2 ) / ( - Y2 ) );
  AREA := R * R * ( ALFA - SIN ( ALFA ) ) / 4
END END ELSE IF Y2 < 0
ELSE IF ( Y1 = 0 ) OR ( Y2 = 0 ) THEN
BEGIN
  AREA := R * R * PI / 4
END END ELSE IF ( Y1 = 0 ) OR ( Y2 = 0 )
ELSE IF Y1 < R THEN
BEGIN
  ALFA := 2 * ARCTAN ( SQRT ( R * R - Y1 * Y1 ) / Y1 );
  GAMMA := PI - ALFA / 2;
  AREA := R * R * ( SIN ( ALFA ) / 2 + GAMMA ) / 2
END END ELSE IF Y1 < R
ELSE IF Y2 < R THEN
BEGIN
  ALFA := 2 * ARCTAN ( SQRT ( R * R - Y2 * Y2 ) / Y2 );
  GAMMA := PI - ALFA / 2;
  AREA := R * R * ( SIN ( ALFA ) / 2 + GAMMA ) / 2
END END ELSE IF Y2 < R
ELSE
  AREA := R * R * PI / 2
END END ELSE IF X1 = 0
END END ELSE IF X1 < R THEN
BEGIN
  IF Y1 < 0 THEN
  BEGIN
    IF R * R > X1 * X1 + Y1 * Y1 THEN
    BEGIN
      ALFA := 2 * ARCTAN ( SQRT ( R * R - X1 * X1 ) / X1 );
      BETA := 2 * ARCTAN ( SQRT ( R * R - Y1 * Y1 ) / ( - Y1 ) );
      GAMMA := 0.5 * PI + ( ALFA + BETA ) / 2;
    END
  END
  END
END
\[
\text{AREA} := R \cdot R \cdot (\gamma - \alpha + \sin(\alpha)) / 2 - \\
\sin(\beta) / 2) / 2 + x1 \cdot y1
\]

\text{END \{END IF \( R \cdot R > x1 \cdot x1 + y1 \cdot y1 \}\}}

\text{ELSE}

\text{BEGIN}

\text{ALFA} := 2 \cdot \text{ARCTAN}(\sqrt{R \cdot R - y1 \cdot y1}) / (-y1);
\text{AREA} := R \cdot R \cdot (\alpha - \sin(\alpha)) / 2
\]

\text{END \{END ELSE - \( R \cdot R <= x1 \cdot x1 + y1 \cdot y1 \)\}}

\text{END \{END IF \( Y1 < 0 \)\}}

\text{ELSE IF \( Y2 < 0 \) THEN}

\text{BEGIN}

\text{IF \( R \cdot R > x1 \cdot x1 + y2 \cdot y2 \) THEN}

\text{BEGIN}

\text{ALFA} := 2 \cdot \text{ARCTAN}(\sqrt{R \cdot R - x1 \cdot x1}) / x1;
\text{BETA} := 2 \cdot \text{ARCTAN}(\sqrt{R \cdot R - y2 \cdot y2}) / (-y2);
\text{GAMMA} := 0.5 \cdot \pi + (\alpha + \beta) / 2;
\text{AREA} := R \cdot R \cdot (\gamma - \alpha + \sin(\alpha)) / 2 - \\
\sin(\beta) / 2) / 2 + x1 \cdot y2
\]

\text{END \{END IF \( R \cdot R > x1 \cdot x1 + y2 \cdot y2 \)\}}

\text{ELSE}

\text{BEGIN}

\text{ALFA} := 2 \cdot \text{ARCTAN}(\sqrt{R \cdot R - y2 \cdot y2}) / (-y2);
\text{AREA} := R \cdot R \cdot (\alpha - \sin(\alpha)) / 2
\]

\text{END \{END ELSE - \( R \cdot R <= x1 \cdot x1 + y2 \cdot y2 \)\}}

\text{END \{END ELSE IF \( Y2 < 0 \)\}}

\text{ELSE IF \( Y1 = 0 \) OR \( Y2 = 0 \) THEN}

\text{BEGIN}

\text{ALFA} := 2 \cdot \text{ARCTAN}(\sqrt{R \cdot R - x1 \cdot x1}) / x1;
\text{GAMMA} := \pi - \alpha / 2;
\text{AREA} := R \cdot R \cdot (\sin(\alpha)) / 2 + \gamma / 2
\]

\text{END \{END ELSE IF \( Y1 = 0 \) OR \( Y2 = 0 \)\}}

\text{ELSE IF \( Y1 < R \) THEN}

\text{BEGIN}

\text{IF \( R \cdot R > x1 \cdot x1 + y1 \cdot y1 \) THEN}

\text{BEGIN}

\text{ALFA} := 2 \cdot \text{ARCTAN}(\sqrt{R \cdot R - x1 \cdot x1}) / x1;
\text{BETA} := 2 \cdot \text{ARCTAN}(\sqrt{R \cdot R - y1 \cdot y1}) / y1;
\text{GAMMA} := 1.5 \cdot \pi - (\alpha + \beta) / 2;
\text{AREA} := x1 \cdot y1 + R \cdot R \cdot (\sin(\alpha)) / 2 + \\
\sin(\beta) / 2 + \gamma / 2
\]

\text{END \{END IF \( R \cdot R > x1 \cdot x1 + y1 \cdot y1 \)\}}

\text{ELSE}

\text{BEGIN}

\text{ALFA} := 2 \cdot \text{ARCTAN}(\sqrt{R \cdot R - x1 \cdot x1}) / x1;
\text{BETA} := 2 \cdot \text{ARCTAN}(\sqrt{R \cdot R - y1 \cdot y1}) / y1;
\text{AREA} := R \cdot R \cdot (2 \cdot \pi - \alpha) / 2 + \\
\beta + \sin(\beta) / 2
\]

\text{END \{END ELSE - \( R \cdot R <= x1 \cdot x1 + y1 \cdot y1 \)\}}

\text{END \{END ELSE IF \( Y1 < R \)\}}

\text{ELSE IF \( Y2 < R \) THEN}

\text{BEGIN}
IF \( R \times R > X1 \times X1 + Y2 \times Y2 \) THEN
BEGIN
\( ALFA := 2 \times \text{ARCTAN} ( \text{SQRT} ( R \times R - X1 \times X1 ) / X1 ) \); \( BETA := 2 \times \text{ARCTAN} ( \text{SQRT} ( R \times R - Y2 \times Y2 ) / Y2 ) \); \( GAMMA := 1.5 \times \text{PI} - ( ALFA + BETA ) / 2 \); \( AREA := X1 \times Y2 + R \times R \times ( \text{SIN} ( ALFA ) / 2 + \text{SIN} ( BETA ) / 2 + GAMMA ) / 2 \); END (END IF \( R \times R > X1 \times X1 + Y2 \times Y2 \)) ELSE
BEGIN
\( ALFA := 2 \times \text{ARCTAN} ( \text{SQRT} ( R \times R - X1 \times X1 ) / X1 ) \); \( BETA := 2 \times \text{ARCTAN} ( \text{SQRT} ( R \times R - Y2 \times Y2 ) / Y2 ) \); \( AREA := R \times R \times ( 2 \times \text{PI} - ALFA + \text{SIN} ( ALFA ) - BETA + \text{SIN} ( BETA ) ) / 2 \); END (END ELSE \( R \leq X1 \times X1 + Y2 \times Y2 \)) END (END ELSE IF \( Y2 < R \)) ELSE
BEGIN
\( ALFA := 2 \times \text{ARCTAN} ( \text{SQRT} ( R \times R - X1 \times X1 ) / X1 ) \); \( AREA := R \times R \times ( 2 \times \text{PI} - ALFA + \text{SIN} ( ALFA ) ) / 2 \); END (END ELSE)
END (END ELSE IF \( X1 < R \))
(* ...
(* \( R \leq X1 \times DMAX \times 2 \) *)
(* ...
ELSE
BEGIN
IF \( Y1 < 0 \) THEN
BEGIN
\( BETA := 2 \times \text{ARCTAN} ( \text{SQRT} ( R \times R - Y1 \times Y1 ) / ( - Y1 ) ) \); \( AREA := R \times R \times ( BETA - \text{SIN} ( BETA ) ) / 2 \); END (END IF \( Y1 < 0 \)) ELSE IF \( Y2 < 0 \) THEN
BEGIN
\( BETA := 2 \times \text{ARCTAN} ( \text{SQRT} ( R \times R - Y2 \times Y2 ) / ( - Y2 ) ) \); \( AREA := R \times R \times ( BETA - \text{SIN} ( BETA ) ) / 2 \); END (END ELSE IF \( Y2 < 0 \)) ELSE IF \( Y1 = 0 \) OR \( Y2 = 0 \) THEN
BEGIN
\( AREA := R \times R \times \text{PI} / 2 \); END (END ELSE IF \( Y1 = 0 \) OR \( Y2 = 0 \)) ELSE IF \( Y1 < R \) THEN
BEGIN
\( BETA := 2 \times \text{ARCTAN} ( \text{SQRT} ( R \times R - Y1 \times Y1 ) / Y1 ) \); \( AREA := R \times R \times ( 2 \times \text{PI} - BETA + \text{SIN} ( BETA ) ) / 2 \); END (END ELSE IF \( Y1 < R \)) ELSE IF \( Y2 < R \) THEN
BEGIN
\( BETA := 2 \times \text{ARCTAN} ( \text{SQRT} ( R \times R - Y2 \times Y2 ) / Y2 ) \); \( AREA := R \times R \times ( 2 \times \text{PI} - BETA + \text{SIN} ( BETA ) ) / 2 \); END (END ELSE IF \( Y2 < R \))
ELSE
    AREA := R * R * PI
END;  (END ELSE - R <= X1 < DMAX / 2)
SUM := SUM + AREA;
LAST := LAST^.NEXT
END;  (END WHILE LAST^.DATA.X < LEFT + DMAX / 2)
WHILE (LAST^.DATA.X <= RIGHT - DMAX / 2) AND (LAST <> Z) DO
BEGIN
WITH LAST^.DATA DO
BEGIN
    X1 := X - LEFT;
    X2 := RIGHT - X;
    Y1 := Y - BOTTOM;
    Y2 := TOP - Y;
    R := DIA / 2
END;  (END WITH LAST^.DATA)
IF Y1 < 0 THEN
BEGIN
    BETA := 2 * ARCTAN ( SQRT ( R * R - Y1 * Y1 ) / ( - Y1 ) );
    AREA := R * R * ( BETA - SIN ( BETA ) ) / 2
END  (END IF Y1 < 0)
ELSE IF Y2 < 0 THEN
BEGIN
    BETA := 2 * ARCTAN ( SQRT ( R * R - Y2 * Y2 ) / ( - Y2 ) );
    AREA := R * R * ( BETA - SIN ( BETA ) ) / 2
END  (END ELSE IF Y2 < 0)
ELSE IF (Y1 = 0) OR (Y2 = 0) THEN
BEGIN
    AREA := R * R * PI / 2
END  (END ELSE IF (Y1 = 0) OR (Y2 = 0))
ELSE IF Y1 < R THEN
BEGIN
    BETA := 2 * ARCTAN ( SQRT ( R * R - Y1 * Y1 ) / Y1 );
    AREA := R * R * ( 2 * PI - BETA + SIN ( BETA ) ) / 2
END  (END ELSE IF Y1 < R)
ELSE IF Y2 < R THEN
BEGIN
    BETA := 2 * ARCTAN ( SQRT ( R * R - Y2 * Y2 ) / Y2 );
    AREA := R * R * ( 2 * PI - BETA + SIN ( BETA ) ) / 2
END  (END ELSE IF Y2 < R)
ELSE
BEGIN
    AREA := R * R * PI
END;  (END ELSE)
SUM := SUM + AREA;
LAST := LAST^.NEXT
END;  (END WHILE LAST^.DATA.X <= RIGHT - DMAX / 2)
WHILE LAST <> Z DO
BEGIN
WITH LAST^.DATA DO
BEGIN
    X1 := X - LEFT;
    X2 := RIGHT - X;

116
IF X2 >= R THEN
BEGIN
IF Y1 < 0 THEN
BEGIN
BETA := 2 * ARCTAN ( SQRT ( R * R - Y1 * Y1 ) / ( - Y1 ) );
AREA := R * R * ( BETA - SIN ( BETA ) ) / 2
END (END IF Y1 < 0)
ELSE IF Y2 < 0 THEN
BEGIN
BETA := 2 * ARCTAN ( SQRT ( R * R - Y2 * Y2 ) / ( - Y2 ) );
AREA := R * R * ( BETA - SIN ( BETA ) ) / 2
END (END ELSE IF Y2 < 0)
ELSE IF ( Y1 = 0 ) OR ( Y2 = 0 ) THEN
BEGIN
AREA := R * R * PI / 2
END (END ELSE IF (Y1 = 0) OF (Y2 = 0))
ELSE IF Y1 < R THEN
BEGIN
BETA := 2 * ARCTAN ( SQRT ( R * R - Y1 * Y1 ) ) / Y1);
AREA := R * R * ( 2 * PI - BETA + SIN ( BETA ) ) / 2
END (END ELSE IF Y1 < R)
ELSE IF Y2 < R THEN
BEGIN
BETA := 2 * ARCTAN ( SQRT ( R * R - Y2 * Y2 ) ) / Y2);
AREA := R * R * ( 2 * PI - BETA + SIN ( BETA ) ) / 2
END (END ELSE IF Y2 < R)
ELSE
AREA := R * R * PI
END (END IF X2 >= R)
ELSE IF X2 > 0 THEN
BEGIN
IF Y1 < 0 THEN
BEGIN
IF R * R > X2 * X2 + Y1 * Y1 THEN
BEGIN
ALFA := 2 * ARCTAN ( SQRT ( R * R - X2 * X2 ) ) / X2)
BETA := 2 * ARCTAN ( SQRT ( R * R - Y1 * Y1 ) ) / ( - Y1 )
GAMMA := 0.5 * PI + ( ALFA + BETA ) / 2
AREA := R * R * ( GAMMA - ALFA + SIN ( ALFA ) ) / 2 -
        SIN ( BETA ) / 2 ) / 2 + X2 * Y1
END (END IF R * R > X2 * X2 + Y1 * Y1)
ELSE
BEGIN
BETA := 2 * ARCTAN ( SQRT ( R * R - Y1 * Y1 ) ) / ( - Y1 )
AREA := R * R * ( BETA - SIN ( BETA ) ) / 2
END (END ELSE - R * R <= X2 * X2 + Y1 * Y1)
ELSE
END (END IF X2 <= 0)
END (END IF X2 > 0)
ELSE
BEGIN
BETA := 2 * ARCTAN ( SQRT ( R * R - Y1 * Y1 ) ) / ( - Y1 )
AREA := R * R * ( BETA - SIN ( BETA ) ) / 2
END (END ELSE - R * R <= X2 * X2 + Y1 * Y1)
END (END IF Y1 < 0)
ELSE IF Y2 < 0 THEN
BEGIN
IF R * R > X2 * X2 + Y2 * Y2 THEN
BEGIN
ALFA := 2 * ACRTAN ( SQRT ( R * R - X2 * X2 ) / X2 );
BETA := 2 * ACRTAN ( SQRT ( R * R - Y2 * Y2 ) / ( - Y2 ) );
GAMA := 0.5 * PI + ( ALFA + BETA ) / 2;
AREA := R * R * ( GAMA - ALFA + SIN ( ALFA ) / 2 -
          SIN ( BETA ) / 2 ) / 2 + X2 * Y2
END (END IF R * R < X2 * X2 + Y2 * Y2)
ELSE
BEGIN
BETA := 2 * ACRTAN ( SQRT ( R * R - Y2 * Y2 ) / ( - Y2 ) );
AREA := R * R * ( BETA - SIN ( BETA ) ) / 2
END (END ELSE - R * R <= X2 * X2 + Y2 * Y2)
END (END ELSE IF Y2 < 0)
ELSE IF ( Y1 = 0 ) OR ( Y2 = 0 ) THEN
BEGIN
ALFA := 2 * ACRTAN ( SQRT ( R * R - X2 * X2 ) / X2 );
GAMA := PI - ALFA / 2;
AREA := R * R * ( GAMA + SIN ( ALFA ) / 2 ) / 2
END (END ELSE IF (Y1 = 0) OR (Y2 = 0))
ELSE IF Y1 < R THEN
BEGIN
IF R * R > X2 * X2 + Y1 * Y1 THEN
BEGIN
ALFA := 2 * ACRTAN ( SQRT ( R * R - X2 * X2 ) / X2 );
BETA := 2 * ACRTAN ( SQRT ( R * R - Y1 * Y1 ) / Y1 );
GAMA := 1.5 * PI - ( ALFA + BETA ) / 2;
AREA := X2 * Y1 + R * R * ( GAMA + SIN ( ALFA ) / 2 +
          SIN ( BETA ) / 2 ) / 2
END (END IF R * R > X2 * X2 + Y1 * Y1)
ELSE
BEGIN
ALFA := 2 * ACRTAN ( SQRT ( R * R - X2 * X2 ) / X2 );
BETA := 2 * ACRTAN ( SQRT ( R * R - Y1 * Y1 ) / Y1 );
AREA := R * R * ( 2 * PI - ALFA + SIN ( ALFA ) -
          BETA + SIN ( BETA ) ) / 2
END (END ELSE - R * R <= X2 * X2 + Y1 * Y1)
END (END ELSE IF Y1 < R)
ELSE IF Y2 < R THEN
BEGIN
IF R * R > X2 * X2 + Y2 * Y2 THEN
BEGIN
ALFA := 2 * ACRTAN ( SQRT ( R * R - X2 * X2 ) / X2 );
BETA := 2 * ACRTAN ( SQRT ( R * R - Y2 * Y2 ) / Y2 );
GAMA := 1.5 * PI - ( ALFA + BETA ) / 2;
AREA := X2 * Y2 + R * R * ( GAMA + SIN ( ALFA ) / 2 +
          SIN ( BETA ) / 2 ) / 2
END (END IF R * R > X2 * X2 + Y2 * Y2)
ELSE
BEGIN
ALFA := 2 * ARCTAN ( SQRT ( R * R - X2 * X2 ) / X2 );
BETA := 2 * ARCTAN ( SQRT ( R * R - Y2 * Y2 ) / Y2 );
AREA := R * R * ( 2 * PI - ALFA + SIN ( ALFA ) - BETA + SIN ( BETA ) ) / 2
END (END ELSE - R * R <= X2 * X2 + Y2 * Y2)
END (END ELSE IF Y2 < R)
ELSE
BEGIN
ALFA := 2 * ARCTAN ( SQRT ( R * R - X2 * X2 ) / X2 );
AREA := R * R * ( 2 * PI - ALFA + SIN ( ALFA ) ) / 2
END (END ELSE)
END (END ELSE IF X2 > 0)
ELSE IF X2 = 0 THEN
BEGIN
IF Y1 < 0 THEN
BEGIN
BETA := 2 * ARCTAN ( SQRT ( R * R - Y1 * Y1 ) / ( - Y1 ) );
AREA := R * R * ( BETA - SIN ( BETA ) ) / 4
END (END IF Y1 < 0)
ELSE IF Y2 < 0 THEN
BEGIN
BETA := 2 * ARCTAN ( SQRT ( R * R - Y2 * Y2 ) / ( - Y2 ) );
AREA := R * R * ( BETA - SIN ( BETA ) ) / 4
END (END ELSE IF Y2 < 0)
ELSE IF ( Y1 = 0 ) OR ( Y2 = 0 ) THEN
BEGIN
AREA := R * R * PI / 4
END (END ELSE IF (Y1 = 0) OR (Y2 = 0))
ELSE IF Y1 < R THEN
BEGIN
BETA := 2 * ARCTAN ( SQRT ( R * R - Y1 * Y1 ) / Y1 );
GAMMA := PI - BETA / 2;
AREA := R * R * ( GAMMA + SIN ( BETA ) / 2 ) / 2
END (END ELSE IF Y1 < R)
ELSE IF Y2 < R THEN
BEGIN
BETA := 2 * ARCTAN ( SQRT ( R * R - Y2 * Y2 ) / Y2 );
GAMMA := PI - BETA / 2;
AREA := R * R * ( GAMMA + SIN ( BETA ) / 2 ) / 2
END (END ELSE IF Y2 < R)
ELSE
AREA := R * R * PI / 2
END (END ELSE IF X2 = 0)
(* *)
(*X2 < 0*)
(* *)
ELSE
BEGIN
IF Y1 < 0 THEN
BEGIN
ALFA := 2 * ARCTAN ( SQRT ( R * R - X2 * X2 ) / ( - X2 ) );
BETA := 2 * ARCTAN ( SQRT ( R * R - Y1 * Y1 ) / ( - Y1 ) );
GAMMA := 0.5 * PI + ( ALFA + BETA ) / 2;
END
\text{AREA} := R \times R \times (\text{ALFA} - \sin(\text{ALFA}) / 2 + \\
\text{BETA} - \sin(\text{BETA}) / 2 - \text{GAMMA}) / 2 + X2 \times Y1
\text{END (END IF Y1 < 0)}
\text{ELSE IF Y2 < 0 THEN}
\text{BEGIN}
\text{ALFA} := 2 \times \arctan(\sqrt{(R \times R - X2 \times X2)} / (-X2));
\text{BETA} := 2 \times \arctan(\sqrt{(R \times R - Y2 \times Y2)} / (-Y2));
\text{GAMMA} := 0.5 \times \pi + (\text{ALFA} + \text{BETA}) / 2;
\text{AREA} := R \times R \times (\text{ALFA} - \sin(\text{ALFA}) / 2 + \\
\text{BETA} - \sin(\text{BETA}) / 2 - \text{GAMMA}) / 2 + X2 \times Y2
\text{END (END ELSE IF Y2 < 0)}
\text{ELSE IF (Y1 = 0) OR (Y2 = 0) THEN}
\text{BEGIN}
\text{ALFA} := 2 \times \arctan(\sqrt{(R \times R - X2 \times X2)} / (-X2));
\text{AREA} := R \times R \times (\text{ALFA} - \sin(\text{ALFA}) / 4
\text{END (END ELSE IF (Y1 = 0) OR (Y2 = 0))}
\text{ELSE IF Y1 < R THEN}
\text{BEGIN}
\text{IF R \times R > X2 \times X2 + Y1 \times Y1 THEN}
\text{BEGIN}
\text{ALFA} := 2 \times \arctan(\sqrt{(R \times R - X2 \times X2)} / (-X2));
\text{BETA} := 2 \times \arctan(\sqrt{(R \times R - Y1 \times Y1)} / Y1);
\text{GAMMA} := 0.5 \times \pi + (\text{ALFA} + \text{BETA}) / 2;
\text{AREA} := R \times R \times (\text{GAMMA} - \text{BETA} + \sin(\text{BETA}) / 2 \\
\quad - \sin(\text{ALFA}) / 2) / 2 + X2 \times Y1
\text{END (END IF R \times R > X2 \times X2 + Y1 \times Y1)}
\text{ELSE}
\text{BEGIN}
\text{ALFA} := 2 \times \arctan(\sqrt{(R \times R - X2 \times X2)} / (-X2));
\text{AREA} := R \times R \times (\text{ALFA} - \sin(\text{ALFA}) / 2)
\text{END (END ELSE - R \times R <= X2 \times X2 + Y1 \times Y1)}
\text{END (END ELSE IF Y1 < R)}
\text{ELSE IF Y2 < R THEN}
\text{BEGIN}
\text{IF R \times R > X2 \times X2 + Y2 \times Y2 THEN}
\text{BEGIN}
\text{ALFA} := 2 \times \arctan(\sqrt{(R \times R - X2 \times X2)} / (-X2));
\text{BETA} := 2 \times \arctan(\sqrt{(R \times R - Y2 \times Y2)} / Y2);
\text{GAMMA} := 0.5 \times \pi + (\text{ALFA} + \text{BETA}) / 2;
\text{AREA} := R \times R \times (\text{GAMMA} - \text{BETA} + \sin(\text{BETA}) / 2 \\
\quad - \sin(\text{ALFA}) / 2) / 2 + X2 \times Y2
\text{END (END IF R \times R > X2 \times X2 + Y2 \times Y2)}
\text{ELSE}
\text{BEGIN}
\text{ALFA} := 2 \times \arctan(\sqrt{(R \times R - X2 \times X2)} / (-X2));
\text{AREA} := R \times R \times (\text{ALFA} - \sin(\text{ALFA}) / 2)
\text{END (END ELSE - R \times R <= X2 \times X2 + Y2 \times Y2)}
\text{END (END ELSE IF Y2 < R)}
ELSE
BEGIN
  ALFA := 2 * ARCTAN ( SQRT ( R * R - X2 * X2 ) / ( - X2 ) );
  AREA := R * R * ( ALFA - SIN ( ALFA ) ) / 2
END { END ELSE }
END; { END ELSE - X2 < 0 }
SUM := SUM + AREA;
LAST := LAST^$.NEXT
END; { END WHILE LAST <> Z }

ETA := SUM / ((RIGHT - LEFT) * (TOP - BOTTOM))
END; { END PROCEDURE FIND_PF }

(***************************************************************)
(*
(*  THE MAIN PROGRAM STARTS FROM HERE  *)
(*
(***************************************************************)
BEGIN
RESET (DISKIN);
REWRITE (NBRFILE);
REWRITE (VTXFILE);
REWRITE (BOX);
REWRITE (DELAUNAY);
REWRITE (SNBRFILE);
REWRITE (TABLE);
READLN (DISKIN, STRING30, NUM_DISK);
READLN (DISKIN, STRING30, EQ_DISKS);
READLN (DISKIN, STRING30, KNOWN_TO_BE_RCP);
READLN (DISKIN, STRING30, WIDE);
READLN (DISKIN, STRING30, PERIODICAL_REQUIRED);
READLN (DISKIN, STRING30, FIND_WALL_EFFECT);
READLN (DISKIN, STRING30, HEIGHT, WIDTH);
READLN (DISKIN, STRING30, DEFINE_PF_BOX);
IF DEFINE_PF_BOX THEN
  READLN (DISKIN, PF_L, PF_R, PF_B, PF_T);
READLN (DISKIN, STRING30, TOLERANCE);
READLN (DISKIN, STRING30, NUM_ANG_DIV);
(*
(*  FORM A DUMMY END NODE FOR LINKED LISTS*)
(*
NEW (Z);
Z^.NEXT := Z;
WITH Z^.DATA DO
BEGIN
  X := 1000;
  Y := 1000
END; { END WITH Z^.DATA }
(*
(*  CALL LIST_CREATE TO FORM A LINKED LIST OF POINTS NOT SORTED*)
(*
RANDOM := LIST_CREATE;
(*CALL MERGESORT TO FORM A LINKED LIST OF POINTS SORTED*)
(*                    *)
SORT := MERGESORT (RANDOM, NUM_DISK);
(*                    *)
(*FIND THE MAXIMUM DIAMETER*)
(*                    *)
IF NOT EQ_DISKS THEN
  BEGIN
    LAST := SORT;
    DMAX := LAST^.DATA.DIA;

    LAST := LAST^.NEXT;
    FOR I := 2 TO NUM_DISK DO
      BEGIN
        IF DMAX < LAST^.DATA.DIA THEN
          DMAX := LAST^.DATA.DIA;
          LAST := LAST^.NEXT
      END (END FOR I := 2 TO NUM_DISK)
    END (END IF NOT EQ_DISKS)
ELSE
  BEGIN
    DMAX := SORT^.DATA.DIA
  END; (END ELSE - EQ_DISKS)

WRITELN (BOX);
WRITELN (BOX, ' $MAXIMUM DISK DIAMETER:', DMAX);
(*                    *)
(*IMPLEMENT THE PERIODICAL BOUNDARY CONDITION IF REQUIRED.*)
(*OTHERWISE, KEEP THE ORIGINAL LINKED LIST OF DISK 'SORT'.*)
(*                    *)
IF PERIODICAL_REQUIRED THEN
  BEGIN
    PERIODICAL (SORT, NUM_DISK, NUM_LOOP, DMAX, WIDE);
    WRITELN (BOX);
    WRITELN (BOX, ' $NUMBER OF DISKS TO BE PROCESSED AFTER IMPLEMENTATION');
    WRITELN (BOX, ' OF THE PERIODICAL BOUNDARY CONDITION:', NUM_LOOP);
  END (END IF PERIODICAL_REQUIRED)
ELSE
  BEGIN
    NUM_LOOP := NUM_DISK;
    WRITELN (BOX);
    WRITELN (BOX, ' $NO PERIODICAL BOUNDARY CONDITION REQUIRED.');
    WRITELN (BOX, ' $NUMBER OF DISKS TO BE HANDLED IS EXACTLY THE SAME');
    WRITELN (BOX, ' AS NUMBER OF DISKS IN THE ENSEMBLE:', NUM_LOOP)
  END; (END ELSE - NOT PERIODICAL_REQUIRED)
(*                    *)
(*FIND THE EXTREMA OF THE COORDINATES OF THE BOX, XMAX, XMIN, YMAX, YMIN*)
(*                    *)
LAST := SORT;
WITH LAST^.DATA DO
  BEGIN
    XMAX := X;

XMIN := X;
YMAX := Y;
YMIN := Y
END;  {END WITH LAST^.DATA}
LAST := LAST^.NEXT;
WHILE LAST <> Z DO
BEGIN
  IF XMAX < LAST^.DATA.X THEN XMAX := LAST^.DATA.X;
  IF XMIN > LAST^.DATA.X THEN XMIN := LAST^.DATA.X;
  IF YMAX < LAST^.DATA.Y THEN YMAX := LAST^.DATA.Y;
  IF YMIN > LAST^.DATA.Y THEN YMIN := LAST^.DATA.Y;
  LAST := LAST^.NEXT
END;  {END WHILE LAST <> Z}
XMAX := XMAX + BOX_C * DMAX; XMIN := XMIN - BOX_C * DMAX;
YMAX := YMAX + BOX_C * DMAX; YMIN := YMIN - BOX_C * DMAX;
WRITELN (BOX);
WRITELN (BOX, ' $THE EXTREMA OF THE BOX:')
WRITELN (BOX);
WRITELN (BOX, ' XMIN = ', XMIN, ' XMAX = ', XMAX);
WRITELN (BOX, ' YMIN = ', YMIN, ' YMAX = ', YMAX);
(*
(*INITIALIZE*)
(*
(*
FOR K := 1 TO NUM_LOOP DO
BEGIN
  NEW (NBR [K]);
  NBR [K]^ .NEXT := Z;
  NBR_LAST [K] := NBR [K];

  NEW (VTX [K]);
  VTX [K]^ .NEXT := Z;
  VTX_LAST [K] := VTX [K];

  NEW (NEW_NBR);
  NEW_NBR^.NEXT := Z;

  NEW (NEW_VTX);
  NEW_VTX^.NEXT := Z
END;  {END FOR K := 1 TO NUM_LOOP}
(*
(*1) FIND THE DISKS WITH CENTROIDS NEAREST TO FOUR CORNERS OF THE BOX*)
(*
CORNER1 := NEAREST (XMIN, YMIN, SORT);
CORNER2 := NEAREST (XMIN, YMAX, SORT);
CORNER3 := NEAREST (XMAX, YMAX, SORT);
CORNER4 := NEAREST (XMAX, YMIN, SORT);
(*
(*FORM THE LISTHEAD OF NBR [1] BY CORNER1*)
(*
NEW (NBR_TEMP);
NBR_TEMP^.NEXT := Z;
NBR_TEMP^.DATA := CORNER1;
NBR_LAST[1]^.*NEXT := NBR_TEMP;
(*
 *)
(* FORM THE LISTHEAD OF VTX[1]*
 (*
 *)
NEW (VTX_TEMP);
VTX_TEMP^.NEXT := Z;
VTX_TEMP^.DATA := CORNER1;

VTX_LAST[1]^.*NEXT := VTX_TEMP;

NEW (VTX_TEMP);
VTX_TEMP^.NEXT := Z;
WITH VTX_TEMP^.DATA DO
BEGIN
  X := XMIN;
  Y := YMIN
END; （END WITH VTX_TEMP^.DATA）
VTX_LAST[1]^.*NEXT := VTX_TEMP;

………………………………………………………………………………
(* FIND NEW NBR’S AND NEW VTX’S *)
………………………………………………………………………………
K := 1; J := 0;
DUMMY_CREATE (CORNER1.X, CORNER1.Y, XMAX, XMIN, YMAX, YMIN,
ДUMMY1, ДUMMY2, ДUMMY3, ДUMMY4);
NBR_FIRST := DUMMY4;
NEW_NBR^.DATA := DUMMY1;
ADD := FALSE;
REPEAT
WITH NBR[K]^.*NEXT^.DATA DO
BEGIN
  X1 := X; Y1 := Y
END; （END WITH NBR[K]^.*NEXT^.DATA）
WITH NEW_NBR^.DATA DO
BEGIN
  X2 := X; Y2 := Y
END; （END WITH NEW_NBR^.DATA）
DUMMY_CREATE (X1, Y1, XMAX, XMIN, YMAX, YMIN,
ДUMMY1, ДUMMY2, ДUMMY3, ДUMMY4);
REPEAT
DUMMY_FIND (X1, Y1, X2, Y2,
    ДUMMY1, ДUMMY2, ДUMMY3, ДUMMY4,
    FIRST, SECOND);
(*
 *)
(* ASSIGN THE FIRST DUMMY POINT AS THE FIRST POTENTIAL NEW NBR*）
(*
 *)
NEW_NBR^.DATA := FIRST;
WITH NEW_NBR^.DATA DO
BEGIN
  X3 := X; Y3 := Y
END; （END WITH NEW_NBR^.DATA）
NEW_VTX^.DATA := VTX_FIND (X1, Y1, X2, Y2, X3, Y3);
WITH NEW_VTX^.DATA DO
BEGIN
   XC := X; YC := Y
END; (END WITH NEW_VTX^.DATA)
R := SORT ( SQR ( X3 - XC ) + SQR ( Y3 - YC ) );
XL := XC - R;
XR := XC + R;
LAST := SORT;

(* " *)
(*VERIFY IF THE POTENTIAL NEW NBR IS A REAL NBR BY")
(*TESTING IN SEQUENCE FROM THE FIRST CENTROID AMONG")
(*CENTROIDS IN THE LINKED LIST 'SORT' *)
(* " *)
REPEAT
WITH LAST^.DATA DO
BEGIN
   X4 := X; Y4 := Y;
   NEW_R := SORT ( SQR ( X4 - XC ) + SQR ( Y4 - YC ) )
END; (END WITH LAST^.DATA)

(* " *)
(*IF X4, THE X-COORDINATE OF THE POINT UNDER TEST, IS IN THE*)
(*INTERVAL (XL, XR): THE LEFT MOST AND THE RIGHT MOST X-*)
(*COORDINATE OF THE CIRCLE DETERMINED BY (X1, Y1), THE CENTROID*)
(*BEING PROCESSED, (X2, Y2), THE PREVIOUS NBR, AND (X3, Y3),")
(*THE PRESENT POTENTIAL NEW NBR... *)
(* " *)
IF ( XL <= X4 ) AND ( X4 <= XR ) THEN
   (* " *)
      (*IF 1) X4 IS IN THE INTERVAL (XL, XR) *)
      (* 2) (X4, Y4) IS IN THE RIGHT HALF PLANE WITH RESPECT*)
      (* TO THE LINE FROM (X1, Y1) TO (X2, Y2)... *)
      (* " *)
      IF ( X2 - X1 ) * ( Y4 - Y1 ) <
         ( X4 - X1 ) * ( Y2 - Y1 ) THEN
         (* " *)
            (*IF 1) X4 IS IN THE INTERVAL (XL, XR) *)
            (* 2) (X4, Y4) IS IN THE RIGHT HALF PLANE WITH *)
            (* RESPECT TO THE LINE FROM (X1, Y1) TO (X2, Y2) *)
            (* 3) (X4, Y4) IS IN THE RIGHT HALF PLANE WITH *)
            (* RESPECT TO THE LINE FROM (X1, Y1) TO (X3, Y3) *)
            (* 4) (X4, Y4) IS IN THE CIRCLE *)
            (*THEN (X4, Y4) REPLACES (X3, Y3) AS THE POTENTIAL NEW")
            (*"NBR. THE POTENTIAL NEW VTX IS TO BE CALCULATED AGAIN")
            (* " *)
            IF (X3 - X1) * (Y4 - Y1) <
               (X4 - X1) * (Y3 - Y1) THEN
               BEGIN
                 IF R > NEW_R THEN
                    BEGIN
                       NEW_NBR^.DATA := LAST^.DATA;
                       X3 := X4; Y3 := Y4;
                    END;
               END;
END;
NEW_VTX^DATA :=
  VTX_FIND (X1, Y1, X2, Y2, X3, Y3);
WITH NEW_VTX^DATA DO
BEGIN
  XC := X; YC := Y
END;  {END WITH NEW_VTX^DATA}
R := SQRT (SQR (X3 - XC) + SQR (Y3 - YC));
XL := XC - R; XR := XC + R;
LAST := SORT
END  {END IF R > NEW_R}
(*   *)
(* IF 1), 2), AND 3) ARE SATISFIED,  *)
(* WHILE (X4, Y4) IS OUT OF THE CIRCLE *)
(* THEN KEEP (X3, Y3) AS THE POTENTIAL NEW *)
(* NEIGHBOUR. GO TO THE NEXT NODE ON THE LINKED*)
(* LIST 'SORT' AND KEEP ON TESTING.       *)
(*   *)
ELSE
LAST := LAST^.NEXT
END
(*   *)
(* IF 1), 2) ARE SATISFIED,  *)
(* WHILE 1) (X4, Y4) IS IN THE LEFT HALF PLANE WITH *)
(* RESPECT TO THE LINE FROM (X1, Y1) TO *)
(*  (X3, Y3)  *)
(*   *)
(* 2) NEW_R IS CLOSE TO OR SIGNIFICANTLY LESS *)
(* THAN R *)
(* THEN REDEFINE THE CIRCLE USING (X1, Y1), (X2, Y2), *)
(* AND (X3, Y3) *)
(*   *)
ELSE IF NEW_R - R < 0.00001 THEN
BEGIN
  VERTEX_TEMP := VTX_FIND (X1, Y1, X2, Y2, X4, Y4);
  WITH VERTEX_TEMP DO
BEGIN
  XC_TEMP := X; YC_TEMP := Y
END;  {END WITH VERTEX_TEMP}
R_TEMP := SQRT (SQR (X3 - XC_TEMP) +
           SQR (Y3 - YC_TEMP));
NEW_R := SQRT (SQR (X4 - XC_TEMP) +
               SQR (Y4 - YC_TEMP));
(*   *)
(* (X4, Y4) IS IN THE CIRCLE *)
(*   *)
IF R_TEMP - NEW_R > 0 THEN
BEGIN
  NEW_NBR^DATA := LAST^.DATA;
  X3 := X4;
  Y3 := Y4;
  NEW_VTX^DATA := VERTEX_TEMP;
  XC := XC_TEMP;
  YC := YC_TEMP;
  R := NEW_R;
END
XL := XC - R; XR := XC + R;
LAST := SORT
END (END IF R_TEMP > NEW_R)

(*
(*)
(* X4, Y4 IS OUT OF THE CIRCLE *)
(*)
*)
ELSE
LAST := LAST^.NEXT
END (END ELSE IF)

(*
(*)
(* IF 1) AND 2) ARE SATISFIED, *)
(* WHILE 1) (X4, Y4) IS ON THE LEFT HALF PLANE WITH *)
(* RESPECT TO THE LINE FROM (X1, Y1) TO *)
(* (X3, Y3) *)
(* 2) NEW_R IS SIGNIFICANTLY GREATER THAN R *)
(* THEN KEEP (X3, Y3) AS THE POTENTIAL NEW NEIGHBOUR. *)
(* GO TO THE NEXT NODE ON THE LINKED LIST 'SORT' AND *)
(* KEEP ON TESTING *)
(*)
ELSE
LAST := LAST^.NEXT

(*
(*)
(*IF X4 IS IN THE INTERVAL (XL, XR) *)
(*WHILE (X4, Y4) IS IN THE LEFT HALF PLANE WITH RESPECT*)
(*TO THE LINE FROM (X1, Y1) TO (X2, Y2) *)
(*THEN KEEP (X3, Y3) AS THE POTENTIAL NEW NBR. GO TO THE*)
(*NEXT NODE ON THE LINKED LIST 'SORT' AND KEEP ON TESTING*)
(*)
ELSE
LAST := LAST^.NEXT

(*
(*)
(*IF X4 IS OUT OF THE INTERVAL (XL, XR) *)
(*KEEP (X3, Y3) AS THE POTENTIAL NEW NBR. GO TO THE NEXT*)
(*NODE ON THE LINKED LIST 'SORT' AND KEEP ON TESTING *)
(*)
ELSE
LAST := LAST^.NEXT

(*
(*)
(*X3, Y3) REMAINS A POTENTIAL NBR AFTER ALL POINTS IN THE 'SORT'*)
(*HAVE BEEN PUT THROUGH THE ABOVE TEST, THEN STOP THE TEST AND*)
(*KEEP (X3, Y3) AS THE NEW NBR, (XC, YC) AS THE NEW VTX *)
(*)
UNTIL
(XR < X4) OR (LAST = Z);
(*
(*UPDATE THE NEW NBR FOR THE SUBSEQUENT NEW NBR FINDING PROCESS*)
(*)
X2 := X3; Y2 := Y3;
IF (XC < XMIN - 0.00001) OR (YC > YMAX + 0.00001) OR
(XC > XMAX + 0.00001) OR (YC < YMIN - 0.00001) THEN
BEGIN
    NEW_NBR^DATA := SECOND;
    WITH SECOND DO BEGIN
        X2 := X; Y2 := Y
    END; (END WITH SECOND)
    NEW_VTX^DATA :=
        VTX_FIND (X1, Y1,
                  NBR_LAST [K]^DATA.X, NBR_LAST [K]^DATA.Y,
                  X2, Y2)
END;

(*
  *)
("DETERMINE IF THE BOOLEAN VARIABLE 'PLUS' IS TRUE")
(*
  *)
PLUS := TRUE;
IF NEW_NBR^DATA.INDEX <> 0 THEN
BEGIN
    I := 1;
    REPEAT
        IF NEW_NBR^DATA.INDEX = NBR [I]^NEXT^DATA.INDEX THEN
            PLUS := FALSE;
        I := I + 1
    UNTIL
        ( NOT PLUS ) OR ( I > K + J )
END
ELSE
    PLUS := FALSE;

(*
  *)
("CONNECT THE NEW NBR AND NEW VTX TO NBR [K]'S AND VTX [K]'S"
  *)
("WHICH CONTAIN THEM"
  *)
("THE NBR'S AND VTX'S IN EACH NBR [K]'S AND VTX [K]'S ARE"
  *)
("SEQUENCED IN CLOCKWISE ORDER"
  *)
(*
  *)
("IF THE NEW NBR IS NOT A DUMMY POINT..."
  *)
("*
  *)
IF ( NEW_NBR^DATA.INDEX <> 0 ) THEN

(*
  *)
("IF THE NEW NBR IS NEITHER A DUMMY POINT NOR THE FIRST NBR..."
  *)
("*
  *)
IF ( ( NEW_NBR^DATA.X <> NBR_FIRST.X ) OR
    ( NEW_NBR^DATA.Y <> NBR_FIRST.Y ) ) THEN
BEGIN

128
(* IF THE NEW NBR IS NEITHER A DUMMY POINT NOR *)
(* THE FIRST NBR, AND THERE IS STILL NO LINKED *)
(* LIST OF NBR'S CORRESPONDING TO IT *)
(* *)
IF PLUS THEN
BEGIN
  J := J + 1;
  IF J >= 1 THEN
BEGIN
  (* *)
  (* CREATE THE NEW LINKED LIST OF NBR'S *)
  (* CORRESPONDING TO THE NEW NBR *)
  (* *)
  NEW (NBR_TEMP);
  NBR_TEMP^.NEXT := Z;
  NBR_TEMP^.DATA := NEW_NBR^.DATA;

  NBR_LAST [K + J]^.NEXT := NBR_TEMP;

  NEW (NBR_TEMP);
  NBR_TEMP^.NEXT := Z;
  NBR_TEMP^.DATA := NBR [K]^.NEXT^.DATA;

  NBR_LAST [K + J]^.NEXT := NBR_TEMP;
END
IF ADD THEN
BEGIN
  NEW (NBR_TEMP);
  NBR_TEMP^.NEXT := Z;
  NBR_TEMP^.DATA := NBR_LAST [K]^.DATA;

  NBR_LAST [K + J]^.NEXT := NBR_TEMP;
END
(* END IF ADD *)
(* CREATE THE NEW LINKED LIST OF VTX'S *)
(* CORRESPONDING TO THE NEW NBR *)
(* *)
NEW (VTX_TEMP);
VTX_TEMP^.NEXT := Z;
VTX_TEMP^.DATA := NEW_NBR^.DATA;

VTX_LAST [K + J]^.NEXT := VTX_TEMP;

NEW (VTX_TEMP);
VTX_TEMP^.NEXT := Z;
VTX_TEMP^.DATA := NEW_VTX^.DATA;
VTX_LAST[K + J]^\$.NEXT := VTX_TEMP;
VTX_LAST[K + J] := VTX_LAST[K + J]^\$.NEXT;

IF ( J > 1 ) AND ( ADD ) THEN
BEGIN
(* *)
(*INSERT THE NEW NBR AS THE FIRST*)
(*NBR TO THE PREVIOUS NBR *)
(* *)
NEW (NBR_TEMP);
NBR_TEMP^.DATA := NEW_NBR^.DATA;

NBR_TEMP^.NEXT :=
    NBR [K + J - 1]^\$.NEXT^.NEXT;
NBR [K + J - 1]^\$.NEXT^.NEXT := NBR_TEMP;

(* *)
(*INSERT THE NEW VTX AS THE FIRST*)
(*VTX TO THE PREVIOUS NBR *)
(* *)
NEW (VTX_TEMP);
VTX_TEMP^.DATA := NEW_VTX^.DATA;

VTX_TEMP^.NEXT :=
    VTX [K + J - 1]^\$.NEXT^.NEXT;
VTX [K + J - 1]^\$.NEXT^.NEXT := VTX_TEMP
END; (END IF (J > 1) AND (ADD))
END; (END IF J >= 1)

(* *)
(*ASSIGN 'ADD' TO BE TRUE AS IN THE CASE*)
(*THAT THE NEW NBR IS NEITHER A DUMMY POINT*)
(*NOR THE FIRST NBR, AND THERE IS STILL NO*)
(*LINKED LIST OF NBR CORRESPONDING TO IT*)
(* *)
ADD := TRUE
END; (END IF PLUS)

(* *)
(*ELSE IF THE NEW NBR IS NEITHER A DUMMY POINT NOR*)
(*THE FIRST NBR, BUT THERE IS ALREADY ONE LINKED*)
(*LIST OF NBR'S CORRESPONDING TO IT*)
(* *)
ELSE
BEGIN
IF ( ADD = TRUE ) AND ( J >= 1 ) THEN
BEGIN
NEW (VTX_TEMP);
VTX_TEMP^.NEXT := Z;
VTX_TEMP^.DATA := NEW_VTX^.DATA;
VTX_TEMP^.NEXT := VTX[K + J]^\$.NEXT^.NEXT;
VTX[K + J]^\$.NEXT^.NEXT := VTX_TEMP
END;
ADD := FALSE
END; (END ELSE - NOT PLUS)

130
/*
  /*ASSIGN THE NEW NBR AS THE LAST NBR*/
  /*TO THE CENTROID BEEN PROCESSED*/
  */
NEW (NBR_TEMP);
NBR_TEMP^NEXT := Z;
NBR_TEMP^.DATA := NEW_NBR^.DATA;

NBR_LAST [K]^.NEXT := NBR_TEMP;

/*
  /*ASSIGN THE NEW VTX AS THE LAST VTX*/
  /*TO THE CENTROID BEEN PROCESSED*/
  */
NEW (VTX_TEMP);
VTX_TEMP^.DATA := NEW_VTX^.DATA;
VTX_TEMP^.NEXT := Z;

VTX_LAST [K]^.NEXT := VTX_TEMP;

END

/*
  /*ELSE IF THE NEW NBR IS NOT A DUMMY*/
  /*POINT, BUT IT IS THE FIRST NBR*/
  */
ELSE
BEGIN
IF ADD THEN
BEGIN
/*
  /*INSERT THE NEW NBR AS THE FIRST*/
  /*NBR TO THE PREVIOUS NBR*/
  */
NEW (NBR_TEMP);
NBR_TEMP^.NEXT := Z;
NBR_TEMP^.DATA := NEW_NBR^.DATA;

NBR_TEMP^.NEXT := NBR [K + J]^.NEXT^.NEXT;
NBR [K + J]^.NEXT^.NEXT := NBR_TEMP;

/*
  /*ASSIGN THE PREVIOUS NBR AS THE*/
  /*LAST NBR TO THE FIRST NBR*/
  */
NEW (NBR_TEMP);
NBR_TEMP^.NEXT := Z;
NBR_TEMP^.DATA := NBR_LAST [K]^.DATA;

NBR_LAST [K + 1]^.NEXT := NBR_TEMP;

IF J >= 1 THEN
BEGIN

END

131
(* INSERT THE NEW VTX AS THE FIRST *)
(* VTX TO THE PREVIOUS NBR *)
(* *)
NEW (VTX_TEMP);
VTX_TEMP^.DATA := NEW_VTX^.DATA;

VTX_TEMP^.NEXT := VTX [K + J]^..NEXT^.NEXT;
VTX [K + J]^..NEXT^.NEXT := VTX_TEMP;

IF J > 1 THEN
BEGIN

(* *)
(* ASSIGN THE NEW VTX AS THE *)
(* LAST VTX TO THE FIRST NBR *)
(* *)
NEW (VTX_TEMP);
VTX_TEMP^.NEXT := Z;
VTX_TEMP^.DATA := NEW_VTX^.DATA;

VTX_LAST [K + 1]^..NEXT := VTX_TEMP;
VTX_LAST [K + 1] :=
VTX_LAST [K + 1]^..NEXT
END (END IF J > 1)
END (END IF J >= 1)
END; (END IF ADD)

(* *)
(* ASSIGN THE NEW VTX AS THE LAST VTX TO THE CENTROID *)
(* *)
NEW (VTX_TEMP);
VTX_TEMP^.NEXT := Z;
VTX_TEMP^.DATA := NEW_VTX^.DATA;

VTX_LAST [K]^..NEXT := VTX_TEMP;
VTX_LAST [K] := VTX_LAST [K]^..NEXT

(* *)
(* LEAVE THE BOOLEAN VARIABLE 'ADD' UNCHANGED AS *)
(* IN THE CASE THAT THE NEW NBR IS THE FIRST NBR *)
(* *)
END

(* *)
(* ELSE IF THE NEW NBR IS A DUMMY POINT... *)
(* *)
ELSE BEGIN
IF ADD THEN
BEGIN
IF J >= 1 THEN
BEGIN

132
(*INSERT THE NEW VTX AS THE FIRST*)
(*VTX TO THE PREVIOUS NBR*)
(*NEW (VTX_TEMP)*
VTX_TEMP^.DATA := NEW_VTX^.DATA;

VTX_TEMP^.NEXT := VTX [K + J]^_.NEXT^.NEXT;
VTX [K + J]^_.NEXT^.NEXT := VTX_TEMP
END; {END IF J >= 1)

(* Assign the new VTX as the last VTX to the centroid *)
(*NEW(VTX_TEMP)*
VTX_TEMP^.NEXT := Z;
VTX_TEMP^.DATA := NEW_VTX^.DATA;

VTX_LAST [K]^_.NEXT := VTX_TEMP;
END {END IF ADD}

(* Else if the new nbr is a dummy point; 'ADD' is false; *)
(*i.e., the previous new nbr is a dummy point or the first*)
(*"NBR EITHER TO THE VERY CENTROID PROCESSED PREVIOUSLY OR*)
(*"TO THE CENTROID PRESENTLY BEEN PROCESSED, AND THE NEW*)
(*"VTX IS ONE OF THE VERTICES OF THE BOX...*)
(*ELSE*
ELSE IF ( NEW_VTX^.DATA.X <> VTX_LAST [K]^_.DATA.X ) OR
( NEW_VTX^.DATA.Y <> VTX_LAST [K]^_.DATA.Y ) THEN
BEGIN
NEW (VTX_TEMP);
VTX_TEMP^.NEXT := Z;
VTX_TEMP^.DATA := NEW_VTX^.DATA;

VTX_LAST [K]^_.NEXT := VTX_TEMP;
END;

(*Assign 'ADD' to be false as in the case*)
(*"THAT THE NEW NBR IS A DUMMY POINT")
(*ADD := FALSE*
END {END ELSE - NEW_NBR^.DATA_INDEX = 0}

(*Stop processing the centroid presently being*)
(*Processed as all it's nbr's had been found*)
(*UNTIL
( NEW_NBR^.DATA.X = NBR_FIRST.X ) AND
( NEW_NBR^.DATA.Y = NBR_FIRST.Y );
K := K + 1; J := J - 1;
IF K <= NUMLOOP THEN
BEGIN
  NBR_FIRST := NBR (K)^.NEXT^.NEXT^.DATA;
  NEW_NBR^.DATA := NBR_LAST [K]^ .DATA
END (END IF K <= NUM_LOOP)
(*
(*STOP THE LOOP AS ALL LINKED LISTS OF NBR'S AND VTX'S HAD BEEN CREATED*)
(*)
UNTIL
K > NUM_LOOP;
FOR K := 1 TO NUM_LOOP DO
BEGIN
  WRITELN (NBRFILE);
  WRITELN (NBRFILE, ' THE # ', K:3, ' LINKED LIST OF NBR IS:');
  NBR_LAST [K] := NBR (K)^.NEXT;
  WITH NBR_LAST [K]^ .DATA DO
  BEGIN
    WRITELN (NBRFILE, INDEX, ', ', X, ', ', Y, ', ', DIA);
    X1 := X;
    Y1 := Y
  END; (END WITH NBR_LAST [K]^ .DATA)
  WHILE NBR_LAST [K] <> Z DO
  BEGIN
    WITH NBR_LAST [K]^ .DATA DO
    BEGIN
      WRITELN (NBRFILE, INDEX, ', ', X, ', ', Y, ', ', DIA);
      WRITELN (DELAUNAY, X1, Y1, X, Y)
    END; (END WITH NBR_LAST [K]^ .DATA)
  END (END WHILE NBR_LAST [K] <> Z)
END; (END FOR K := 1 TO NUM_LOOP)
FOR K := 1 TO NUM_LOOP DO
BEGIN
  WRITELN (VTXFILE);
  WHILE VTX_LAST [K] <> Z DO
  BEGIN
    WITH VTX_LAST [K]^ .DATA DO
    BEGIN
      WRITELN (VTXFILE, X, ', ', Y);
    END (END WHILE VTX_LAST [K] <> Z)
  END; (END FOR K := 1 TO K DO)
(*
**************************************************************************************************************/
(* THE VORONOI DIAGRAM IS COMPLETED HERE *)
(***************************************************************************************************************/
134
(*) FIND 1 THE 'TOP', 'BOTTOM', 'LEFT', AND 'RIGHT', SUCH THAT  *
(*) THE FOUR LINES Y = TOP, Y = BOTTOM, X = LEFT, X = RIGHT TOGETHER  *
(*) FORMS THE INNER BOX.  *
*******************************************************************************
(*)
(*) SORT THE ARRAY OF LINKED LIST OF NBR'S USING THE INDICES OF CENTER DISKS *)
(*) AS THE KEY. NAME THE SORTED ARRAY 'GNBR'.  *
(*)
(*) FOR K := 1 TO NUM_LOOP DO
BEGIN
J := NBR [K]^.NEXT^.DATA_INDEX;
GNBR [J] := NBR [K]^.NEXT;
NBR_LAST [J] := GNBR [J];
WITH NBR_LAST [J]^.DATA DO
BEGIN
X1 := X; Y1 := Y
END;  (END WITH NBR_LAST [J]^.DATA)
NBR_LAST [J] := NBR_LAST [J]^.NEXT;
REPEAT
WITH NBR_LAST [J]^.DATA DO
BEGIN
X2 := X; Y2 := Y
END;  (END WITH NBR_LAST [J]^.DATA)
NBR_LAST [J]^.DATA_ANGLE := THETA (X1, Y1, X2, Y2);
NBR_LAST [J] := NBR_LAST [J]^.NEXT
UNTIL
NBR_LAST [J] = Z
END;
(*)
(*) FIND THE MINIMUM Y-COORDINATE ON THE TOP BOUNDARY*)
(*) NAME IT 'TOP' TEMPORARILY  *
(*)
(*)
K := CORNER2.INDEX;
NBR_LAST [K] := GNBR [K];
TOP := NBR_LAST [K]^.DATA.Y;
REPEAT
MIN := 180;
REPEAT
ALFA := 90 - NBR_LAST [K]^.DATA_ANGLE;
IF ALFA > 0 THEN
BEGIN
IF ALFA < MIN THEN
BEGIN
MIN := ALFA;
J := NBR_LAST [K]^.DATA_INDEX
END
END;  (END IF ALFA > 0)
ELSE IF ( ALFA < -180 ) AND ( MIN - ALFA > 360 ) THEN
BEGIN
MIN := 360 + ALFA;
J := NBR_LAST [K]^.DATA_INDEX
END
END;
(*)
(*)
(*)
END;
UNTIL
NBR_LAST [K] = Z;
K := J;
NBR_LAST [K] := GNBR [K];
IF NBR_LAST [K]^DATA.Y < TOP THEN
TOP := NBR_LAST [K]^DATA.Y;
UNTIL
K = CORNER3.INDEX;

(*
(*
(*FIND THE MAXIMUM Y-COORDINATE ON THE BOTTOM BOUNDARY*)
(*NAME IT 'BOTTOM' TEMPORARILY*)
(*
(*)
K := CORNER4.INDEX;
NBR_LAST [K] := GNBR [K];
BOTTOM := NBR_LAST [K]^DATA.Y;
REPEAT
MIN := 180;
REPEAT
ALFA := 270 - NBR_LAST [K]^DATA.ANGLE;
IF ( 0 < ALFA ) AND ( ALFA < MIN ) THEN
BEGIN
MIN := ALFA;
J := NBR_LAST [K]^DATA.INDEX
END;
UNTIL
NBR_LAST [K] = Z;
K := J;
NBR_LAST [K] := GNBR [K];
IF NBR_LAST [K]^DATA.Y > BOTTOM THEN
BOTTOM := NBR_LAST [K]^DATA.Y;
UNTIL
K = CORNER1.INDEX;

(*
(*
(*FIND THE MAXIMUM X-COORDINATE ON THE LEFT BOUNDARY*)
(*NAME IT 'LEFT' TEMPORARILY*)
(*
(*)
K := CORNER1.INDEX;
NBR_LAST [K] := GNBR [K];
LEFT := NBR_LAST [K]^DATA.X;
REPEAT
MIN := 180;
REPEAT
ALFA := 180 - NBR_LAST [K]^DATA.ANGLE;
IF ( 0 < ALFA ) AND ( ALFA < MIN ) THEN
BEGIN
MIN := ALFA;
END;
136
J := NBR_LAST [K]^..DATA_INDEX
END;
UNTIL
NBR_LAST [K] = Z;
K := J;
NBR_LAST [K] := GNBR [K];
IF NBR_LAST [K]^..DATA.X > LEFT THEN
  LEFT := NBR_LAST [K]^..DATA.X;
UNTIL
K = CORNER2_INDEX;
(*
(*)
("FIND THE MINIMUM X-COORDINATE ON THE RIGHT BOUNDARY")
("NAME IT 'RIGHT' TEMPORARILY")
(*
*)
K := CORNER3_INDEX;
NBR_LAST [K] := GNBR [K];
RIGHT := NBR_LAST [K]^..DATA.X;
REPEAT
MIN := 180;
REPEAT
ALFA := - NBR_LAST [K]^..DATA.ANGLE;
IF (ALFA < -180) AND (MIN - ALFA > 360) THEN
BEGIN
MIN := 360 + ALFA;
J := NBR_LAST [K]^..DATA_INDEX
END;
UNTIL
NBR_LAST [K] = Z;
K := J;
NBR_LAST [K] := GNBR [K];
IF NBR_LAST [K]^..DATA.X < RIGHT THEN
  RIGHT := NBR_LAST [K]^..DATA.X;
UNTIL
K = CORNER4_INDEX;
WRITELN (BOX);
WRITELN (BOX, ' MINIMUM DISTANCE BETWEEN THE TOP AND BOTTOM BOUNDARIES: ',
  TOP - BOTTOM);
WRITELN (BOX, ' MINIMUM DISTANCE BETWEEN THE LEFT AND RIGHT BOUNDARIES: ',
  RIGHT - LEFT);
IF TOP - BOTTOM < 4 * DMAX THEN
BEGIN
WRITELN (BOX);
WRITELN (BOX, ' NOT EXECUTABLE ENSEMBLE!' );
WRITELN (BOX, ' NO INNER BOX CAN BE FIXED FOR THE GIVEN ENSEMBLE ' );
WRITELN (BOX, ' BECAUSE THE MINIMUM DISTANCE BETWEEN THE TOP AND ' );
WRITELN (BOX, ' BOTTOM BOUNDARIES IS TOO SMALL.' );
WRITELN (BOX, ' BYE!' );
WRITELN (BOX);
GOTO 100
END;  \{END IF TO - BOTTOM < 4 * DMAX\}
IF RIGHT - LEFT < 4 * DMAX THEN
BEGIN
WRITELN (BOX);
WRITELN (BOX, ' NOT EXECUTABLE ENSEMBLE!');
WRITELN (BOX, ' NO INNER BOX CAN BE FIXED FOR THE GIVEN ENSEMBLE.');
WRITELN (BOX, ' BECAUSE THE MINIMUM DISTANCE BETWEEN THE RIGHT AND');
WRITELN (BOX, ' LEFT BOUNDARIES IS TOO SMALL.');
WRITELN (BOX, ' BYE!');
WRITELN (BOX);
GOTO 100
END;  \{END IF RIGHT - LEFT < 4 * DMAX\}

(*
  *)
(*GENERATE A INNER BOX WITH 'HEIGHT' AND 'WIDTH'*
(*
  *)
Y1 := BOTTOM;
Y2 := TOP;
X1 := LEFT;
X2 := RIGHT;

YC := (Y1 + Y2) / 2;
XC := WIDE / 2;

IF (XC - X1 < 2 * DMAX) OR (X2 - XC < 2 * DMAX) THEN
BEGIN
WRITELN (BOX);
WRITELN (BOX, ' NOT EXECUTABLE ENSEMBLE!');
WRITELN (BOX, ' NO INNER BOX CAN BE FIXED FOR THE GIVEN ENSEMBLE.');
WRITELN (BOX, ' BECAUSE THE MINIMUM DISTANCE BETWEEN THE CENTER AND');
WRITELN (BOX, ' EITHER THE LEFT OR THE RIGHT BOUNDARY IS TOO SMALL.');
WRITELN (BOX, ' BYE!');
WRITELN (BOX);
GOTO 100
END;  \{END IF\}

BOTTOM := YC - HEIGHT / 2;
TOP := YC + HEIGHT / 2;
LEFT := XC - WIDTH / 2;
RIGHT := XC + WIDTH / 2;

STOP := FALSE;
IF (BOTTOM < Y1 + 2 * DMAX) OR (TOP > Y2 - 2 * DMAX) THEN
BEGIN
STOP := TRUE;
WRITELN (BOX);
WRITELN (BOX, ' THE HEIGHT OF THE INNER BOX SELECTED IS TOO ',
'LARGE.');
WRITELN (BOX, ' CHOOSE A HEIGHT SMALLER THAN ',
'\{Y2 - Y1 - 4 * DMAX, ' AND RUN AGAIN.'\})
END;  \{END IF\}
IF (LEFT < X1 + 2 * DMAX) OR (RIGHT > X2 - 2 * DMAX) THEN
BEGIN
STOP := TRUE;
WRITELN (BOX);
WRITELN (BOX, ' THE WIDTH OF THE INNER BOX SELECTED IS TOO ',
' LARGE.');
IF (XC - X1 < X2 - XC) THEN
WRITELN (BOX, ' CHOOSE A WIDTH SMALLER THEN ',
2 * (XC - X1 - 2 * DMAX), ' AND RUN AGAIN.'),
ELSE
WRITELN (BOX, ' CHOOSE A WIDTH SMALLER THEN ',
2 * (X2 - XC - 2 * DMAX), ' AND RUN AGAIN.');
END; (END IF)
IF DEFINE_PF_BOX THEN
BEGIN
IF PF_L < 0 THEN
BEGIN
STOP := TRUE;
WRITELN (BOX);
WRITELN (BOX, 'PF_L IS TOO SMALL.');
WRITELN (BOX, 'CHOOSE A PF_L NO LESS THAN 0 AND RUN AGAIN.');
END; (END IF)
IF PF_R > WIDE THEN
BEGIN
STOP := TRUE;
WRITELN (BOX);
WRITELN (BOX, 'PF_R IS TOO LARGE.');
WRITELN (BOX, 'CHOOSE A PF_R NO GREATER THAN ', WIDE,
' AND RUN AGAIN.');
END; (END IF)
IF PF_B < 0 THEN
BEGIN
STOP := TRUE;
WRITELN (BOX);
WRITELN (BOX, 'PF_B IS TOO SMALL.');
WRITELN (BOX, 'CHOOSE A PF_B NO LESS THAN 0 AND RUN AGAIN.');
END; (END IF)
IF PF_T > Y2 - 2 * DMAX THEN
BEGIN
STOP := TRUE;
WRITELN (BOX);
WRITELN (BOX, 'PF_T IS TOO LARGE.');
WRITELN (BOX, 'CHOOSE A PF_T NO GREATER THAN ', Y2 - 2 * DMAX,
' AND RUN AGAIN.');
END (END IF)
END; (END IF DEFINE_PF_BOX)
IF STOP THEN
GOTO 100
ELSE
BEGIN
WRITELN (BOX);
WRITELN (BOX, ' THE EXTREMA OF THE INNER BOX IS:');
WRITELN (BOX);
WRITEU (BOX, ' XMIN = ', LEFT, ' XMAX = ', RIGHT);
WRITEU (BOX, ' YMIN = ', BOTTOM, ' YMAX = ', TOP);
IF DEFINE_PF_BOX THEN
  BEGIN
    WRITEU (BOX);
    WRITEU (BOX,' $THE EXTREMA OF THE BOX FOR CALCULATING P.F. IS:');
    WRITEU (BOX);
    WRITEU (BOX, ' PF_L = ', PF_L, ' PF_R = ', PF_R);
    WRITEU (BOX, ' PF_B = ', PF_B, ' PF_T = ', PF_T)
  END (END IF DEFINE_PF_BOX)
END; (END ELSE - NOT STOP)

*******************************************************************************
(* 1) FIND ARRAY OF LINKED LIST OF GEOMETRIC NEIGHBOURS FOR INNER DISKS, *)
(* NAME THE ARRAY 'INNERG' *)
(* 2) FIND THE NUMBER DISTRIBUTION AND EXPECTED NUMBER OF GEOMETRIC *)
(* NEIGHBOUR FOR INNER DISKS *)
*******************************************************************************
LAST := INNER;
K := 1;
REPEAT
  J := LAST^.DATA_INDEX;
  INNERG [K] := GNBR [J];
  NUM_GN [K] := 0;
  REPEAT
    NUM_GN [K] := NUM_GN [K] + 1
  UNTIL
    NBR_LAST [K] = Z;
  LAST := LAST^.NEXT;
  K := K + 1
  UNTIL
  LAST = Z;
  NUM_IN := K - 1;

WRITEU (BOX);
WRITEU (BOX, ' NUMBER OF INNER DISKS FOUND EXISTING WITHIN OR ON THE ',
           'INNER BOX: ', NUM_IN);
NUM_DST (NUM_GN, NUM_IN, NMIN, NMAX, DST);
NUM_DST_GN := DST;
SUM := 0;
FOR K := NMIN TO NMAX DO SUM := SUM + NUM_DST_GN [K] * K;
AVG_GN := SUM / NUM_IN;
IF SQRT ( SQR ( X2 - X1 ) + SQR ( Y2 - Y1 ) ) <
( 1 + TOLERANCE ) * ( DIA1 / 2 + DIA2 / 2 ) THEN
BEGIN
NEW ( NBR_TEMP);
NBR_TEMP^.NEXT := Z;
NBR_TEMP^.DATA := NBR_LAST [K]^.DATA;
SNBR_LAST [K] := NBR_TEMP;
NUM_SN [K] := NUM_SN [K] + 1
END;
UNTIL
NBR_LAST [K] = Z
END;  (END FOR K := 1 TO NUM_IN)
FOR K := 1 TO NUM_IN DO
BEGIN
WRITELN (SNBRFILE);
WRITELN (SNBRFILE, ' THE # ', K:5, ' LINKED LIST OF STRUCTURAL',
' NEIGHBOUR FOR INNER DISKS IS:');

(*1) PICK OUT STRUCTURAL NEIGHBOURS FROM GEOMETRICAL NEIGHBOURS, FORM ARRAY*)
(*2) FIND THE NUMBER DISTRIBUTION AND EXPECTED NUMBER OF STRUCTURAL *)
(*3) FIND THE ANGULAR DISTRIBUTION AND EXPECTED ANGLE OF STRUCTURAL *)
(************************************************************************************)
FOR K := 1 TO NUM_IN DO
BEGIN
NBR_LAST [K] := INNERG [K];
WITH NBR_LAST [K]^.DATA DO
BEGIN
X1 := X;
Y1 := Y;
DIA1 := DIA
END;  (END WITH NBR_LAST [K]^.DATA)
NEW (INNERS [K]);
INNERS [K]^.NEXT := Z;
SNBR_LAST [K] := INNERS [K];
NUM_SN [K] := 0;
REPEAT
WITH NBR_LAST [K]^.DATA DO
BEGIN
X2 := X;
Y2 := Y;
DIA2 := DIA
END;  (END WITH NBR_LAST [K]^.DATA)
IF SQRT ( SQR ( X2 - X1 ) + SQR ( Y2 - Y1 ) ) <
( 1 + TOLERANCE ) * ( DIA1 / 2 + DIA2 / 2 ) THEN
BEGIN
NEW ( NBR_TEMP);
NBR_TEMP^.NEXT := Z;
NBR_TEMP^.DATA := NBR_LAST [K]^.DATA;
SNBR_LAST [K]^.NEXT := NBR_TEMP;
NUM_SN [K] := NUM_SN [K] + 1
END;
UNTIL
NBR_LAST [K] = Z
END;  (END FOR K := 1 TO NUM_IN)
FOR K := 1 TO NUM_IN DO
BEGIN
WRITELN (SNBRFILE);
WRITELN (SNBRFILE, ' THE # ', K:5, ' LINKED LIST OF STRUCTURAL',
' NEIGHBOUR FOR INNER DISKS IS:');

141
SNBR_LAST[K] := INNERS[K];
WITH SNBR_LAST[K]\.DATA DO
  WRITELN (SNBRFILE, INDEX, ' ', X, ' ', Y, ' ', DIA);
SNBR_LAST[K] := SNBR_LAST[K]\.NEXT;
WHILE SNBR_LAST[K] <> 2 DO
  BEGIN
    WITH SNBR_LAST[K]\.DATA DO
      WRITELN (SNBRFILE, INDEX, ' ', X, ' ', Y, ' ', DIA, ' ', ANGLE);
SNBR_LAST[K] := SNBR_LAST[K]\.NEXT
  END  (END WHILE SNBR_LAST[K] <> 2)
END;  (END FOR K := 1 TO NUM_IN)
SUM := 0;
NUM_DST (NUM_SN, NUM_IN, SMIN, SMAX, DST);
NUM_DST_SN := DST;
FOR K := SMIN TO SMAX DO
  SUM := SUM + NUM_DST_SN[K] * K;
AVG_SN := SUM / NUM_IN;

******************************************************************************
(* 1. COMPUTE THE MEAN NEAREST NEIGHBOUR RADIUS *)
(* AND THE MEDIAN NEAREST NEIGHBOUR RADIUS *)
(* IN CASE WHETHER THE GIVEN ENSEMBLE IS A RANDOM CLOSE PACKING *)
(* OR NOT IS UNKNOWN. *)
(* *)
(* 2. FIND THE CUMULATIVE PROBABILITY OF THE DISTRIBUTION OF *)
(* NEAREST NEIGHBOUR RADIUS IF THE GIVEN ENSEMBLE ARE IN SAME SIZE. *)
******************************************************************************
IF NOT KNOWN_TO_BE_RCP THEN
  BEGIN
    FIND_NNR (INNERS, GNBR, NUM_IN, NNR);
    SUM := 0;
    FOR K := 1 TO NUM_IN DO
      SUM := SUM + NNR[K];
    MEAN_NNR := SUM / NUM_IN;
    WRITELN (BOX);
    WRITELN (BOX, ' $MEAN NEAREST NEIGHBOUR RADIUS IS: ', MEAN_NNR);
    ARRAY_SORT (NNR, NUM_IN);
    IF NUM_IN - NUM_IN DIV 2 * 2 = 0 THEN
      MEDIAN_NNR := NNR[NUM_IN DIV 2]
    ELSE
      MEDIAN_NNR := NNR[NUM_IN DIV 2 + 1];
    WRITELN (BOX);
    WRITELN (BOX, ' $MEDIAN NEAREST NEIGHBOUR RADIUS IS: ', MEDIAN_NNR);
    IF EQ_DISKS THEN
      IF MEDIAN_NNR > (1 + TOLERANCE) * DMAX THEN
        BEGIN
          WRITELN (BOX, ' WHICH IS GREATER THEN (1 + TOLERANCE) * ',
                    'DMAX.');
          WRITELN (BOX, ' IT IS THUS SHOWN THAT THE GIVEN ENSEMBLE IS',
                    ' NOT A RANDOM CLOSE PACKING!');
          WRITELN (BOX, ' BYE!');
          GOTO 100
        END  (END IF MEDIAN_NNR > (1 + TOLERANCE) * DMAX)
ELSE

BEGIN

CP_NNR[0] := 0;
DIVISION := (MEDIAN_NNR - DMAX) / 5;
MIN := DMAX + DIVISION;
K := 1;
J := 0;
I := 0;
REPEAT

I := I + 1;
WHILE (NRR[K] <= MIN) AND (K <= NUM_IN) DO

K := K + 1;
CP_NNR[I] := CP_NNR[I - 1] + (K - J) / NUM_IN;
J := K - 1;
MIN := MIN + DIVISION

UNTIL

(CP_NNR[I] > 0.99999) OR (I = 100);
NUM_RAD_DIV := I

END (END ELSE)

END; (END IF NOT KNOWN_TO_BE_RCP)

(*
  *)
(*SORT EACH ELEMENT IN THE ARRAY OF LINKED LIST 'INNERS', SUCH*
(*THAT THE STRUCTURAL NEIGHBOURS IN EACH ELEMENT ARE LINKED TO*"
(*THE CENTER DISK IN INCREASING ANGLE SEQUENCE
(*
(*
FOR K := 1 TO NUM_IN DO

BEGIN

ALFA := NBR_LAST [K]^.DATA.ANGLE;
STOP := FALSE;
WHILE (NBR_LAST [K]^.NEXT <> Z) AND (NOT STOP) DO

IF ALFA < NBR_LAST [K]^.NEXT^.DATA.ANGLE THEN

BEGIN

STOP := TRUE;
NBR_TEMP := NBR_LAST [K]^.NEXT;
NBR_LAST [K]^.NEXT := Z;
NBR_LAST [K] := NBR_TEMP;
WHILE NBR_LAST [K]^.NEXT <> Z DO

INNERS [K]^.NEXT := NBR_TEMP

END (END IF ALFA < NBR_LAST [K]^.NEXT^.DATA.ANGLE)

ELSE


END; (END FOR K := 1 TO NUM_IN)

(*
(*1) FORM ARRAY 'GROUP_SN [K]' WHICH GROUP INNER DISKS WITH THE SAME NUMBER*
(* 'K', OF STRUCTURAL NEIGHBOUR TOGETHER.
(*
(* NOTE: DISKS WITH 0 STRUCTURAL NEIGHBOURS ARE EXCLUDED.
(*
(*2) GET 'GROUP_SUM [K]', THE NUMBER OF DISK WITH K STRUCTURAL NEIGHBOUR*
(* THAT ARE GROUPED IN GROUP_SN [K].
(*
FOR K := 1 TO 6 DO

GROUP_SUM [K] := 0;

143
FOR K := 1 TO NUM_IN DO
CASE NUM_SN [K] OF
 1: BEGIN
    GROUP_SN [1, GROUP_SUM [1]] := INNERS [K]
END;  (END CASE NUM_SN [K] OF 1)
 2: BEGIN
    GROUP_SN [2, GROUP_SUM [2]] := INNERS [K]
END;  (END CASE NUM_SN [K] OF 2)
 3: BEGIN
    GROUP_SN [3, GROUP_SUM [3]] := INNERS [K]
END;  (END CASE NUM_SN [K] OF 3)
 4: BEGIN
    GROUP_SN [4, GROUP_SUM [4]] := INNERS [K]
END;  (END CASE NUM_SN [K] OF 4)
 5: BEGIN
    GROUP_SN [5, GROUP_SUM [5]] := INNERS [K]
END;  (END CASE NUM_SN [K] OF 5)
 6: BEGIN
    GROUP_SN [6, GROUP_SUM [6]] := INNERS [K]
END;  (END CASE NUM_SN [K] OF 6)
END;  (END CASE)

(*)
(*FOR EACH GROUP OF DISKS WITH K STRUCTURAL NEIGHBOURS, FIND THE EXPECTED*)
(*ANGLE FOR THE JTH STRUCTURAL NEIGHBOUR. WHERE J IS FROM 1 TO K. *)
(*

ANG_DST (GROUP_SN, GROUP_SUM, SMIN, SMAX, NUM_ANG_DIV, DST_SN);
ANG_DST_SN := DST_SN;
FOR K := SMIN TO SMAX DO
BEGIN
  FOR J := 1 TO GROUP_SUM [K] DO
  FOR J := 1 TO K DO
  BEGIN
    SUM := 0;
    FOR I := 1 TO GROUP_SUM [K] DO
    BEGIN
      SUM := SUM + NBR_LAST [I]^..DATA_ANGLE;
      NBR_LAST [I] := NBR_LAST [I]^..NEXT
    END;  (END FOR I := 1 TO GROUP_SUM [K])
    EXPECT_ANGLE [K, J] := SUM / GROUP_SUM [K]
  END;  (END FOR J := 1 TO K)
END;  (END FOR K := SMIN TO SMAX)

FOR K := SMIN TO SMAX DO
BEGIN
  FOR J := 1 TO GROUP_SUM [K] DO

END

FOR J := 1 TO K DO
BEGIN
  SUM := 0;
  SUMS := 0;
  FOR I := 1 TO GROUP_SUM [K] DO
  BEGIN
    SUM := SUM + NBR_LAST [I]^ . DATA . ANGLE ;
    SUMS := SUMS + SQR ( NBR_LAST [I]^ . DATA . ANGLE );
    NBR_LAST [I] := NBR_LAST [I]^ . NEXT
  END; (END FOR I := 1 TO GROUP_SUM [K])
  IF GROUP_SUM [K] > 1 THEN
    STD_DVA [K, J] := SQRT ( ABS
    ( (SUMS - SQR (SUM)) / GROUP_SUM [K]) / (GROUP_SUM [K] - 1) )
  END; (END FOR J := 1 TO K)
END; (END FOR K := SMIN TO SMAX)

IF SMIN < NMIN THEN NMIN := SMIN;
IF SMAX > NMAX THEN NMAX := SMAX;

("**************************************************************************
 (*                             MAKE OUTPUT FILE TABLE.DAT
 "**************************************************************************
)
WRITELN (TABLE);
WRITELN (TABLE, "**************************************************************************");
WRITELN (TABLE, "PARAMETERS");
WRITELN (TABLE, "**************************************************************************");
WRITELN (TABLE);
WRITELN (TABLE, ' NUMBER OF DISK      : ', NUM_DISK);
WRITELN (TABLE, ' TOLERANCE       : ', TOLERANCE);
WRITELN (TABLE, ' HEIGHT OF THE INNER BOX: ', HEIGHT);
WRITELN (TABLE, ' WIDTH OF THE INNER BOX : ', WIDTH);
WRITELN (TABLE, ' AREA OF THE INNER BOX  : ', HEIGHT * WIDTH);
WRITELN (TABLE);
WRITELN (TABLE, "**************************************************************************");
WRITELN (TABLE, "INFORMATIONS ON INNER DISKS: ");
WRITELN (TABLE, "**************************************************************************");
WRITELN (TABLE);
WRITELN (TABLE, ' INDEX # OF G.W. GEOMETRIC NBRS # OF S.W. ',
      ' STRUCTURAL NBRS THETA');
WRITELN (TABLE);
LAST := INNER;
FOR K := 1 TO NUM_IN DO
BEGIN
  WRITELN (TABLE, "**************************************************************************");
END; (END FOR K := 1 TO NUM_IN)
145
WRITELN (TABLE, ' ', LAST^DATA.INDEX:5, 
', ' , NUM_GN [K]:1, 
', ' , NBR_LAST [K]^DATA. INDEX:5, 
', ' , NUM_SN [K]:1, 
', ' , SNBR_LAST [K]^DATA. INDEX:5, 
', ' , SNBR_LAST [K]^DATA. ANGLE);
WHILE SNBR_LAST [K] <> Z DO 
BEGIN
  WRITELN (TABLE, ' :25, 
               NBR_LAST [K]^DATA. INDEX:5, ' :23, 
               SNBR_LAST [K]^DATA. INDEX:5, ' :5, 
               SNBR_LAST [K]^DATA. ANGLE);
END; (END WHILE SNBR_LAST [K] <> Z)
WHILE NBR_LAST [K] <> Z DO
BEGIN
  WRITELN (TABLE, ' :20, 
               NBR_LAST [K]^DATA. INDEX);
END; (END WHILE NBR_LAST [K] <> Z)
LAST := LAST^NEXT
END; (FOR K := 1 TO NUM_IN)
WRITELN (TABLE);
WRITELN (TABLE, '-------------------------------', 
              '-------------------------------');
WRITELN (TABLE);
WRITELN (TABLE, ' MEAN ', AVG_GN, ' ', AVG_SN);
WRITELN (TABLE);
WRITELN (TABLE, '*************************************************************************', 
              '*************************************************************************');
WRITELN (TABLE, 'CUMULATIVE PROBABILITY OF THE RADIAL DISTRIBUTION', 
              '*');
WRITELN (TABLE, 'OF THE NEAREST NEIGHBOUR RADIUS', 
              '*');
WRITELN (TABLE, '*************************************************************************', 
              '*************************************************************************');
WRITELN (TABLE);
WRITELN (TABLE, 'RADIAL RANGE CUMULATIVE PROBABILITY');
WRITELN (TABLE);
WRITELN (TABLE, '-------------------------------', 
              '-------------------------------');
MIN := DMAX;
FOR K := 1 TO NUM_RAD_DIV DO 
BEGIN
  WRITELN (TABLE, ' ', MIN, ' ~ ', MIN + DIVISION, 
            ' ', CP_NNR(K));
  MIN := MIN + DIVISION
END;
WRITELN (TABLE);
WRITELN (TABLE, '********************************************************************',
'********************************************************************');
WRITELN (TABLE, '*
  DISTRIBUTION OF THE NUMBER OF GEOMETRIC ',
  ' *
');
WRITELN (TABLE, '*
  AND STRUCTURAL NEIGHBOUR FOR INNER DISKS. ',
  ' *
');
WRITELN (TABLE, '********************************************************************',
'********************************************************************');
WRITELN (TABLE);
WRITELN (TABLE, '  NUMBER', ' ', ' GEOMETRIC NEIGHBOUR', ' ',
  ' STRUCTURAL NEIGHBOUR');
WRITELN (TABLE);
WRITELN (TABLE, '----------------------------------------',
'----------------------------------------');
FOR K := NMIN TO NMAX DO
  WRITELN (TABLE, K:6, ' ', NUM_DST_GN [K],
  ' ', NUM_DST_SN [K]);
WRITELN (TABLE, '----------------------------------------',
'----------------------------------------');
WRITELN (TABLE, ' TOTA', ' ', NUM_IN:10,
  ' ', NUM_IN:10);
WRITELN (TABLE);
WRITELN (TABLE, '********************************************************************',
'********************************************************************');
WRITELN (TABLE, ' THE MEAN VALUE OF ANGLE ASSOCIATED WITH EACH SEQUENCIAL ',
  ' STRUCTURAL NEIGHBOUR * ');
WRITELN (TABLE, '********************************************************************',
'********************************************************************');
WRITELN (TABLE);
WRITELN (TABLE, '  NUMBER OF ', ' STRUCTURAL NEIGHBOUR IN ',
  ' SEQUENCE');
WRITELN (TABLE, '  STRUCTURAL ', ' ', ' NEIGHBOUR ', ' 1 2 3 ',
  ' 4 5 6');
WRITELN (TABLE, '----------------------------------------',
'----------------------------------------');
FOR K := SMIN TO SMAX DO
  BEGIN
    IF K = 1 THEN
      BEGIN
        WRITE (TABLE, ' ', K:1);
        WRITELN (TABLE, ' ', EXPECT_ANGLE [K, 1]:10)
      END  (END IF K = 1)
    ELSE IF K > 1 THEN
      BEGIN
        WRITE (TABLE, ' ', K:1);
        WRITE (TABLE, ' ', EXPECT_ANGLE [K, 1]:10);
        FOR J := 2 TO K - 1 DO
          WRITE (TABLE, ' ', EXPECT_ANGLE [K, J]:10);
        WRITELN (TABLE, ' ', EXPECT_ANGLE [K, K]:10)
      END  (ELSE IF K > 1)
  END;  (END FOR K := SMIN TO SMAX)
WRITELN (TABLE);
WRITELN (TABLE, '******************************************************************************,
'  '******************************************************************************);
WRITELN (TABLE, 'THE STANDARD DEVIATION OF THE ANGLE ASSOCIATED,
'  '*');
WRITELN (TABLE, 'WITH EACH SEQUENCIAL STRUCTURAL NEIGHBOUR ',
'  '*');
WRITELN (TABLE, '******************************************************************************,
'  '******************************************************************************);
WRITELN (TABLE);
WRITELN (TABLE, 'NUMBER OF STRUCTURAL NEIGHBOUR IN',
'  'SEQUENCE');
WRITELN (TABLE, 'STURCUTURAL ---------------',
'  '----------------');
WRITELN (TABLE, 'NEIGHBOUR 1 2 3 ',
'  4 5 6');
WRITELN (TABLE, '-------------------',
'  -------------------');
FOR K := SMIN TO SMAX DO
  IF GROUP_SUM [K] > 1 THEN
    BEGIN
      WRITE (TABLE, ' ', K:1);
      IF K = 1 THEN
        WRITELN (TABLE, ', STD_DVA [K, 1]:10)
      ELSE IF K > 1 THEN
        BEGIN
          WRITE (TABLE, ', STD_DVA [K, 1]:10);
          FOR J := 2 TO K - 1 DO
            WRITE (TABLE, ', STD_DVA [K, J]:10);
          WRITELN (TABLE, ', STD_DVA [K, K]:10)
        END
      END
    END
  END;
WRITELN (TABLE);
WRITELN (TABLE, '******************************************************************************,
'  '******************************************************************************);
WRITELN (TABLE, 'ANGULAR DISTRIBUTION OF STRUCTURAL NEIGHBOURS',
'  '*');
WRITELN (TABLE, '******************************************************************************,
'  '******************************************************************************);
GAMMA := 360 / NUM_ANG Div;
FOR K := SMIN TO SMAX DO
  FOR J := 1 TO K DO
    BEGIN
      WRITELN (TABLE);
      WRITELN (TABLE, 'THE ANGULAR DISTRIBUTION OF # ', J:1,
'  STRUCTURAL NEIGHBOUR');
      WRITELN (TABLE, 'IN DISKS WITH ', K:1,
'  STRUCTURAL NEIGHBOURS');
      WRITELN (TABLE, '-------------------',
'  -------------------');
      WRITELN (TABLE, 'ANGLE RANGE', ' ',
'  # OF STRUCTURAL NEIGHBOUR');
    END
END.
WRITE (TABLE, '----------------------------------------------------------------------------------',
'----------------------------------------------------------------------------------');
ALFA := 0;
BETA := GAMMA;
FOR I := 0 TO NUM_ANG_DIV - 1 DO
BEGIN
  WRITE (TABLE, ALFA, ' - ', BETA, ' - ',
        ANG_DST_SN (K, J, I));
  ALFA := BETA;
  BETA := ALFA + GAMMA
END;  (END FOR I := 0 TO NUM_ANG_DIV - 1)
WRITE (TABLE, '----------------------------------------------------------------------------------',
'----------------------------------------------------------------------------------');
WRITE (TABLE, ' TOTAL', ': ', NUM_DST_SN (K));
WRITE (TABLE)
END;  (END FOR J := 1 TO K)

******************************************************************************
(* FIND THE PACKING FRACTION *)
******************************************************************************
WRITE (TABLE);
WRITE (TABLE, '----------------------------------------------------------------------------------',
'----------------------------------------------------------------------------------');
WRITE (TABLE, ' PACKING FRACTION ',
        ' ');{**}
WRITE (TABLE, '----------------------------------------------------------------------------------',
'----------------------------------------------------------------------------------');
WRITE (TABLE);
WRITE (TABLE, ' PF_L  PF_R  PF_B  PF_T  INNER_DISK',
        ' AREA PACKING FRACTION');
WRITE (TABLE);
WRITE (TABLE, '----------------------------------------------------------------------------------',
'----------------------------------------------------------------------------------');
IF DEFINE_PF_BOX THEN
BEGIN
  (*
  (*)
  (*DISPOSE THE LINKED LIST SORT_COPY*)
  (*)
  LAST := INNER;
  WHILE LAST <> Z DO
  BEGIN
    NBR_TEMP := LAST;
    LAST := LAST^.NEXT;
    DISPOSE (NBR_TEMP)
  END;  (END WHILE LAST <> Z)

  LEFT := PF_L;
  RIGHT := PF_R;
  BOTTOM := PF_B;
  TOP := PF_T;
  MAKE_COPY (SORT, SORT_COPY);
  FIND_INNER (SORT_COPY, LEFT, RIGHT, BOTTOM, TOP, INNER);
  FIND_PF (INNER, LEFT, RIGHT, BOTTOM, TOP, ETA, SUM);
WRITELN (TABLE);
WRITELN (TABLE, ' PACKING FRACTION WITHIN USER DEFINED PF_BOX:');
WRITELN (TABLE);
WRITELN (TABLE, LEFT, RIGHT, BOTTOM, TOP, ' ', SUM, ' ', ETA);
END (END IF DEFINE_PF_BOX)
ELSE
BEGIN
WRITELN (TABLE);
WRITELN (TABLE, ' PACKING FRACITON WITHIN THE INNER BOX:');
WRITELN (TABLE);
FIND_PF (INNER, LEFT, RIGHT, BOTTOM, TOP, ETA, SUM);
WRITELN (TABLE, LEFT, RIGHT, BOTTOM, TOP, ' ', SUM, ' ', ETA);
END; (END ELSE)

IF EQ_DISKS AND FIND_WALL_EFFECT THEN
BEGIN
WRITELN (TABLE);
WRITELN (TABLE, ' PACKING FRACTION VARIATION BY WALL EFFECT:');
WRITELN (TABLE);
LEFT := 0;
RIGHT := DMAX;

(*
   *)

(*DISPOSE THE LINKED LIST SORT_COPY*)

(*
   *)
REPEAT
   LAST := INNER;
   WHILE LAST <> Z DO
   BEGIN
   NBR_TEMP := LAST;
   LAST := LAST^.NEXT;
   DISPOSE (NBR_TEMP)
   END; (END WHILE LAST <> Z)

MAKE_COPY (SORT, SORT_COPY);
FIND_INNER (SORT_COPY, LEFT, RIGHT, BOTTOM, TOP, INNER);
FIND_PF (INNER, LEFT, RIGHT, BOTTOM, TOP, ETA, SUM);
WRITELN (TABLE, LEFT, RIGHT, BOTTOM, TOP, ' ', SUM, ' ', ETA);
RIGHT := RIGHT + 0.001 * WIDE
UNTIL
RIGHT > WIDE;

LAST := INNER;
WHILE LAST <> Z DO
BEGIN
   NBR_TEMP := LAST;
   LAST := LAST^.NEXT;
   DISPOSE (NBR_TEMP)
   END; (END WHILE LAST <> Z)

RIGHT := WIDE;
MAKE_COPY (SORT, SORT_COPY);
FIND_INNER (SORT_COPY, LEFT, RIGHT, BOTTOM, TOP, INNER);
FIND_PF (INNER, LEFT, RIGHT, BOTTOM, TOP, ETA, SUM);
WRITELN (TABLE, LEFT, RIGHT, BOTTOM, TOP, ', ', SUM, ', ', ETA)
END;  (END IF EQ_DISKS)
100: CLOSE (DISKIN);
CLOSE (NBRFILE);
CLOSE (VTXFILE);
CLOSE (BOX);
CLOSE (DELAUNAY);
CLOSE (SNBRFILE);
CLOSE (TABLE)
END.
References


