2006

AN EXPANDING POLYGON BASED METHOD FOR MINIMUM-ZONE STRAIGHTNESS EVALUATION

Horatiu G. Cociu
Western University

AN EXPANDING POLYGON BASED METHOD FOR MINIMUM-ZONE STRAIGHTNESS EVALUATION

(Spine title: Expanding polygons for minimum-zone straightness evaluation)

(Thesis format: Monograph)

by

Horatiu G. Cociu

Graduate Program in Engineering Science
Department of Mechanical and Materials Engineering

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Engineering Science

Faculty of Graduate Studies
The University of Western Ontario
London, Ontario, Canada

THE UNIVERSITY OF WESTERN ONTARIO
FACULTY OF GRADUATE STUDIES

CERTIFICATE OF EXAMINATION

The thesis by

Horatiu Gheorghita Cociu

entitled:

An expanding polygon based method for minimum-zone straightness evaluation

is accepted in partial fulfillment of the
requirements for the degree of
Master of Engineering Science

Date__________________________

____
Chair of the Thesis Examination Board

ABSTRACT

A novel algorithm for accurate and efficient evaluation of straightness is presented in this thesis. The algorithm starts with an initial solution and verifies if it is the global solution, based on the minimum zone for a subset of points rule as well as the proposed polygon expansion method. If the initial solution is not global, a final solution is computed by the polygon expansion method. It is based on the construction of convex polygons including the furthest point from the corresponding minimum zone edge of the polygon and thereby ensuring that, all the vertices of the polygon are vertices of the convex hull for the same set of points. Being a computational geometric approach, the present method always guarantees the minimum zone.

Moreover the expected computational complexity of the proposed method is less than the complexity of $O(n \log n)$ of the existing techniques based on the convex hull. These techniques construct the convex hull for the set of points and then evaluate its edges to find the minimum zone whereas the convex polygon in the proposed method is only a part of the convex hull for the same points.

The same data sets obtained from previous work, were used to determine the accuracy of the present method which matches with the best results published so far. From the simulated data, the efficiency is validated by the polygon size which is up to four times smaller than the convex hull size for uniformly distributed points in rectangle.

Keywords: computational geometry; minimum zone; straightness; convex hull.

ACKNOWLEDGEMENTS

It is rarely the case that a book, a paper or a thesis comes together without the author receiving a great deal of help and support. This one is no exception.

My greatest appreciation goes to my supervisor, Professor Hsi-Yung (Steve) Feng, who so kindly gave his time, knowledge and dedication to guide me to the finish of this research work. For the past two years he has been my mentor, my inspiration and my supportive friend. His advice, support and encouragement helped me develop important skills that will be essential in my future career.

A special thank you to my colleagues: Remus Tutunea-Fatan, Srikanth Durga, Avisekh Banerjee, Hao Song, for their continuing support and encouragement. I would also like to thank Avisekh Banerjee for his valuable suggestion and feedback.

Thanks also go to Cristian Alexandru Popescu and Oana Bogdanel who were always there when I needed them.

At last, but by no means least, a special thank you to my family for their support, encouragement and love.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

An error represents the difference between a computed, estimated, or measured value and the true, specified, or theoretically correct value. In other words it is a deviation from a correct value caused by an irregularity in a system or a functional unit. Manufactured features are different from the theoretically designed features due to presence of many types of errors. The designer specifies a tolerance in which the feature must be included to be accepted.

The most common types of errors are related to form, size and position of the manufactured parts with respect to other part features. Straightness, flatness, circularity and cylindricity represent the form errors and apart from the straightness which has been discussed later in details, the other form errors are defined by the ASME Y14.5M-1994 standard [1] as follows:

- "Flatness is the condition of a surface having all elements in one plane. A flatness tolerance specifies that all points of the surface must lie in some zone bounded by two parallel planes which are separated by the specified tolerance".
- "Circularity is a condition where: (a) for a feature other than a sphere, all points of the surface intersected by any plane perpendicular to an axis are equidistant from that axis; (b) for a sphere, all points of the surface intersected by any plane passing

through a common center are equidistant from that center. A circularity tolerance specifies that all points of each circular element of the surface must lie in some zone bounded by two concentric circles whose radii differ by the specified tolerance".

- "Cylindricity is a condition of a surface of revolution in which all points of the surface are equidistant from a common axis. A cylindricity tolerance specifies that all points of the surface must lie in some zone bounded by two coaxial cylinders whose radii differ by the specified tolerance. In the case of cylindricity, unlike that of circularity, the tolerance applies simultaneously to both circular and longitudinal elements of the surface (the entire surface)".

1.1 Straightness and its Evaluation

Straightness is defined as "a condition where an element of a surface, or an axis, is a straight line. A straightness tolerance for the line elements of a feature specifies that each line element must lie in a zone bounded by two parallel lines which are separated by the specified tolerance and which are in the cutting plane defining the line element" [1].

In order to verify if the feature is in concordance with the designer's specifications, some measurements should be taken. Coordinate measuring machines (CMM) are flexible, accurate devices for taking measurements and determining the acceptability of manufactured parts. They can be used for dimensional measurement or inspection, profile or form measurement, angularity or orientation, depth mapping, digitizing or imaging and shaft measurement.

Coordinate measuring machines consist of four main components: the machine itself, the measuring probe, the control or computing system, and the measuring software. The machines are available in a wide range of sizes and designs with a variety of different probe technologies. They can be mounted on a benchtop or desk, can be freestanding, handheld or portable. They can be controlled and operated manually, or by CNC (computer numerical control) or PC (personal computer). The measuring probes detect the point of beginning material viewing from a chosen direction, usually perpendicular. They can be mechanical, optical or laser probes. The most common type is the touch probe which touches the surface of the work piece and a signal containing the coordinates of that point is sent to the CMM. Other probes are laser triangulation probes which scan the surface and transmit a continuous flow of data to the measurement system. Video cameras and still cameras are other probe heads. A multi-sensor coordinate measuring machine has capabilities to mount more than one sensor, camera, or probe at a time. Their features are crash protection, offline programming, reverse engineering, shop floor suitable, statistical analysis and temperature compensation.

There are machines which provide continuous path tracing capabilities, but most of them rely on point sampling. The form deviations are calculated from the sampled coordinate points, so fitting algorithms must be employed to describe the feature. There is no information about the surface lying outside the sampled points. These sampled points represent the whole feature and therefore accurate fitting algorithms are required. The purpose of the algorithm, in the case of form errors, is to find the minimum zone which encloses all the measured points.

1.2 Least Squares and Minimum Zone Fitting

The most common of the existing fitting techniques for finding the form errors, are least squares and minimum zone. The least squares method fits an ideal form to coordinate data by minimizing the sum of squared deviations. In the case of determining straightness the ideal form is represented by a straight line. This method tries to get every point as equally close to the line as it can get it. The way to do this is to make sure that no point is far from the average distance away from the line. This line is called the best fit line, regression line, or least squares line. Let the equation of least squares line be:

$$y = ax + c \tag{1.1}$$

where a is the slope and c is the y-intercept of the least squares line. The summation of all the linear deviations, e_i, is minimized in Equation (1.2), where (x_i, y_i) are the coordinates of the sample points.

$$MinS = \sum e_i^2 = \sum (y_i - (ax_i + c))^2 \tag{1.2}$$

The least squares parameters are determined [2] to be:

$$a = \left(N\sum x_i y_i - \sum x_i \sum y_i\right) / \left(N\sum x_i^2 - \left(\sum x_i\right)^2\right) \text{ and} \tag{1.3}$$

$$c = \left(\sum y_i - a\sum x_i\right)/N \tag{1.4}$$

where N is the number of sample points. Having the parameters of the least squares line, the orthogonal distances (d_i) of all the points from it can be calculated with Equation (1.5).

$$d_i = \left(y_i - \left(ax_i + c\right)\right) / \sqrt{\left(1 + a^2\right)} \tag{1.5}$$

The least squares straightness tolerance can be expressed as:

$$Z = \max\left(d_i\right) - \min\left(d_i\right) \tag{1.6}$$

Figure 1.1 shows an example of some sample points and their corresponding least squares line. The dashed arrows point the furthest point from the least squares line in each side. These two points have the maximum deviation from the fitted line and the sum of their absolute deviations is the tolerance zone.

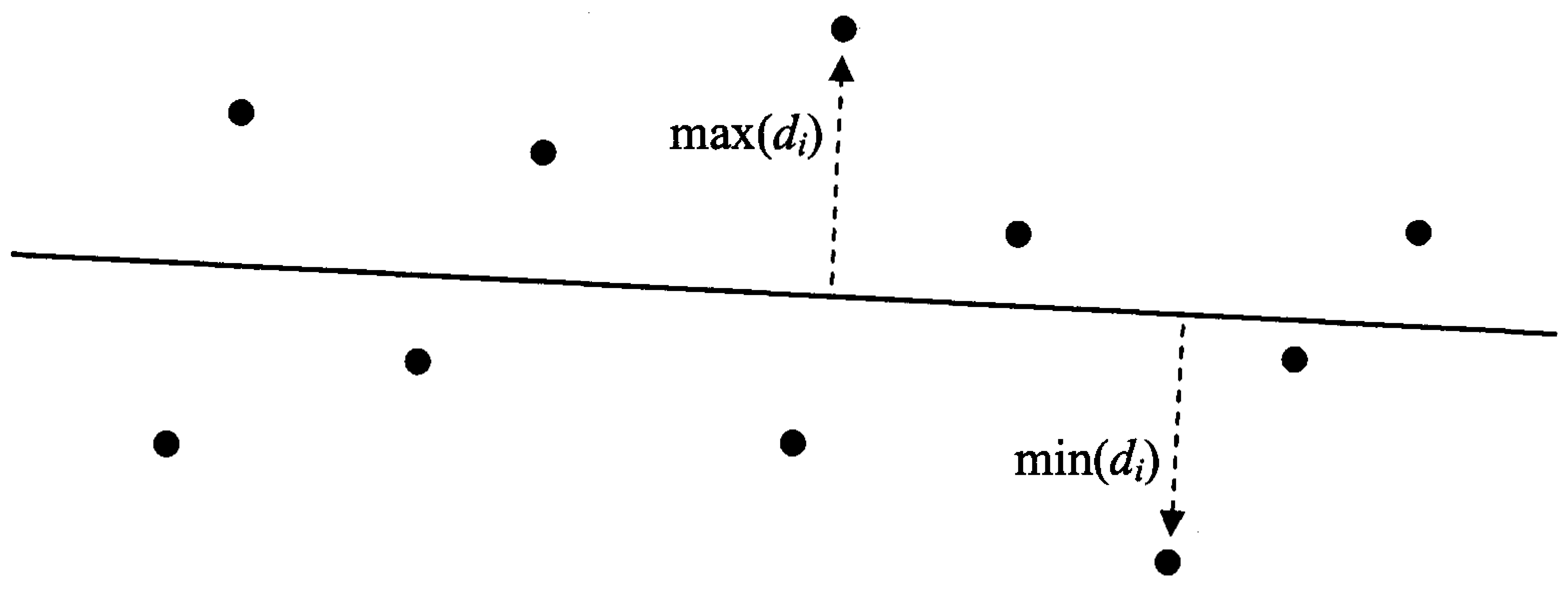


Figure 1.1 Least squares straightness tolerance.

The least squares algorithm is very easy to implement and computationally fast but it cannot guarantee the minimum zone. The least squares line is only an approximation of the points, not an exact solution. Also, each point influences the position and orientation of the least squares line and this makes the line to be dependent on each point, even on the outliers which are the points further from the rest of the data points. More formally, an outlier is an observation that lies outside the overall pattern of a distribution. Moreover, group of points concentrated in a region, bring the line close to them and hinders the proper representation of all the points by the line.

The minimum zone method is based on the envelope principle. The normal distance between the maximum inscribing and minimum circumscribing features that bound the entire feature of interest is the deviation range. Minimum zone in the straightness case is given by two parallel lines which enclose all the points at minimum distance apart. The minimum zone is given by three points, two contacting one line, and one, the other line [3, 4] as shown in Figure 1.2, where *MZ* is the minimum zone. One proof for it is given in [4] and it is described in the next paragraph.

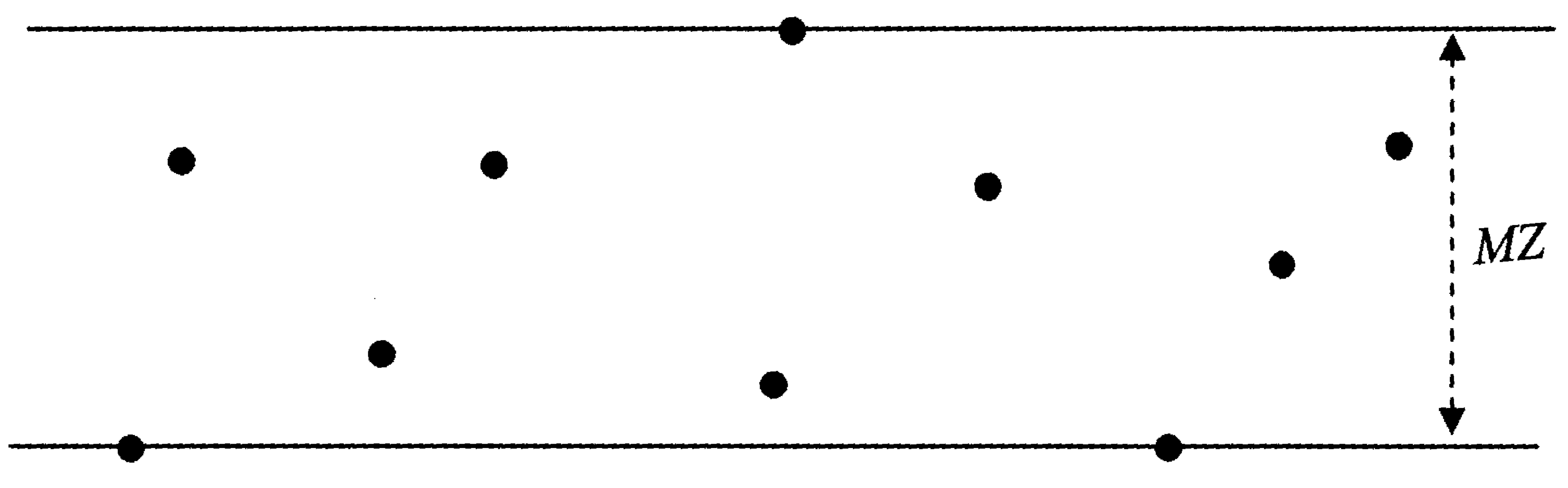


Figure 1.2 Minimum zone straightness tolerance.

Let us consider the two points *A* and *B* from Figure 1.3 and two parallel enclosing lines passing through these two points. These lines are represented in Figure 1.3 by the continuous lines. The line which connects the points *A* and *B* form with the parallel enclosing lines an angle α and an angle θ such that $\alpha \leq \theta$. There exists an angle β in such a way the lines can be rotated around points *A* and *B*, to decrease the angle α. The angle β corresponds to the first point touched by one of the enclosing lines, in this case point *C*. The parallel lines still enclose all the points, point *C* being the first touched point and the rotation angle (β) being the smallest angle of rotation. The distance between the initial enclosing lines is d_1 and the distance between them after the rotation is d_2.

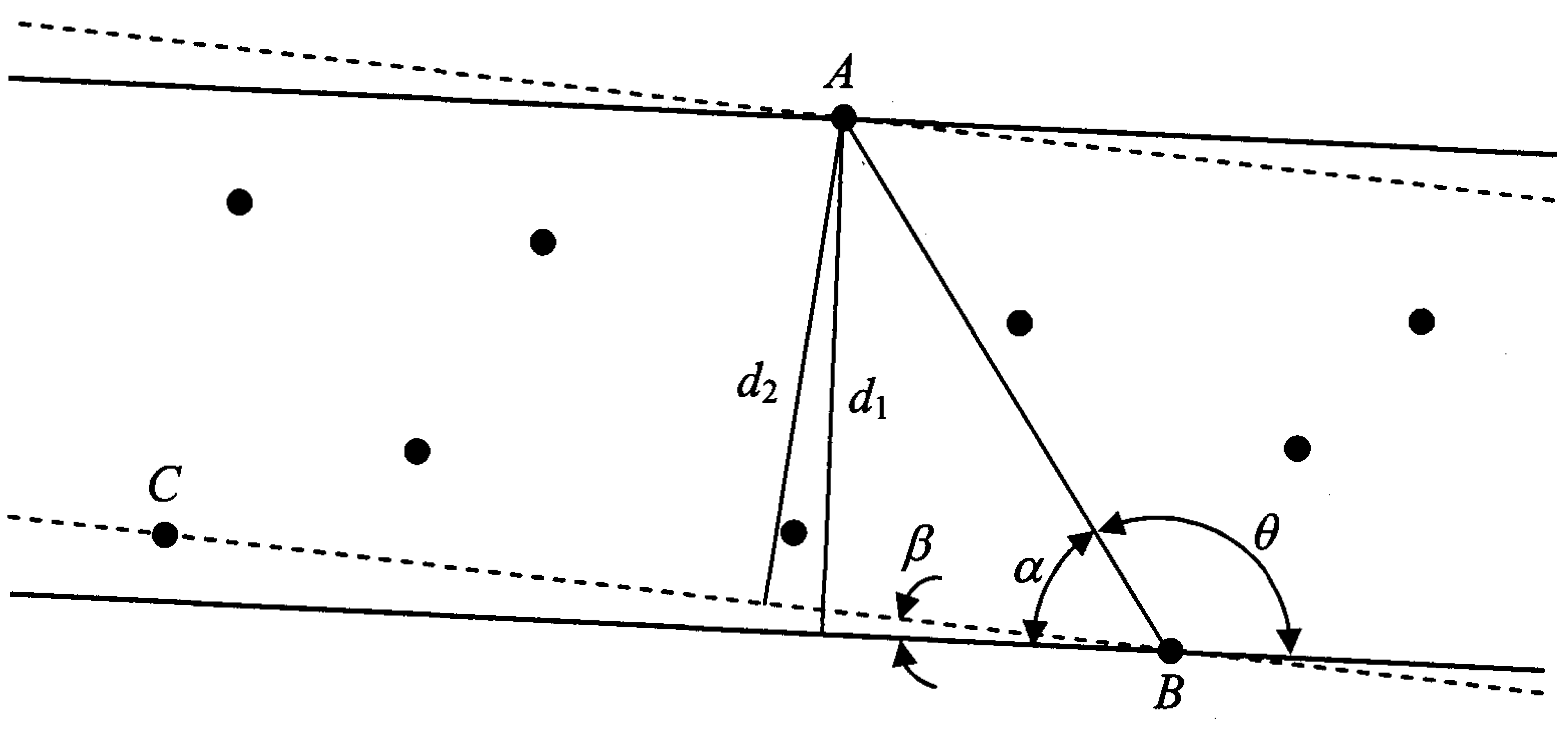


Figure 1.3 Minimum zone defined by three points.

$$AB = d_1/\sin(\alpha) \tag{1.7}$$

$$AB = d_2/\sin(\alpha - \beta) \tag{1.8}$$

$$d_1/\sin(\alpha) = d_2/\sin(\alpha - \beta) \tag{1.9}$$

$$d_2/d_1 = \sin(\alpha - \beta)/\sin(\alpha) < 1 \tag{1.10}$$

$$d_2 < d_1 \tag{1.11}$$

Equations (1.7-1.11) prove that the distance between the parallel enclosing lines after the rotation is smaller than before the rotation. Before the rotation, the enclosing lines touched two points and after the rotation, they touch three points and they give a smaller distance between them. Therefore the minimum zone of a set of points is defined by three points.

The minimum zone method is more accurate than the least squares method but the computational complexity is higher. The minimum zone fitting contains two different approaches. One is the optimization approach, where a function is defined and the parameters of that function found in such a way to minimize or maximize it, in order to find the solution. The algorithms based on this approach are very fast but they cannot guarantee the minimum zone because they cannot avoid local maxima or minima problems. The other approach is based on computational geometry; it takes in consideration the geometry of the points. The most common geometric shape used in this approach is the convex hull.

The next paragraphs introduce the convex polygon and the convex hull principle. A polygon is a closed planar path composed of a finite number of sequential line segments. The straight line segments that make up the polygon are called *sides* or *edges* and the points where the sides meet are called *vertices* of the polygon. A simple polygon is a polygon which does not intersect itself anywhere as shown in Figure 1.4; otherwise it is called complex polygon (Figure 1.5).

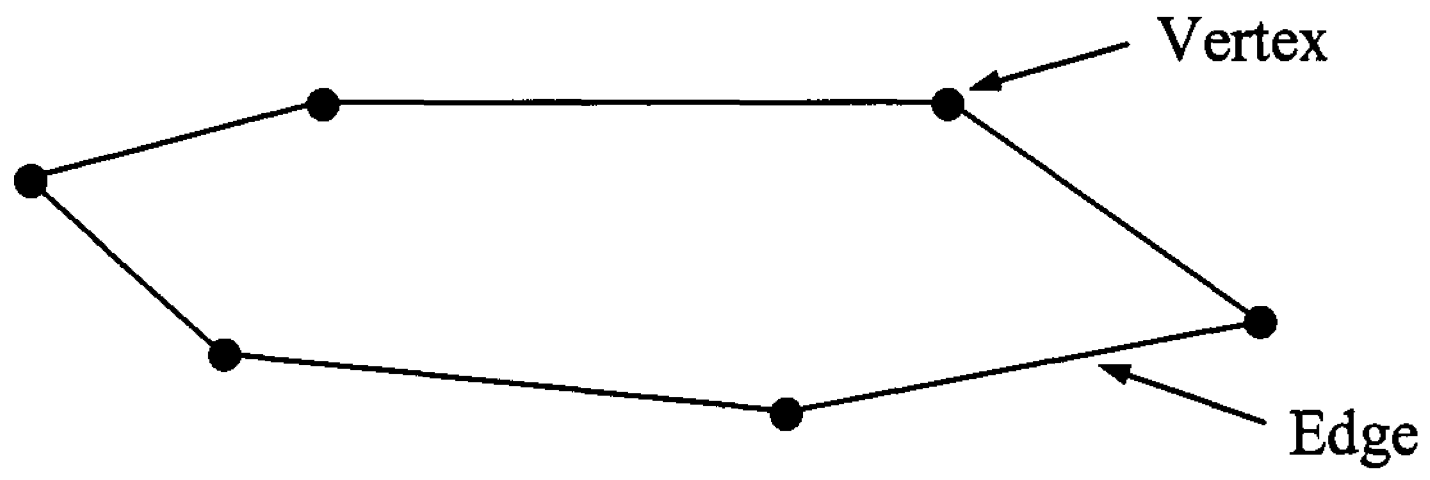


Figure 1.4 A simple polygon.

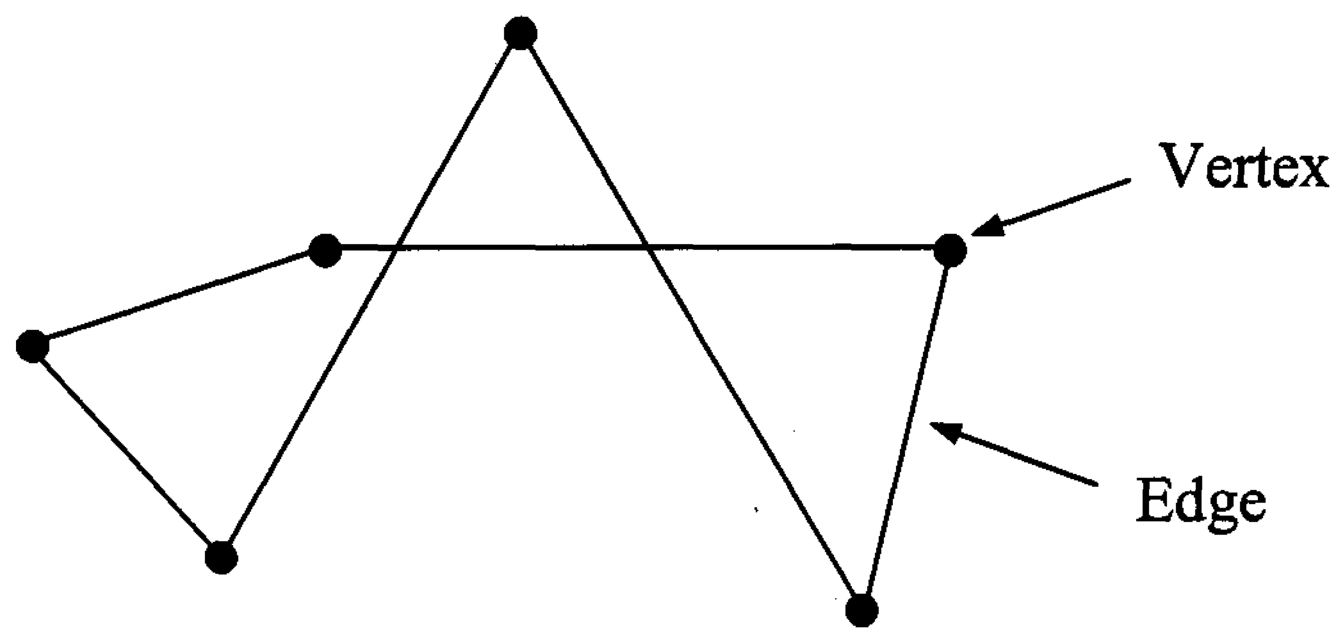


Figure 1.5 A complex polygon.

A simple polygon is convex (Figure 1.6) if, given any two points on its boundary or in its interior, all points on the line segment, drawn between them, are contained in the boundary or the interior of the polygon [5]. Another definition of a convex polygon can be given as a function of the internal angles. A polygon is convex, if every internal angle is at most 180 degrees. An internal angle (or interior angle) is an angle formed by two sides of a simple polygon that share an endpoint and is on the inner side of the polygon. An example of an internal angle is the angle α in Figure 1.6. A simple polygon has only one internal angle per vertex.

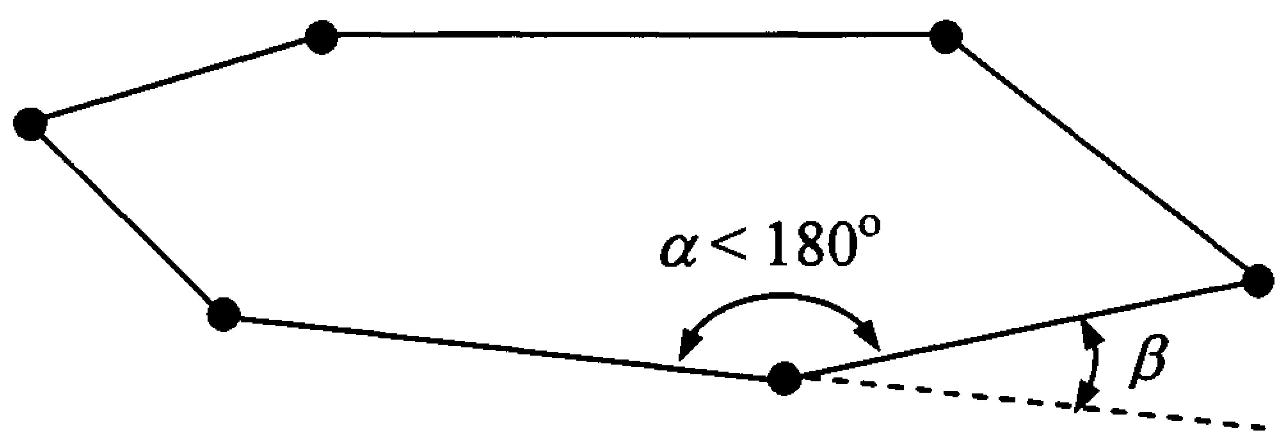


Figure 1.6 A convex polygon.

The polygon in Figure 1.7 is a non-convex polygon. There exists at least one line segment (*AB*), connecting two points, which is not all contained in the polygon. If a simple polygon is not convex it is called concave. From the definition based on the angles, it can be seen that the polygon from Figure 1.7 has one internal angle (α) greater than 180 degrees which makes the polygon to be concave.

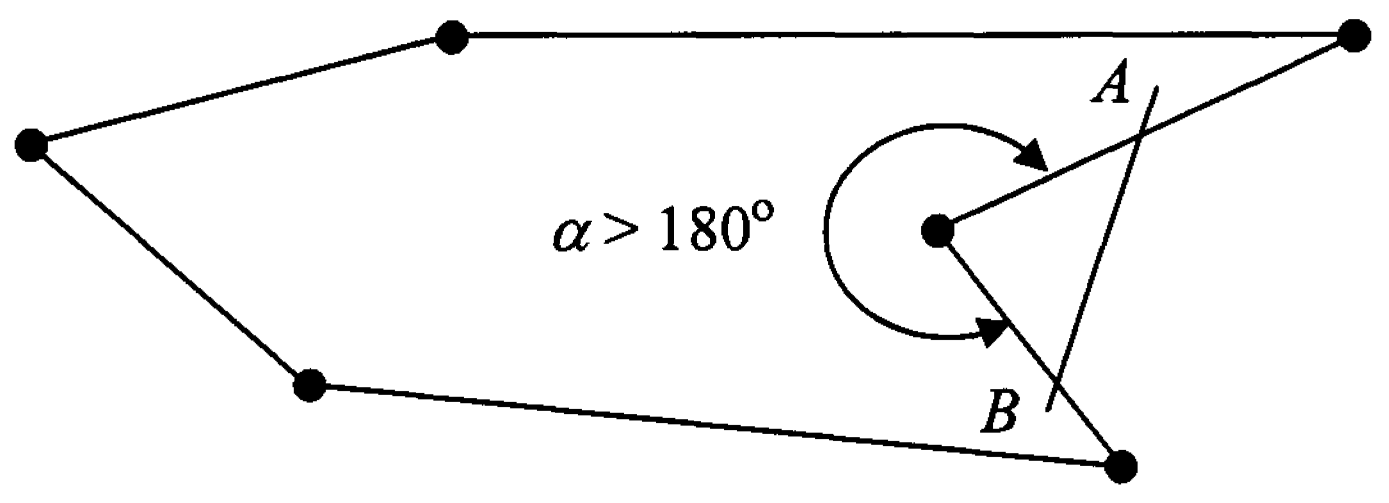


Figure 1.7 A concave polygon.

The convex hull of a set of points is the smallest convex set that contains the points. For two dimensional sets of points, the convex hull is a convex polygon as shown in Figure 1.8. So, the convex hull is the smallest convex polygon which encloses all the points.

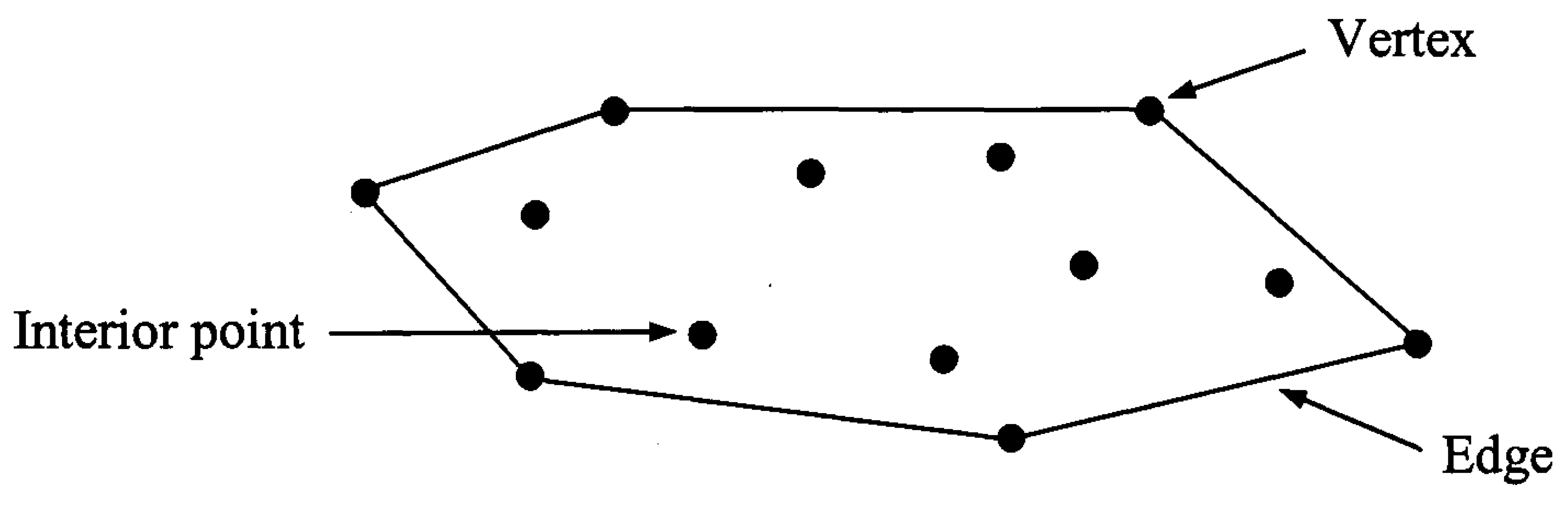


Figure 1.8 Convex hull in two dimensions.

The minimum zone of a set of points is the minimum zone of the convex hull of that set [4]. The parallel enclosing lines pass through the extreme points of a set, which are also vertices of the convex hull. The parallel lines cannot pass through interior points because they will not be enclosing lines anymore. Being out of the interest for this purpose, the interior points of the convex hull can be deleted. The minimum zone is parallel to one of the edges of the convex hull, and more than this, one of the minimum zone lines coincides with that edge [4]. In Figure 1.9, the edge P_0P_1 of the convex hull coincides with one of the parallel enclosing lines of the minimum zone and more specific, with the upper one.

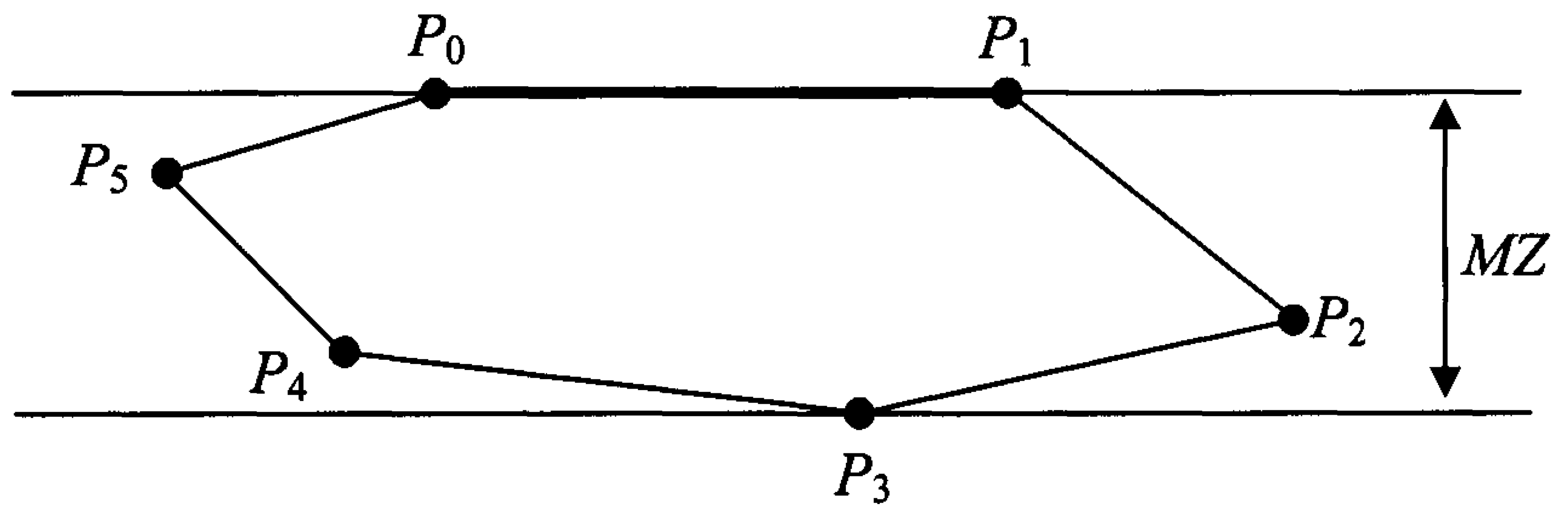


Figure 1.9 Minimum zone for the convex hull.

If the minimum zone lines do not coincide with an edge of the convex hull as in Figure 1.10, they represent only a feasible solution for the convex hull and they can be rotated in such a way around the points P_0 and P_3, to reduce the distance between them, until one of them meets another point, as shown in Figure 1.3. After the rotation, one line touches two points and the other line touches one point. The points which define the minimum zone are extreme points; they are located at the boundary of the set. From definition, the convex hull encloses all the points and is represented by the extreme points. The two points which are on the same minimum zone line also define an edge of the convex hull for points. Therefore, one of the minimum zone lines coincides with an edge of the convex hull for the points.

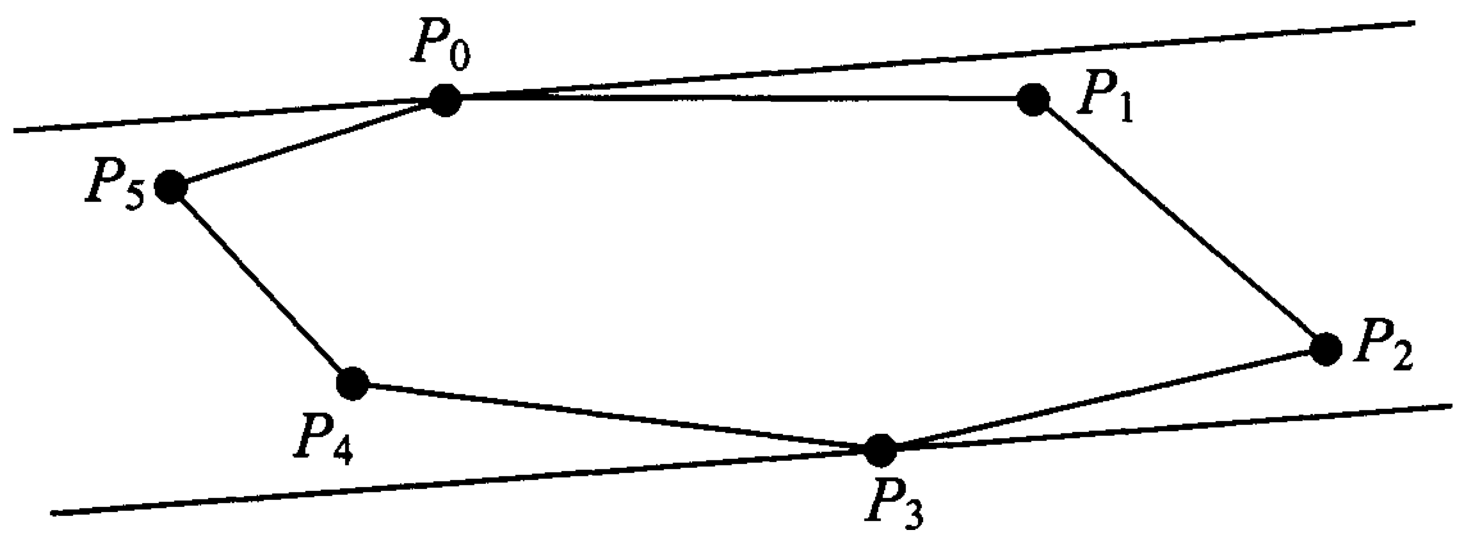


Figure 1.10 Feasible minimum zone solution for the convex hull.

The other minimum zone line passes the opposite vertex of the convex hull. The opposite vertex for an edge is the furthest vertex of the convex hull from that edge. Being the furthest point from that edge, the parallel lines enclose all the points. This satisfies one of the conditions from the straightness definition. Each edge of the convex hull and its opposite vertex represent a feasible solution. The pairs of points which represent feasible solutions are called antipodal. In other words, a pair of points that admit parallel enclosing lines is called antipodal [4, 6]. The second condition from the straightness definition is that the enclosing lines should be at minimum distance apart. Therefore, the feasible solution which gives the smallest distance between the parallel lines is the minimum zone for the convex hull and respectively, for the set of points.

1.3 Literature Review

The algorithms obtained from the literature can be classified in two main categories: optimization and computational geometric algorithms. The algorithms from the first category represent the straightness tolerance with a function and then find the parameters of that function in such a way to minimize the straightness error. These algorithms are very fast but may not yield exact solution because of mathematical approximation and convergence problems. On the other hand, the computational geometric techniques, which take in consideration the geometry of the sets of points, guarantee the accurate solution in expense of more computational time.

Many algorithms based on nonlinear optimization approach were developed in the past. Chen and Fan [7] used a generalized reduced gradient method embedded into Microsoft Excel spreadsheet to minimize the straightness error. Cheraghi et al. [8] formulated the straightness error evaluation as nonlinear optimization problem with linear objective function and nonlinear constraints. Wang [9] adopted the same strategy using a sequential quadratic programming method to determine the search direction and an augmented Langragian function for the search step. Carr and Ferreira [10] reduced the nonlinear optimization problem into a sequence of linear programs that converge to the solution of the nonlinear problem. Endrias and Feng [11] reduced and optimized the rigid-body coordinate transformation parameters to evaluate the form errors. Another linearization of nonlinear equations is done by Weber et al. [12] using Taylor expansion. Kanada and Suzuki [13] also linearized the objective function and applied some linear search techniques. Shunmugam [14, 15] introduced two new approaches, the median technique and an approach based on the minimum average deviation. These techniques are very fast computationally but they cannot guarantee the accuracy of the found solution.

Another way to solve nonlinear optimization problem was proposed by Sharma et al. [16]. A genetic algorithm has been used to solve a generalized minimax problem that is applied to straightness. This approach forms multiple search zones throughout the data set for the search variables to arrive at a global optimal solution. An improved genetic algorithm was proposed by Wen and Song [17]. The new algorithm employs the generation alternation model based Minimal Generation Gap and blend crossover operators and is more efficient and robust. These methods are powerful searching techniques and overcome localized minima because they work on a group of points so that multimodal peaks are searched simultaneously.

Suen and Chang [18] applied a neural network interval regression method for straightness evaluation. The neural network is used to adjust the coefficients of the linear function of the interval model. The learning algorithm is the least mean squares learning algorithm. The network is adjusted using the error between the actual output and the target output until the actual output satisfies the constraints. If the decay rate of the penalty coefficient is too slow, then extra training time is used to converge to the accurate solution, resulting in a waste of processing time. If the coefficient decays too rapidly, the neural network is unable to get an accurate result [18].

More accurate methods are those based on the computational geometric approach. Accurate algorithms were described by Traband et al. [4] and Samuel et al. [6]. The algorithms are based on the convex hull principle. The minimum zone for a set of points is the minimum zone for the convex hull of that set of points and one of the parallel supporting lines coincides with an edge of the convex hull. Once the convex hull is determined, each edge is evaluated with respect to its furthest point from the hull. The edge and its opposite vertex of the convex hull that give the smallest distance between them, yields the value of the minimum zone straightness error. Computational complexity of an algorithm represents the number of steps it takes to solve a problem and it shows how fast the computation time grows with the problem size. One of the notations of the computational complexity is O-notation, which has been discussed in details in Chapter 3. The computational complexity for finding the convex hull is $O(n \log n)$, where n is the number of points. All the edges of the convex hull have to be evaluated and for each edge all the vertices should be tested in order to find the furthest one. In the worst case, all the points lie on the convex hull and therefore the computational complexity is $O(n^2)$. A similar but faster algorithm takes in consideration only the antipodal pairs of points. Using the antipodal pairs [4, 6] reduces the complexity of the algorithm to $O(n \log n)$. The complexity for finding the convex hull dominates in this algorithm because, for determining the antipodal pairs the complexity is only $O(n)$, and for computing the minimum distance with respect to the antipodal pairs is constant. Therefore, the overall complexity of the algorithm is $O(n \log n)$.

Huang et al. [19] introduced the control line rotation scheme. This algorithm is based on the criteria for the minimum zone solution and data exchange rules. The least squares line is used to find the first two control points. Using the control points sequence rule, each control line rotate with respect to its control point in the half-field to find the third control point. The solution is found when the three control points satisfy the sequence rule. This algorithm is fast, it can stop in few iterations but the minimum zone is not guaranteed because more points can satisfy the sequence rule and they can represent only a feasible solution, not the minimum one. The least squares line cannot be considered all the time a good initial solution. By rotating the control lines with the smallest angle in order to find the third point, the algorithm can find only a local solution with the control points respecting the sequence rule.

It can be seen that every method has advantages and disadvantages. If a method can guarantee the accuracy of the solution it is not fast enough, and if it is fast, it cannot guarantee the exact solution. Therefore, the main concern in the straightness and the other form errors evaluation is to find an algorithm which is accurate as well as fast.

1.4 Thesis Overview

The goal of this thesis is to develop an accurate and efficient algorithm for minimum zone straightness evaluation. This method guarantees the minimum zone and has a less computational complexity for the expected case than the convex hull based methods described above. The present chapter describes the straightness error and the most common fitting techniques for its evaluation including the convex hull concept. The literature related to the present research work is also included in this chapter.

Chapter 2 introduces the relationship between minimum zone for a subset of points and the minimum zone for the entire set of points. It continues with the description of the proposed method and ends with the converged solution. An initial solution is found and a test is performed to check if it is the minimum zone. The algorithm stops if the initial solution for the set is the minimum zone for the three points which define it. This condition ensures the initial solution is the minimum zone. If the initial solution does not satisfy the condition, the polygon expansion starts. The expansion ends when a subset of points (polygon) is found for which the initial solution is the minimum zone or when the minimum zone for a subset (polygon) is a feasible solution for the entire set. Therefore, the minimum zone for the set can be the initial solution or a different one. The polygon is enlarged only as much as it is required to find the minimum zone solution.

Chapter 3 presents the computational complexity of the algorithm including the expected case. The expected computational complexity depends on the size of the constructed polygon. The expected size of the convex hull for a set of points uniformly distributed in a convex polygon is $O(\log n)$ [5, 20]. The constructed polygon in the present method is smaller than the convex hull, resulting in a expected size less than $O(\log n)$ for the same distribution of points. The expected complexity of the algorithm is less than $O(n \log n)$ for points uniformly distributed in a convex polygon. For points uniformly distributed in a circle, the expected size of the convex hull is $O(n^{1/3})$ [5, 20]. The expected size of the constructed polygon for the same distribution of points is less than the expected size of the convex hull. Thus, the expected complexity of the present algorithm for points uniformly distributed in a circle is less than $O(n^{4/3})$.

In Chapter 4, existing data and simulation data are used to determine the accuracy and validate the efficiency of the proposed method. Chapter 5 concludes the thesis.

CHAPTER 2

PROPOSED METHOD

2.1 Theoretical Background

The minimum zone for a set of points is defined by three points [3, 4] as discussed previously. Three points must contact the parallel enclosing lines; two points must contact one line and one point the other line. Thus, the smallest subset of points for which the minimum zone can be found contains three points. Minimum zone for the three points is defined by two parallel lines at minimum distance apart, which enclose all the points. As discussed in Chapter 1, one of the minimum zone lines coincides with an edge of the convex polygon defined by the points. Each edge of the polygon together with its opposite vertex of the polygon defines a possible solution. In this case, the polygon is a triangle and each of the three edges of the triangle and its corresponding opposite vertex define a feasible solution as shown in Figure 2.1. The figure shows the three possible pairs of parallel enclosing lines for the three points. From each pair of lines, one line coincides with an edge of the triangle and the other one passes through its opposite vertex. These pairs of lines are represented in the figure by continuous and two types of dashed lines and the triangle by the dot lines. The smallest feasible solution is the minimum zone. From these three feasible solutions or zones (Z_1, Z_2, Z_3), the parallel lines which have the smallest distance between them, is the minimum zone. In the Figure 2.1, the case (a) having the smallest distance between the parallel lines ($Z_1 < Z_2$ and $Z_1 < Z_3$) is the minimum zone for these three points ($MZ_3 = Z_1$).

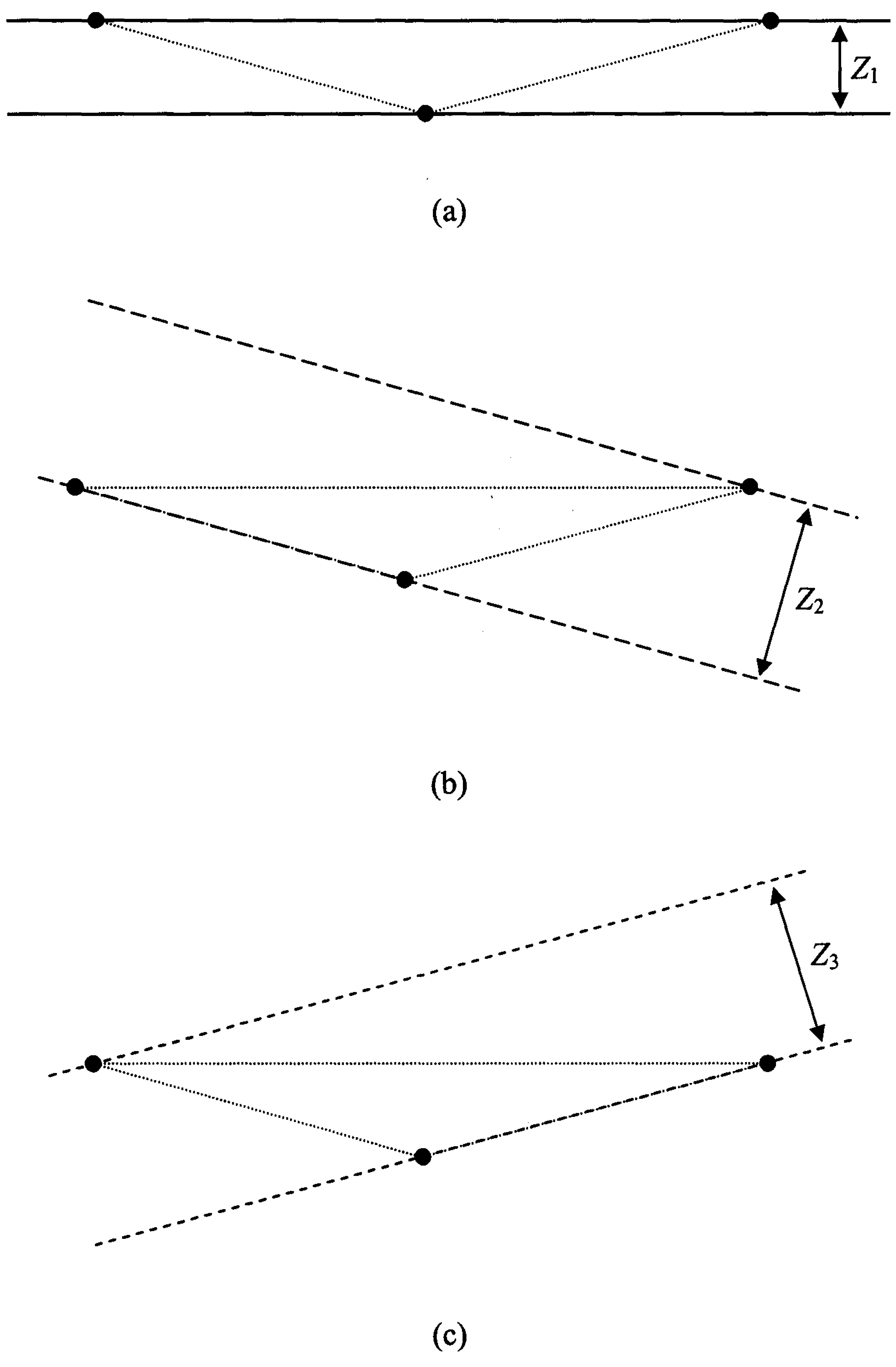


Figure 2.1 The three feasible minimum zone solutions for triangle: (a) Z_1; (b) Z_2; (c) Z_3.

Three points have the smallest value for their minimum zone than any other number of points. Let us consider that three points are given and their minimum zone is found as shown in Figure 2.1. Then other points are added to these three points and the minimum zone for all the points is recalculated. If the new points are added between the minimum zone lines for the initial three points, the minimum zone for all the points is the same, because the lines enclose the new points. An example is illustrated in Figure 2.2 where the new added points are between the parallel enclosing lines and they are represented by white colour. In this case, the minimum zone for the three points is a feasible solution for the entire set of points and more than this, because that is the minimum zone for the smallest subset, it is also the minimum zone for the complete set of points ($MZ_3 = MZ_s$). MZ_3 and MZ_s denote the minimum zones for three points and for the entire set of points respectively.

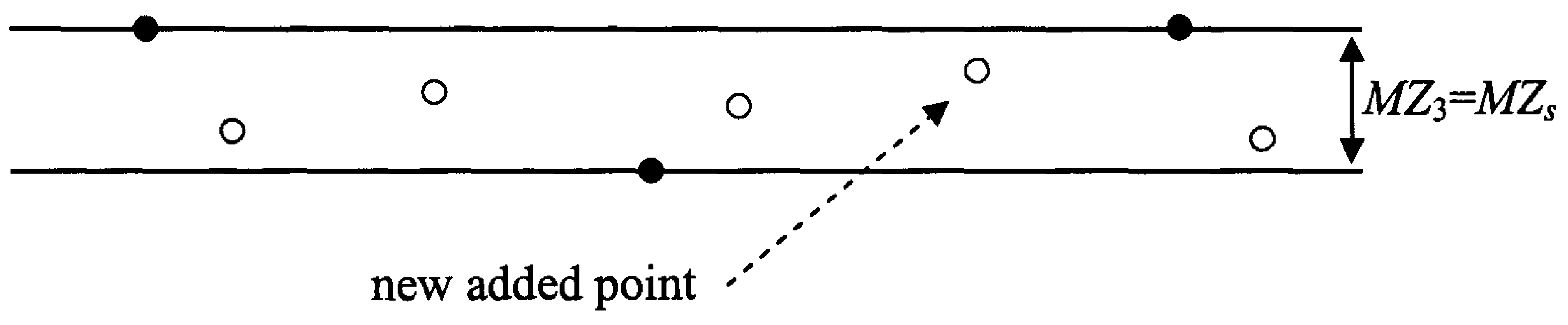


Figure 2.2 The minimum zone for three points is minimum zone for the set.

If at least one of the new points is inserted outside the minimum zone lines for the starting three points as in Figure 2.3, the minimum zone for all the points will be different and it can be only bigger. The minimum zone for the starting three points is not a feasible solution in this case, for the entire set, because of the outside added points. Because it is not a feasible solution, does not include all points, it also cannot be the minimum zone for the set. In Figure 2.3 there are four points added outside the previous minimum zone lines shown in white colour.

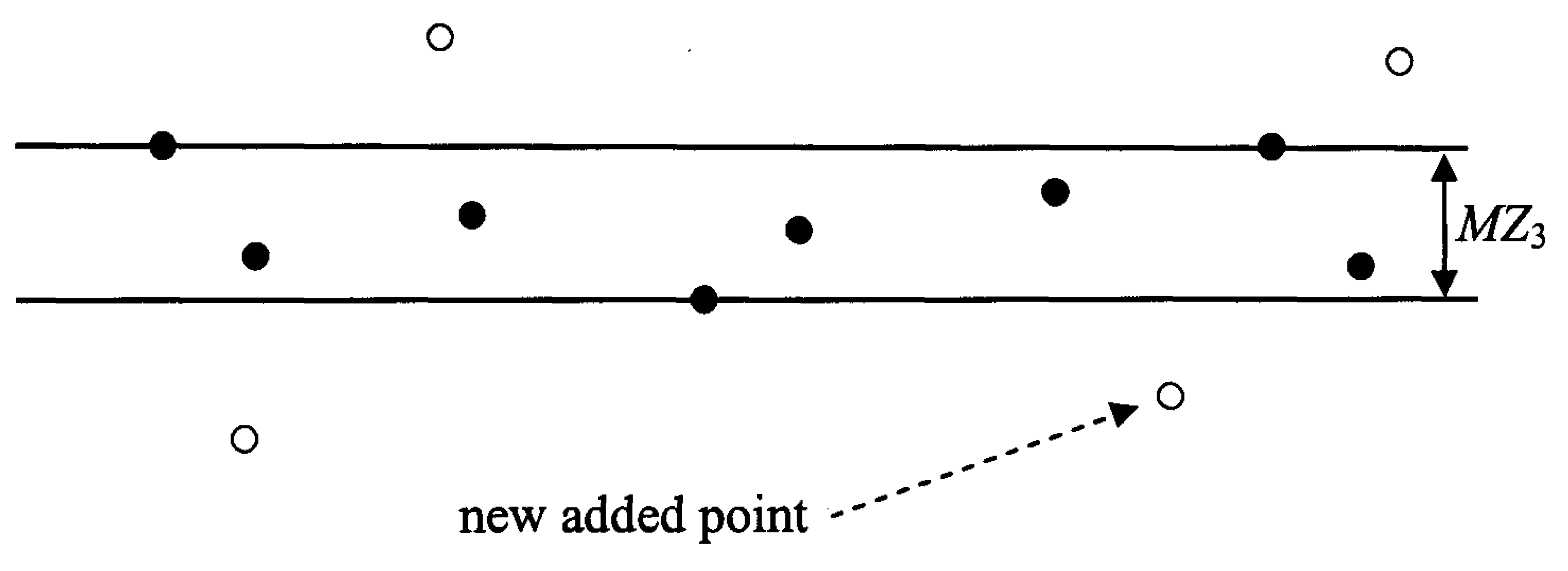


Figure 2.3 The minimum zone for three points is not minimum zone for the set.

The minimum zone for all the points is found and it is greater than the one for the starting three points. In Figure 2.4 the dashed lines represent the minimum zone for the initial three points and the continuous lines define the minimum zone for the entire set which is larger than the previous one ($MZ_s > MZ_3$).

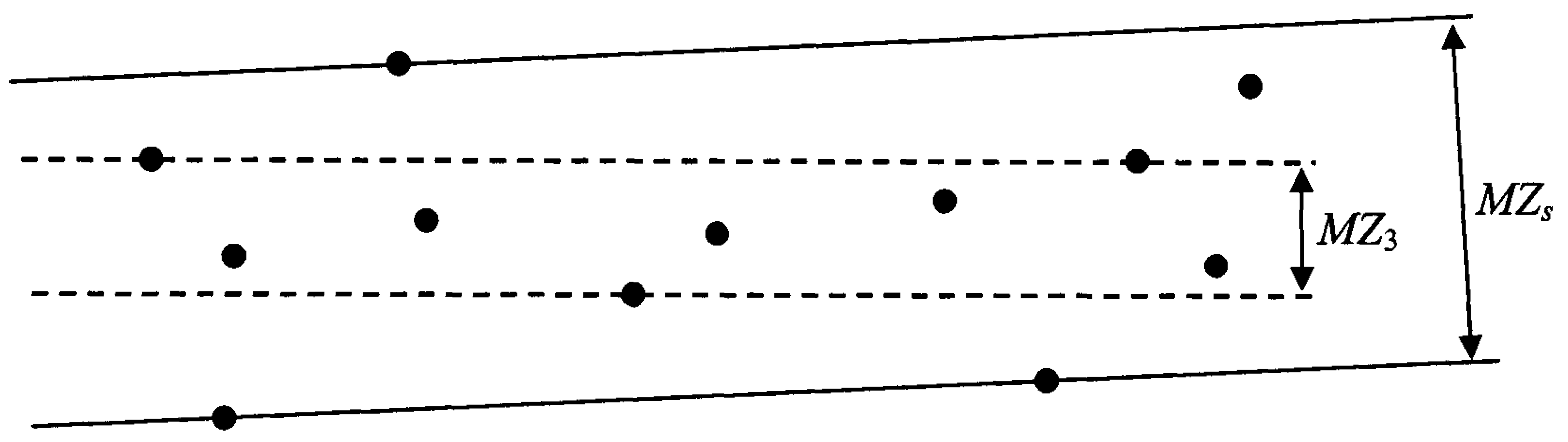
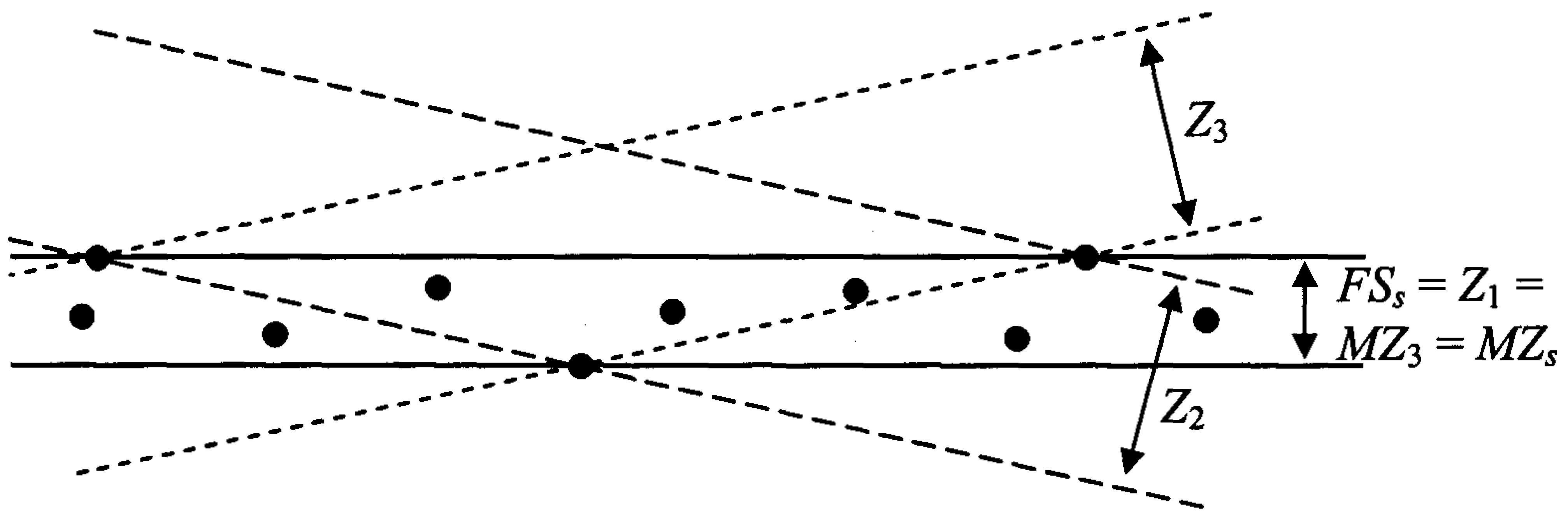


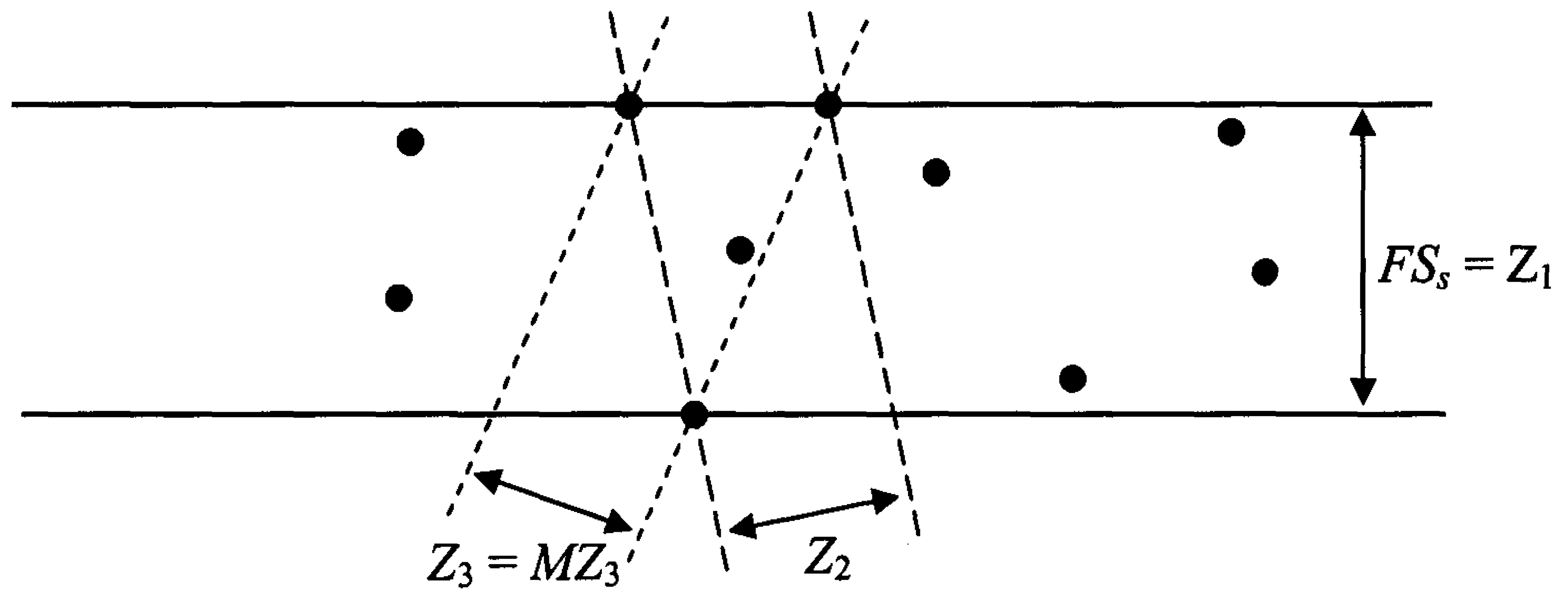
Figure 2.4 The minimum zones for three points and for the set.

For more points it is impossible to find a smaller minimum zone than the one for only three points. Adding more points can only keep the same minimum zone or enlarge it, it cannot reduce it. If the minimum zone for a subset and especially for the smallest subset is a feasible solution for the entire set, that is also the minimum zone for the set.

The reverse of this principle is that a feasible solution for a set of points is the minimum zone for the set if it is also the minimum zone for the smallest subset of points found starting with that feasible solution. Having a feasible solution is easy to check if it is the minimum zone for the three points which define it. This feasible solution for the set is also a feasible solution for the three points ($FS_s = Z_1$). It is compared with the other two feasible solutions of the three points (Z_2, Z_3) in order to find if it is their minimum zone or not. In Figure 2.5(a), the continuous parallel lines represent a feasible solution for the set of points (FS_s). It can be observed that the distance between these continuous lines is smaller than the distances between the dashed parallel lines which are the other two feasible solutions for the three points ($Z_1 < Z_2$ and $Z_1 < Z_3$). This implies the continuous lines represent the minimum zone for the three points ($Z_1 = MZ_3$) and because of that, it is also the minimum zone for the set ($FS_s = MZ_3 = MZ_s$). In Figure 2.5(b), the feasible solution for the set ($FS_s = Z_1$), given by the continuous lines, is not the minimum zone for the three points ($Z_1 \neq MZ_3$). The distances between the dashed lines are smaller than the one given by the continuous lines ($Z_2 < Z_1$ and $Z_3 < Z_1$). So, the continuous lines do not define the minimum zone for the three points. They represent only a feasible solution. The pair of dashed parallel lines which gives the smallest distance represents the minimum zone for the three points ($Z_3 = MZ_3$).



(a)



(b)

Figure 2.5 Two feasible solutions for a set of points: (a) $FS_s = MZ_3$; (b) $FS_s \neq MZ_3$.

If the feasible solution for the set is not the minimum zone for the three points ($FS_s \neq MZ_3$) does not imply that it cannot be the minimum zone for the set. It can be, if it is the minimum zone for a larger subset than the previous three points. The triangle or the smallest subset of points can be enlarged by adding new points from the set to these three points. The new points connect the end points of the *visible edge* [20, 21] of the polygon, the *visible edge* being deleted and the polygon expanded. The feasible solution for the set can be the minimum zone for the enlarged polygon or the minimum zone for the polygon can be a feasible solution for the set and in both cases that is the minimum zone for the set.

One way to determine the *visible edge* or what edge a point *can see* [20] is to store the vertices of the polygon in a clockwise or anti-clockwise order. Then it can be determined if a point is on the left or right side of an edge of the polygon. Let us consider a fixed line P_0P_1 defined by the points $P_0(x_0,y_0)$ and $P_1(x_1,y_1)$ is given. A third point $P_2(x_2,y_2)$ is added and its position with respect to the line P_0P_1 is unknown. The new point forms with the other two points a triangle. If the vertices of the triangle are oriented anti-clockwise as shown in Figure 2.6, the sign of the area of the triangle $P_0P_1P_2$ is positive.

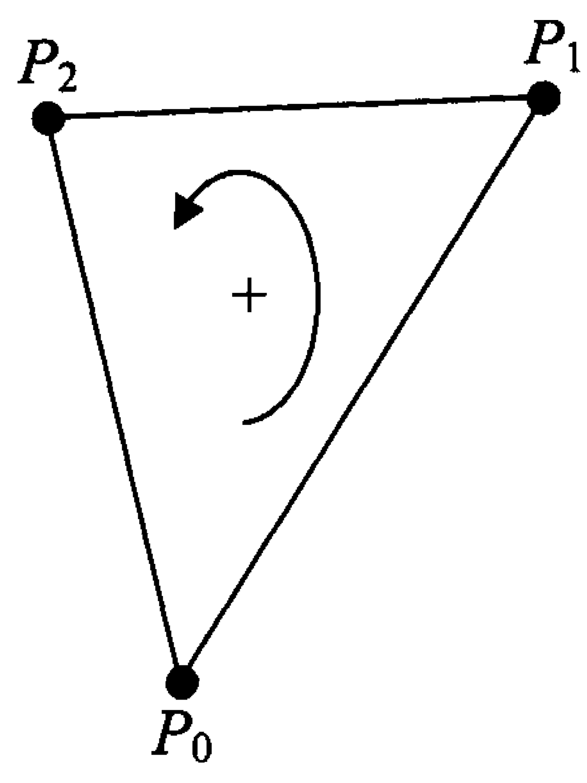


Figure 2.6 Anti-clockwise orientation of vertices around the triangle.

The Equation (2.1) represents the coordinates of the three points P_0, P_1, P_2. These coordinates are substituted in the Equation (2.2) which provides the area of the triangle formed by three points. If the sign of the area is positive, the point P_2 is to the left (positive) of the line P_0P_1. The sign of the area is negative if the triangle is oriented clockwise and P_2 is to the right (negative) of the line P_0P_1.

$$P_0 = (x_0, y_0), \quad P_1 = (x_1, y_1), \quad P_2 = (x_2, y_2) \tag{2.1}$$

$$A(\Delta) = \frac{1}{2}\begin{vmatrix} x_0 & y_0 & 1 \\ x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{vmatrix} = \frac{1}{2}[(x_1 - x_0)(y_2 - y_0) - (x_2 - x_0)(y_1 - y_0)] \tag{2.2}$$

If the polygon is kept in anti-clockwise order then the sign of the area will be positive for any three consecutive points on the polygon. For the polygon kept in clockwise order, the sign of the area for any three consecutive points on the polygon will be negative. If P_0P_1 is an edge of the polygon and P_2 is a new added point to the polygon, the point P_2 can see this edge if the area of the triangle $P_0P_1P_2$ has a different sign than the area of any three consecutive points from the polygon. In other words, a point can see an edge of the polygon if the orientation of the triangle defined by that point and the points which define that edge, is different than the orientation of the polygon.

2.2 Procedure

The proposed method is shown in Figure 2.7. It starts with a feasible solution, called *initial solution*, for the entire set of points obtained from least squares line and the smallest angle rotation principle [19]. Once the three points which define the initial solution are found, their minimum zone is calculated. The edge of the polygon which coincides with one of its minimum zone lines is called in this thesis, *minimum edge* and the edge which coincides with one of the initial solution line is called *reference edge* as shown in Figure 2.8, where these edges are symbolized by bold lines. If the minimum edge for the three points coincides with the reference edge, the solution is found. In other words if the feasible solution (*reference edge*) for the set coincides with the minimum zone for the three points (*minimum edge*), that is the minimum zone for the set. If they are not equal, the algorithm tries to expand the subset. The growing subset is called *polygon expansion* and is based on the furthest point principle. The added points are the *furthest points outward* or *inward from the minimum edge*. By finding the furthest point outward from the minimum edge, the algorithm tries to find the smallest polygon whose minimum zone coincides with the feasible solution for the set of points. Furthest point inward from the minimum edge is used to find if the minimum zone for the polygon is a feasible solution for the set of points. After the connection of the new added points with the existing polygon, the edges of the *update polygon* are evaluated and its minimum zone is found. The algorithm stops when the initial solution for the set is the minimum zone for a polygon, or when the minimum zone for a polygon is a feasible solution for the set of points. This satisfies the concept described earlier, which maintain that if a feasible solution for a set of points is the minimum zone for a subset of that set of points, or if the minimum zone for a subset of points is a feasible solution for the set, it is the minimum zone for the entire set of points.

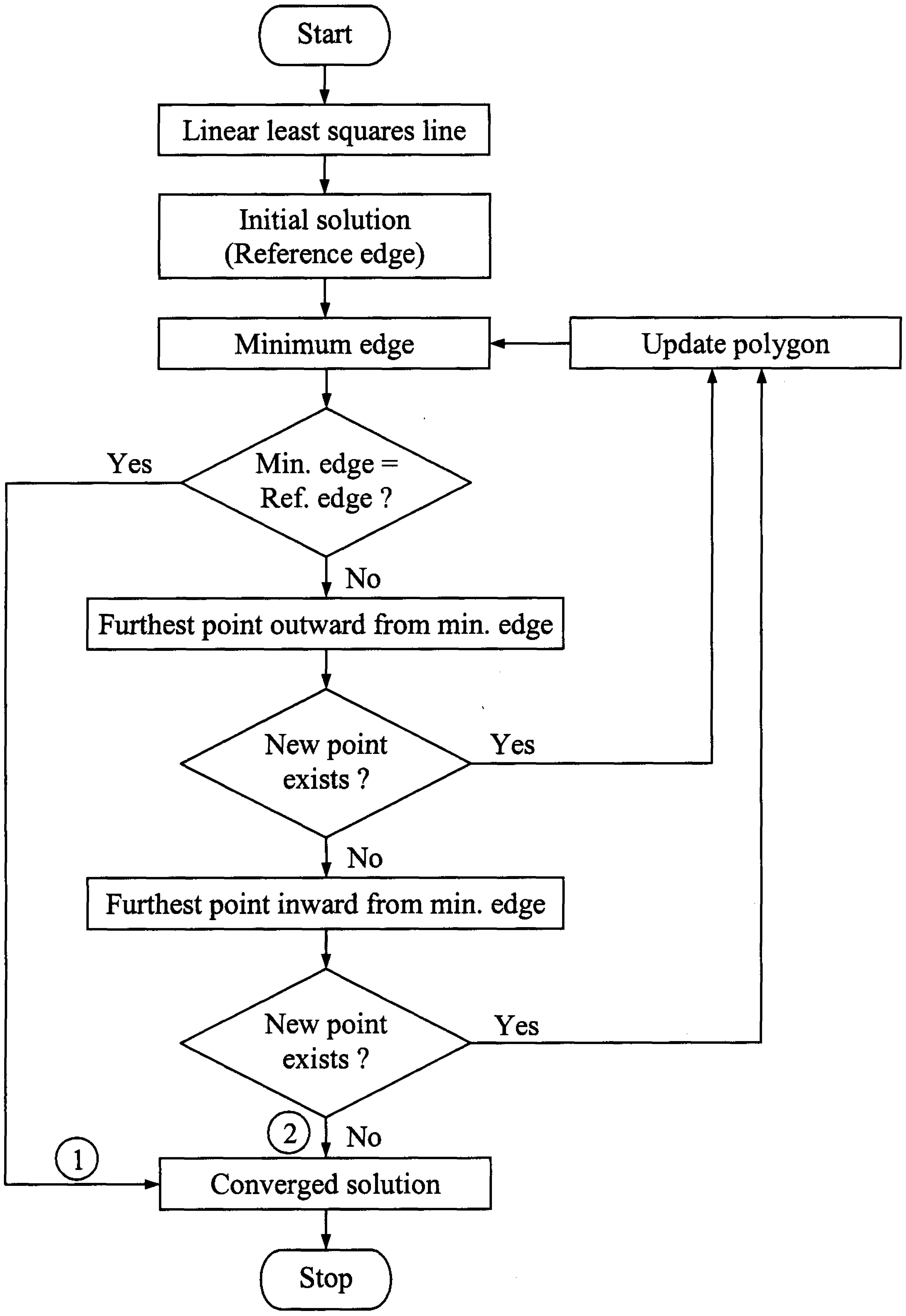

Start
Linear least squares line
Initial solution (Reference edge)
Minimum edge
Update polygon
Min. edge = Ref. edge ?
Yes
No
Furthest point outward from min. edge
New point exists ?
Yes
No
Furthest point inward from min. edge
New point exists ?
Yes
No
1
2
Converged solution
Stop

Figure 2.7 Straightness error evaluation procedure.

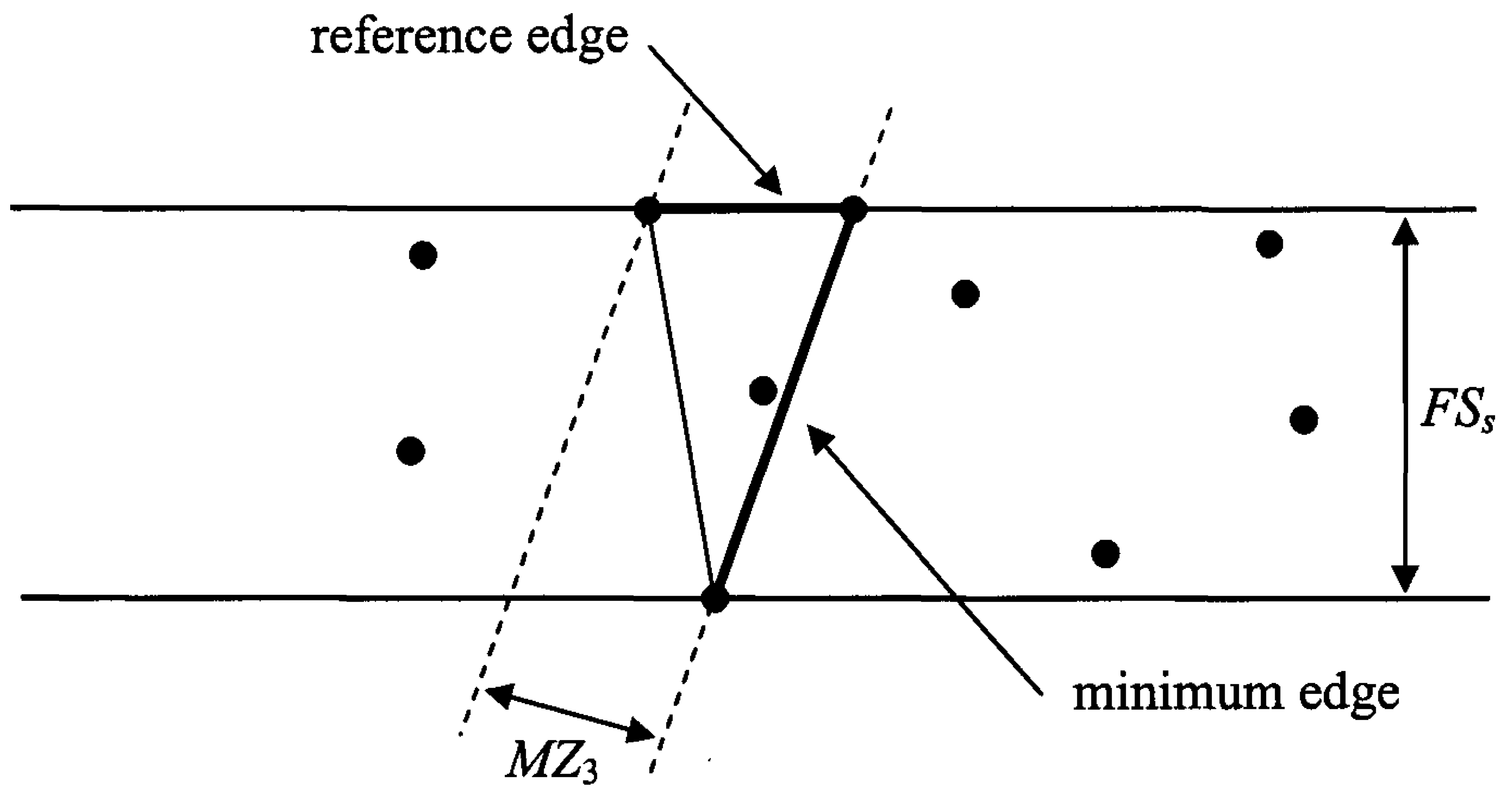


Figure 2.8 Minimum and reference edges.

An important observation about the constructed polygon is that, due to the starting triangle and the extreme added points, the polygon is, all the time, a convex polygon. Being extreme points (furthest from minimum edges), all the added points to the polygon are vertices of the convex hull for the set of points. Also the starting three points, because they define a feasible solution for the set, are vertices of the convex hull. Therefore, all the points in the subset are vertices of the convex hull. The polygon defined by these points is a part of the convex hull. Not all the edges of the polygon are edges of the convex hull but all the vertices of the polygon are vertices of the convex hull. It ensures that the constructed polygon is all the time a convex polygon. At each step, the subset grows with one point. The triangle defined by the initial three points expands to a quadrilateral, then a pentagon and so on, until the initial solution becomes minimum zone for the subset or another smaller feasible solution for the set is found, which will be the final solution.

2.3 Algorithm

2.3.1 Initial Solution

The least squares method does not provide exact values of straightness error but confers an acceptable initial solution. Substituting the coordinates of the points in Equations (1.3) and (1.4), the parameters of the least squares line are found. The linear least squares line is fitted through the points and the points with the maximum deviation in each side of the line are found with Equation (1.5) as shown in Figure 1.1. A parallel line with the least squares line is fitted through each found point as shown in Figure 2.9. Because these two parallel lines are fitted through the furthest point from least squares line in each side (P_0 and P_1), it ensures that they enclose all the points. Enclosing all the points, it denotes it is a feasible solution for that set of points.

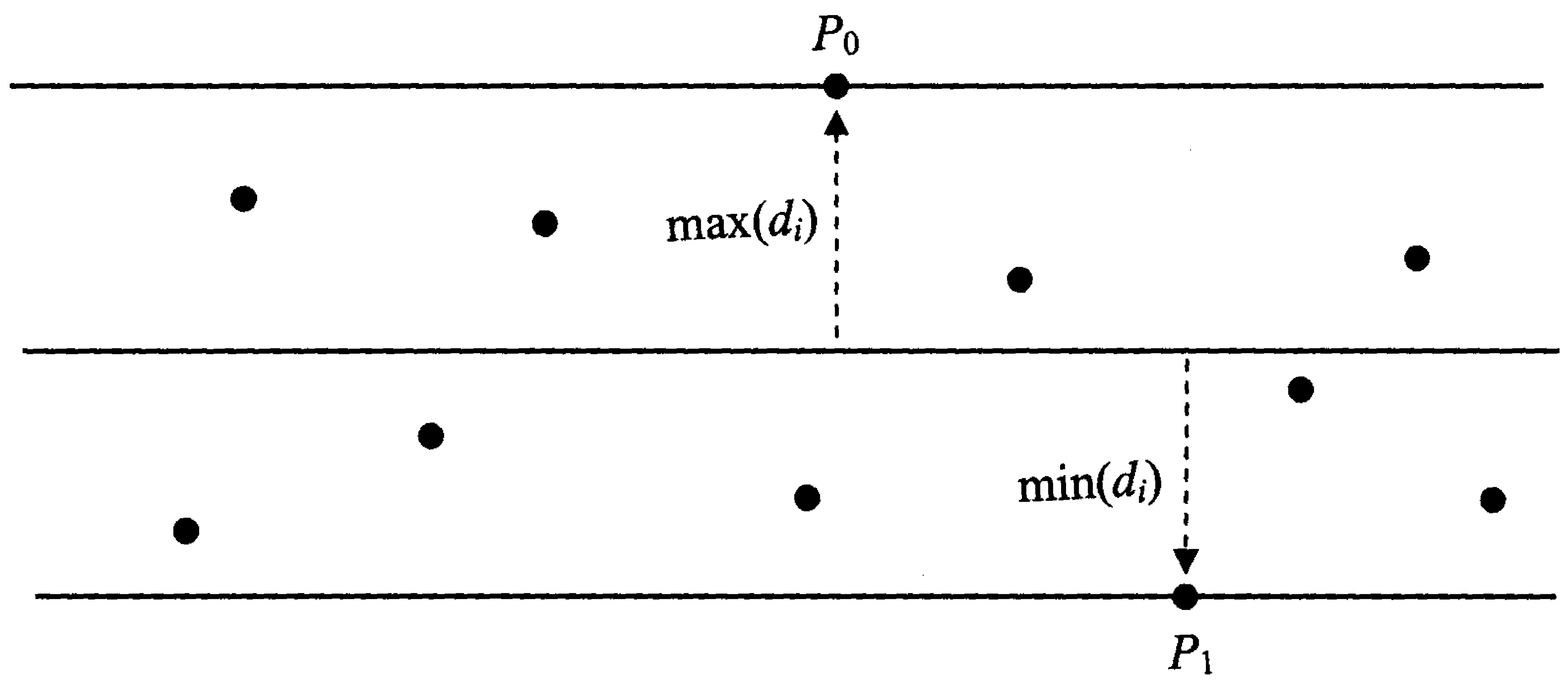


Figure 2.9 Parallel enclosing lines of a feasible solution.

In order to find the third point of the initial solution, the angles between each parallel enclosing line and the lines which connect each of the starting two points (P_0, P_1) with the other points from the set were calculated. The smallest angle and its corresponding point are selected. The selected point is the third point of the initial solution. This concept can be seen as a *rotation* of the parallel enclosing lines in the full-field around the starting two points (P_0, P_1) until one of the lines touches another point from the set as shown in Figure 2.10. Each point from the set corresponds to a rotation angle of the parallel enclosing lines. The first point touched by one of the parallel lines (P_2) has the smallest angle with respect to that line ($\alpha < \beta < \gamma < \theta$). The same concept was used by Huang et al. [19], but the rotation process was performed only in the half-field.

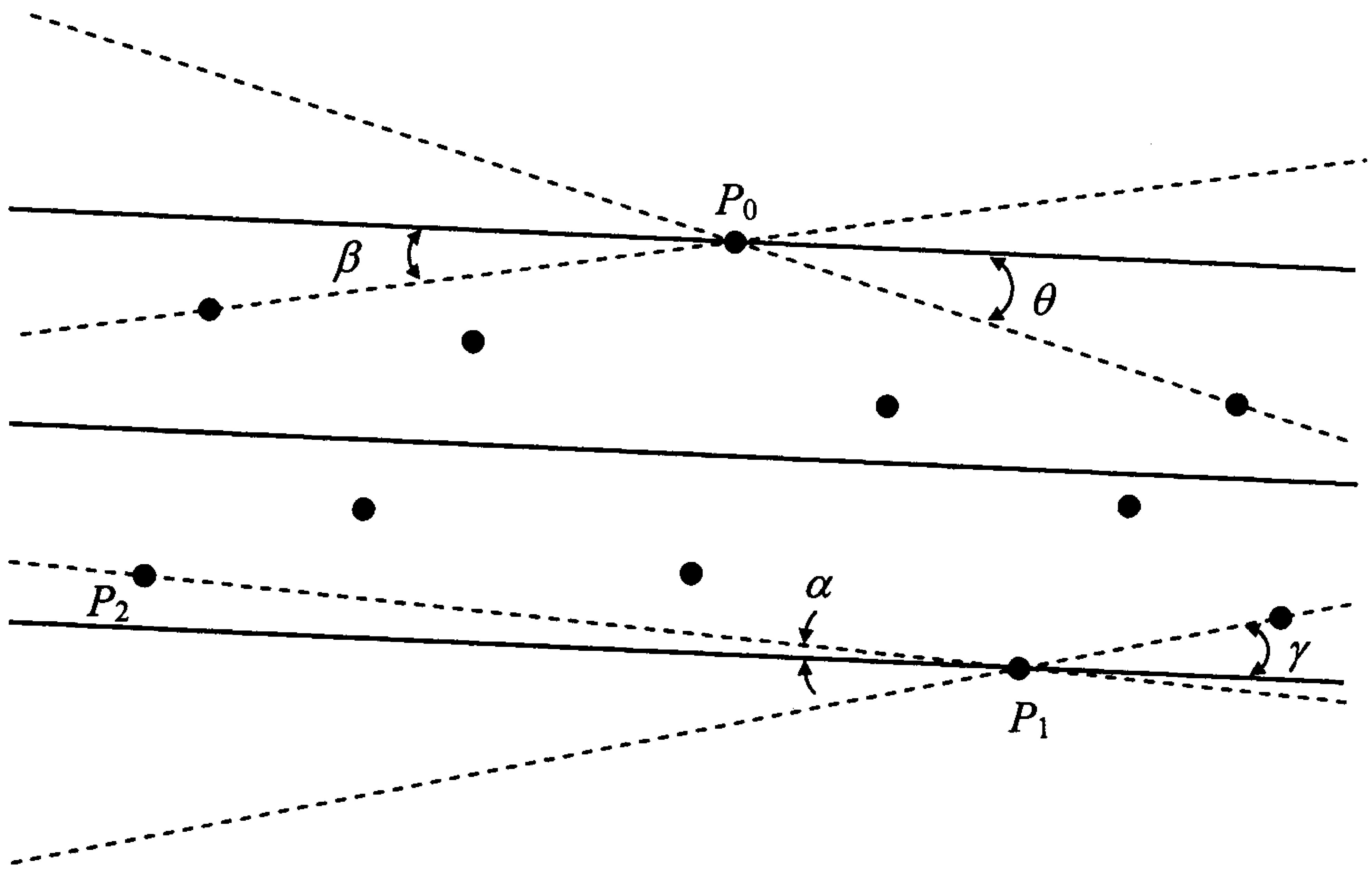


Figure 2.10 The smallest angle rotation of the parallel enclosing lines.

The smallest rotation angle guarantees that all the points are between the lines and the three points represent a feasible solution, the initial solution for the set (Figure 2.11).

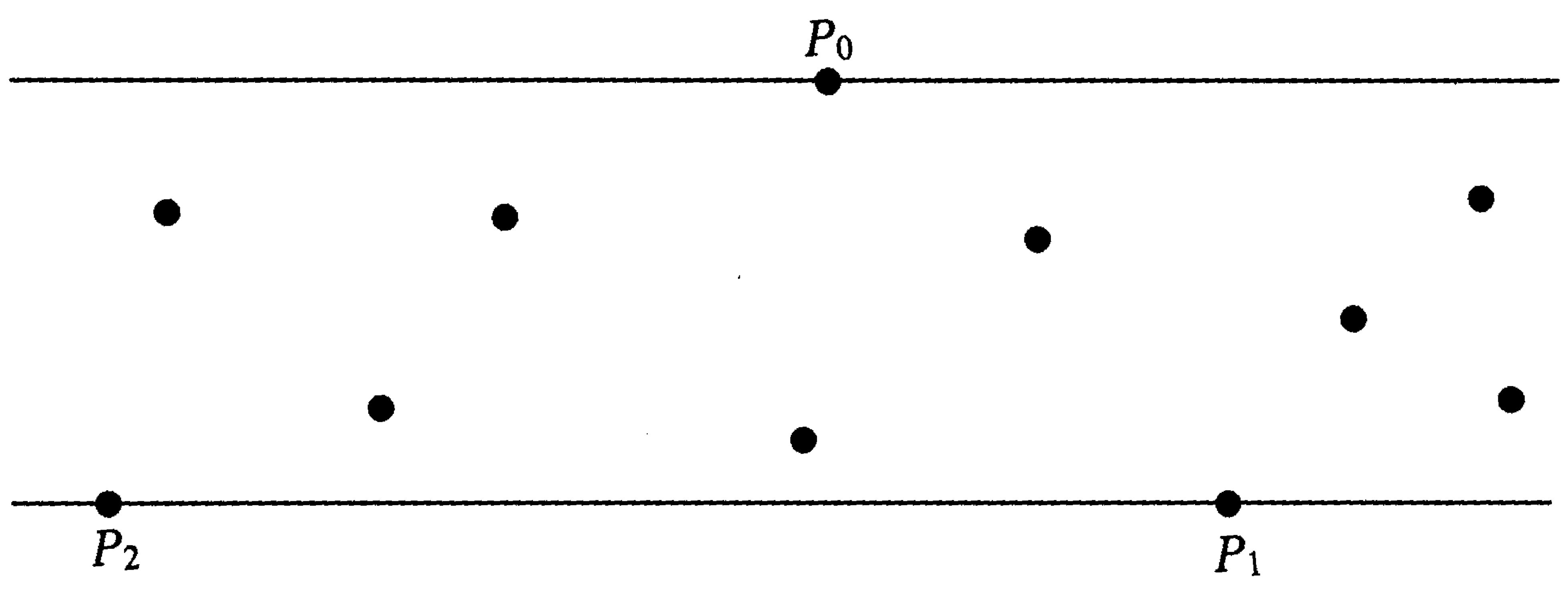


Figure 2.11 Initial solution.

Having the initial solution (*reference edge*) for the set, the next step is to find the minimum zone (*minimum edge*) for the three points and to check if they are equal as shown in Figure 2.5. The *reference edge* is given by the triangle edge which is located on the initial solution line. The *minimum edge* is the edge of the triangle which corresponds to the minimum zone for the triangle defined by the three points. If this condition is satisfied the algorithm takes the route 1 in Figure 2.7 and stops because the initial solution is the minimum zone. If the condition is not satisfied, the algorithm tries to find if there exists a bigger subset for which the initial solution is the minimum zone. The polygon formed by the three points, starts to grow by adding a new vertex at every step.

2.3.2 Polygon Expansion

When the minimum edge is not equal to the reference edge, the algorithm tries to find the furthest point outward from minimum edge as shown in Figure 2.12.

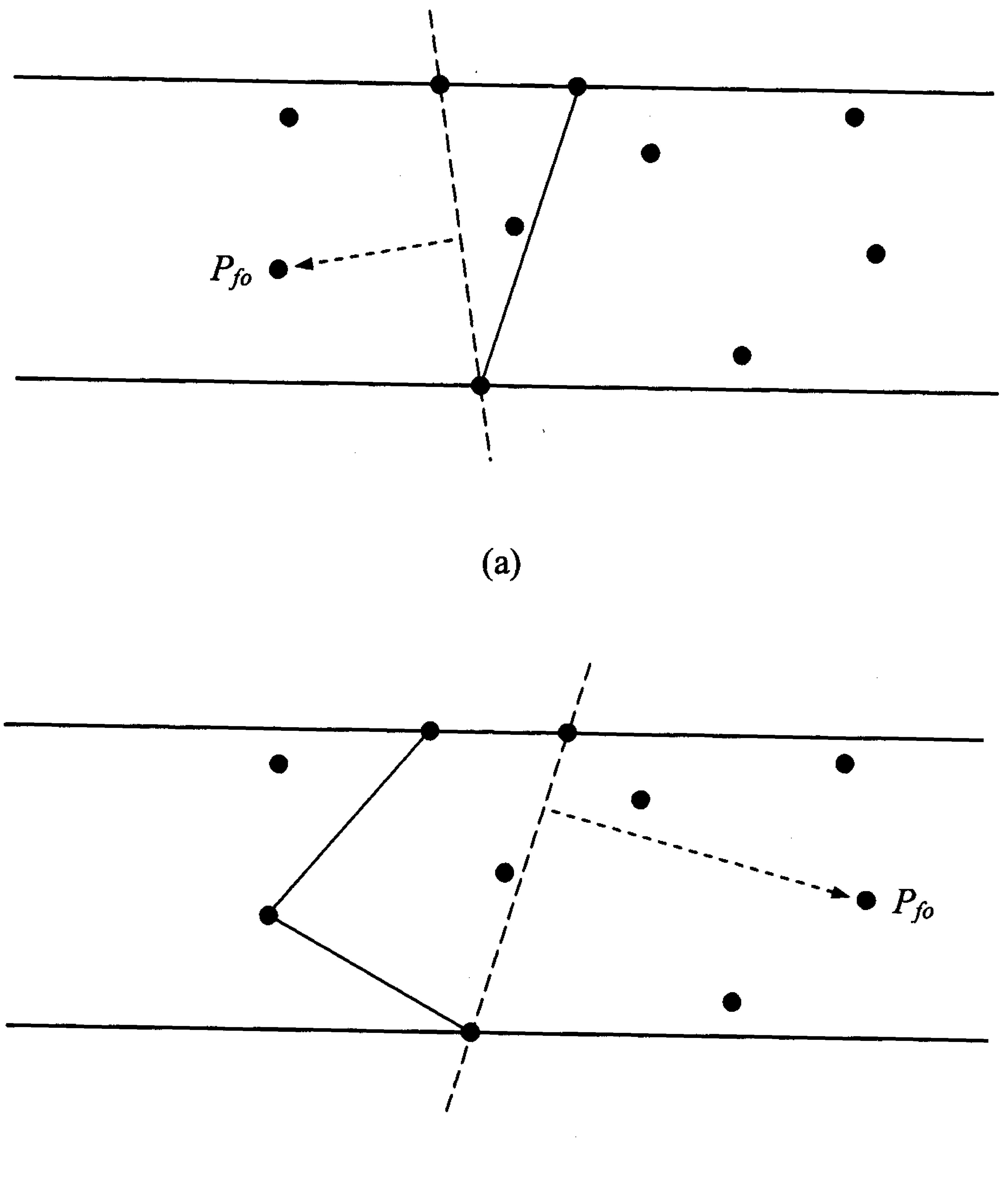


Figure 2.12 Furthest point outward from minimum edge: two typical cases.

The role of the furthest point outward is to enlarge the polygon as much as possible at each expansion step in order to find a polygon for which the initial solution for the set is the minimum zone.

In Figure 2.12, the dashed lines represent the minimum edges and the arrows point the furthest points outward (P_{fo}) from them. If there are points outside the polygon in the side of the minimum edge, the furthest one is chosen to be the new point added in the polygon. The new point is connected with the visible two points which in this case are the end points of the minimum edge and two new edges are created. After the new added point is connected with the visible points, the minimum edge remains inside the recent polygon and being out of the interest is deleted. The edges of the updated polygon are evaluated and the minimum edge for the updated polygon is found. If the new minimum edge does not coincide with the reference edge a new furthest point outward from the actual minimum edge is searched. If there is no point outward for the minimum edge, the polygon cannot be expanded in that direction and implies the minimum edge is an edge of the convex hull for the set of points. In this case the algorithm tries to find the furthest point inward from the minimum edge.

When there is no point outward for the minimum edge, the furthest point inward from the minimum edge is searched as shown in Figure 2.13. The minimum edge is an edge of the convex hull for the set of points and with its furthest point from the set defines a feasible solution for the set of points. The algorithm finds the furthest point inward from the minimum edge and checks if the new feasible solution is the minimum zone for the polygon. In the figure, the dashed line is the minimum edge and the arrow indicates the furthest point (P_{fi}) from it. If there are points inward for the minimum edge, the furthest one is selected and the polygon is updated. Two new edges are created between the new point and the two visible points. In this case the visible edge is not the minimum edge. The edge that the new point can see remains in the interior of the updated polygon after the construction of the new edges. Being in the interior it is not an edge anymore for the new polygon and is deleted. If there is no point inward for the minimum edge further than its opposite vertex from the polygon, the polygon cannot expand in that direction. Because there is no new furthest point outward or inward from the minimum edge, that minimum edge and its opposite vertex from the polygon represents a smaller feasible solution for the set than the initial one. This feasible solution for the set is the minimum zone for the subset because is given by the minimum edge and like discussed previously it is also the minimum zone for the set.

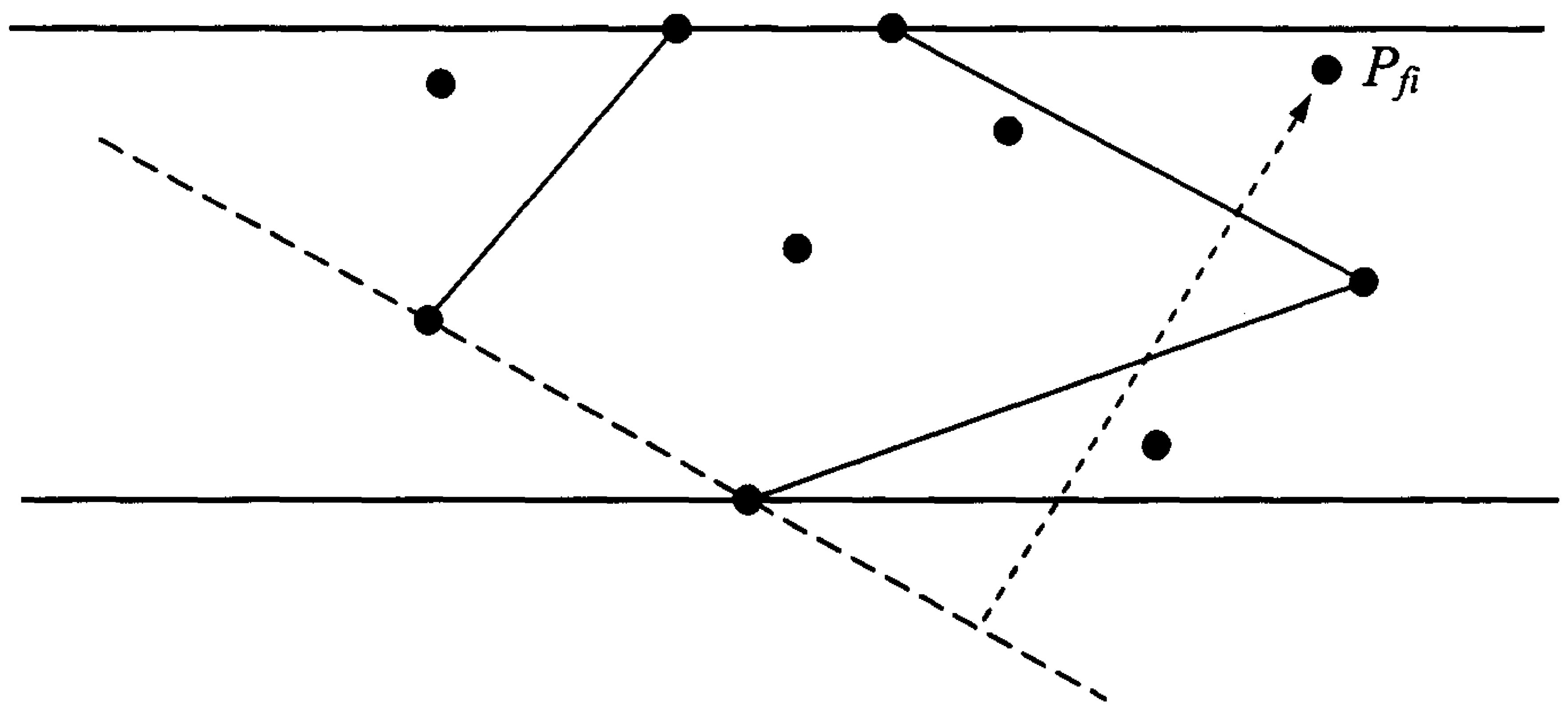
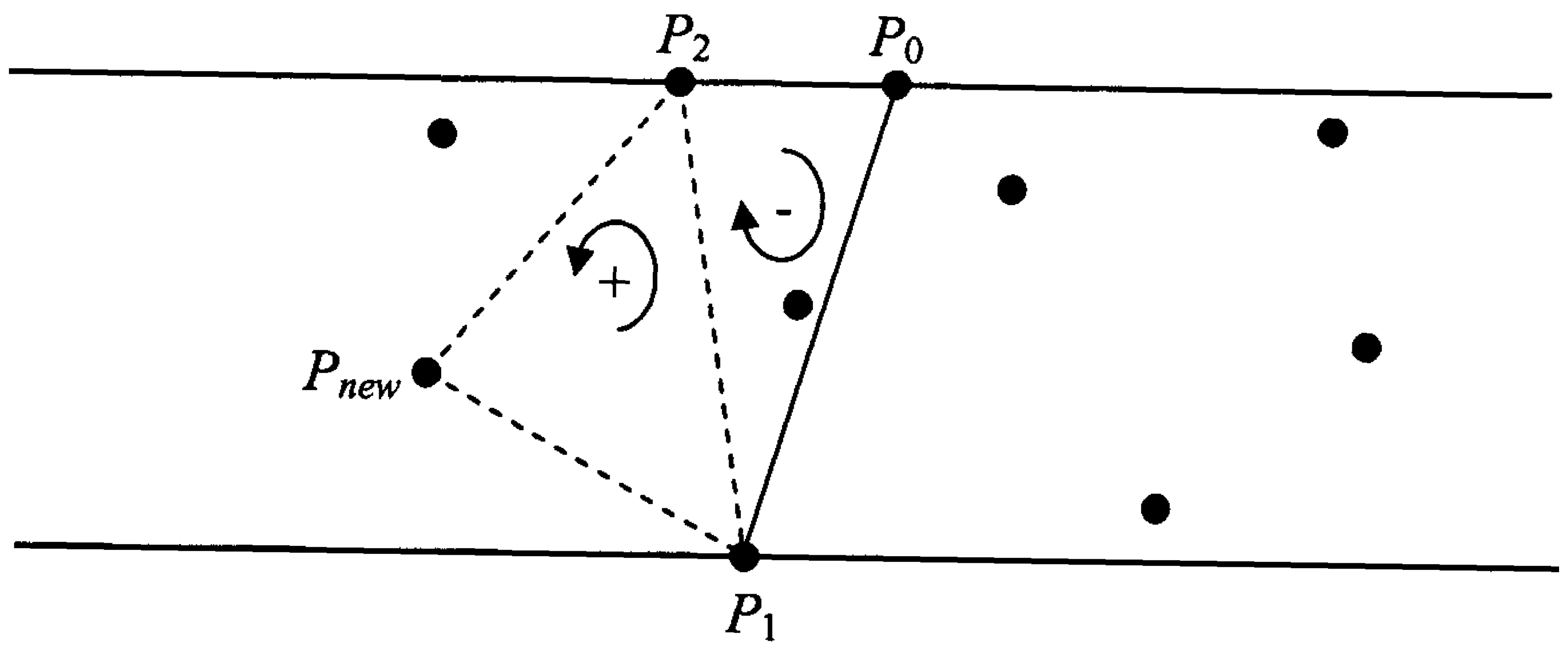


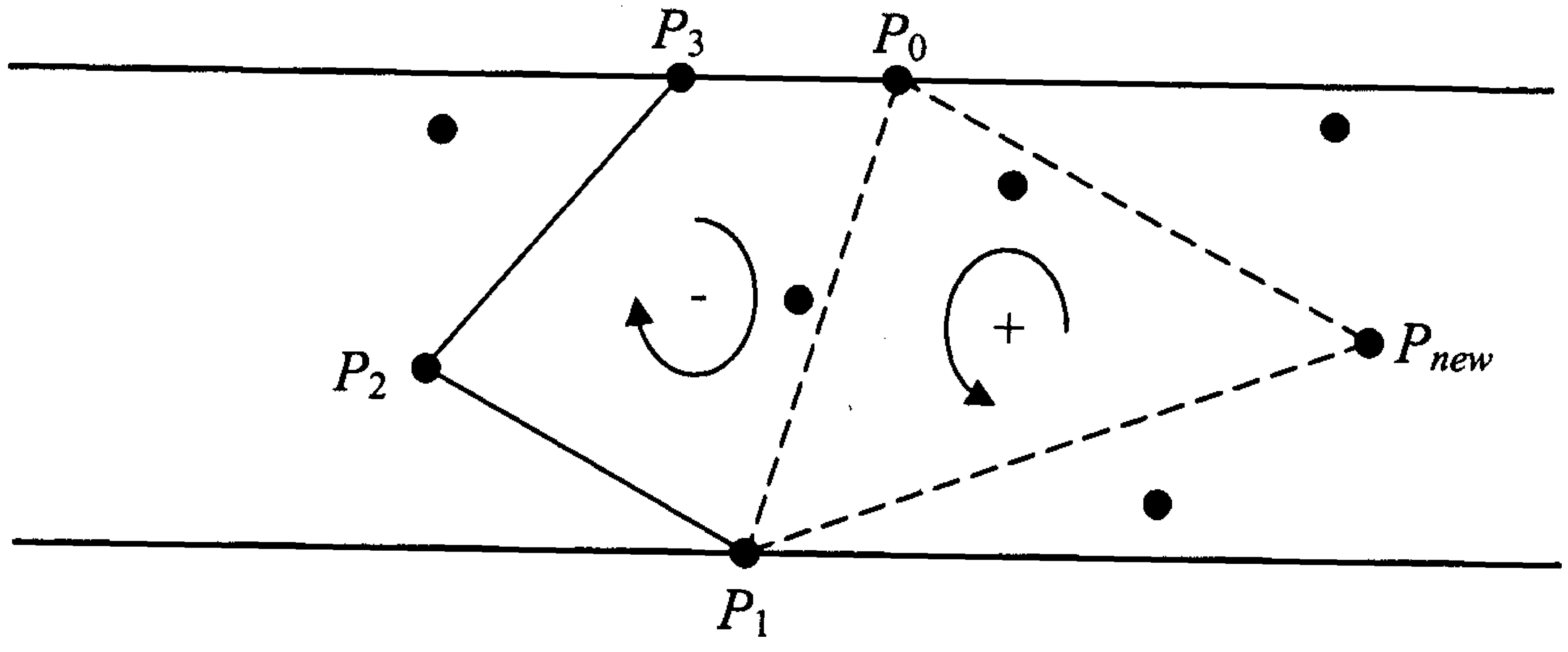
Figure 2.13 Furthest point inward from minimum edge.

The polygon expansion process starts with the triangle and continues with a quadrilateral and then a pentagon and so on. At each step a new point is added to the polygon. The new added points (P_{new}) are the furthest points outward (P_{fo}) or the furthest points inward (P_{fi}) from the minimum edge and because they are extreme points, they are vertices of the convex hull of the set of points. Each time when a new point is added to the existing polygon, two new edges are created and one is deleted. The two new edges connect the new point with the end points of the visible edge. The examples from Figures 2.12 and 2.13 were used to illustrate the connection of the points in Figure 2.14.

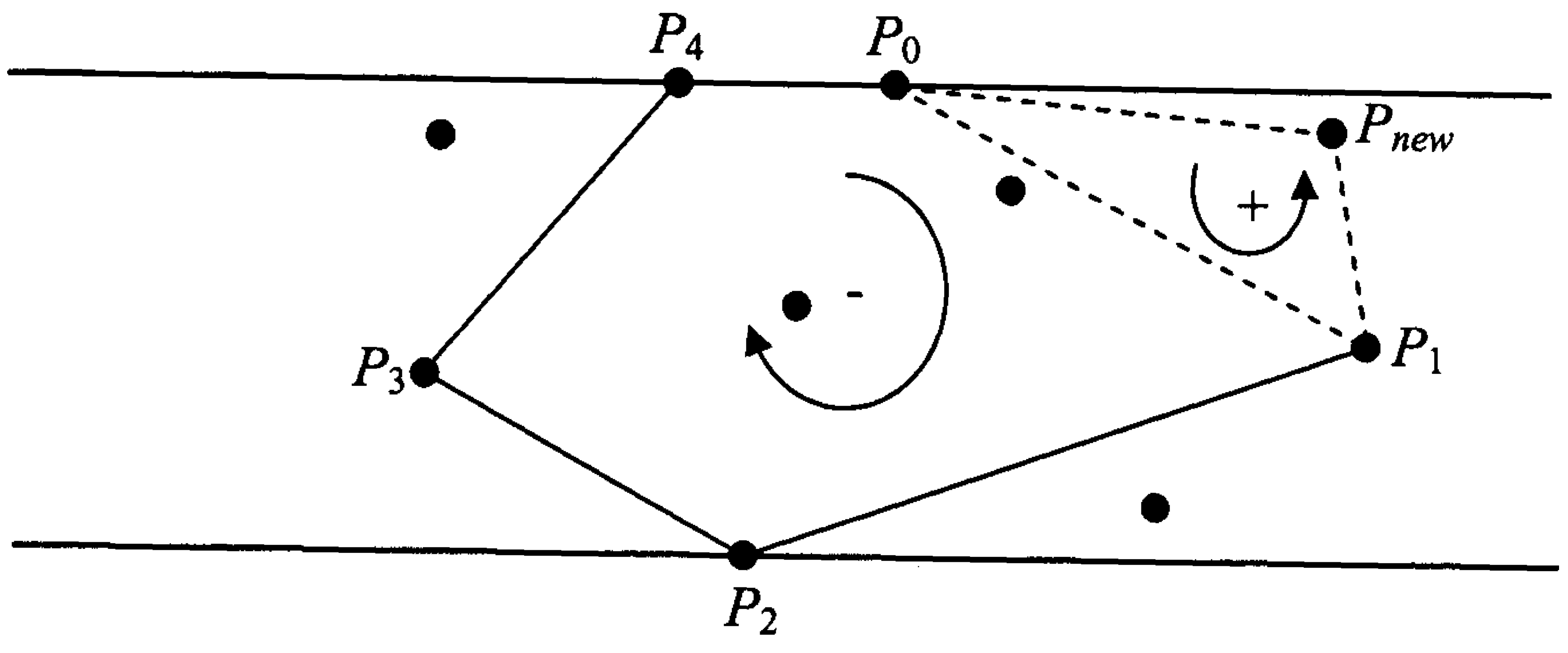
The visible edges and points are determined like discussed previously from the orientation of the vertices around the polygon. In the Figure 2.14 the orientations of the polygon vertices and the orientations of the new point with the visible vertices are different. The polygon is oriented in clockwise order which makes the area of any triangle defined by any three consecutive vertices of the polygon to have a negative sign. The new point (P_{new}) with the visible vertices from the polygon form a triangle oriented anti-clockwise and its area has a positive sign. The dashed lines are the visible edges and the new edges. The visible edge of the previous polygon is not an edge for the updated polygon and is deleted. The number of edges and vertices of the polygon grows one by one until the solution converges.



(a)



(b)



(c)

Figure 2.14 The new point connecting the end points of the visible edge.

Figure 2.15 presents the updated polygons for the examples from Figures 2.12 and 2.13. The bold edges are the two new edges of the new polygon. The visible edges are deleted after the connection of the new point with the visible points. The vertices are kept in the same orientation of the polygon. After a point is inserted in the polygon, the vertices are reordered as shown in Figure 2.15 so the polygon vertices are kept in the same orientation. In the first case the new added point is inserted between P_1 and P_2 and in the second and third case between P_0 and P_1. Therefore the new point in the first case becomes P_2 and in the second and third case the new point becomes P_1. All the next vertices are shifted one position. After each upgrade of the polygon, the edges are evaluated and if the condition is satisfied the solution is found, otherwise the process continues. In the worst case, the process stops when the constructed polygon coincides with the convex hull for the entire set of points.

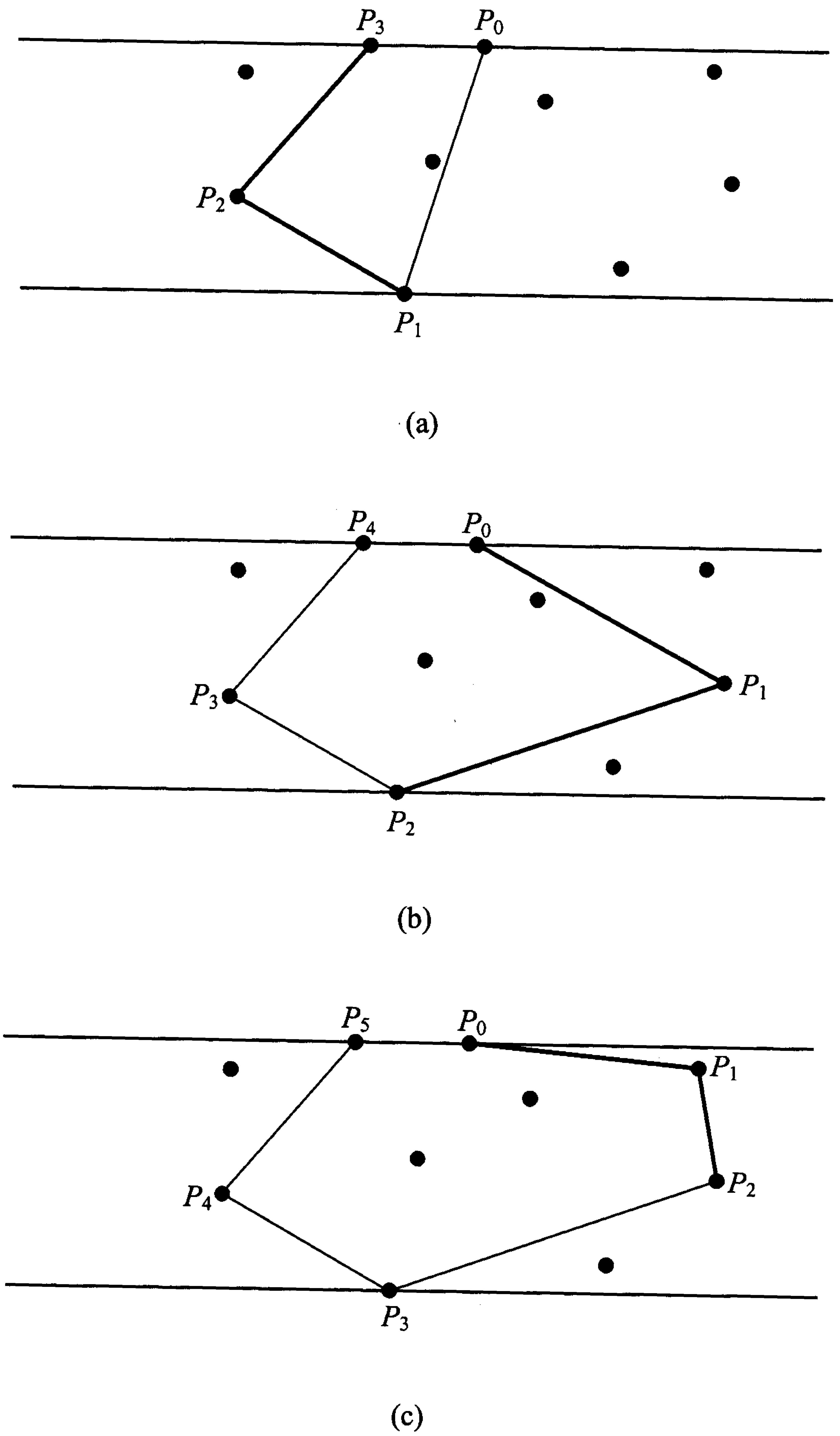


Figure 2.15 Update polygons.

2.3.3 Convergence Criteria

The algorithm guarantees to find the global solution. The edges are evaluated at the beginning of the process and at each polygon expansion step. Once the condition described in the beginning of this chapter is satisfied, solution converges and the algorithm stops. There are two possibilities:

1) The first possibility is the initial solution to be the converged solution (route 1 in Figure 2.7). The initial solution can be found to be the final solution at the beginning of the algorithm or after the polygon expansion.

The first case happens when least squares line is a good approximation of the points and by the smallest rotation angle principle the initial solution is found to be the final solution. If the initial solution (FS_s) is the minimum zone for the three points (MZ_3) it is also the minimum zone for the whole set (MZ_s). Figure 2.16 shows an example for this case.

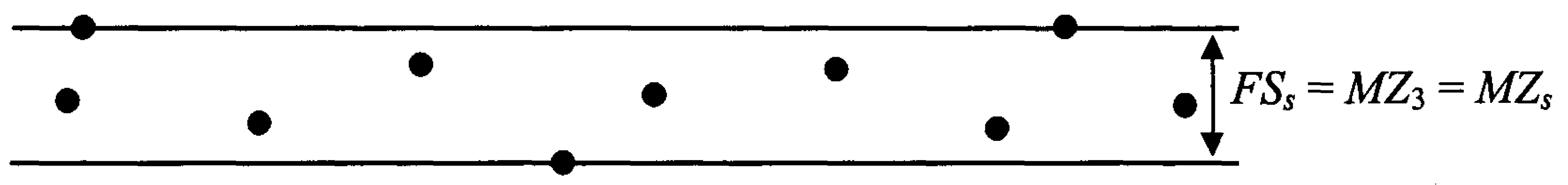


Figure 2.16 Initial solution is found to be the minimum zone without polygon expansion.

The initial solution can also be found to be the minimum zone after the polygon expansion. If the initial solution is not the minimum zone for the three points, the polygon starts to expand. The initial solution is the minimum zone for the set (MZ_s) if in the polygon expansion process is found a polygon for which the initial solution is the minimum zone ($FS_s = MZ_6$) as shown in Figure 2.17. MZ_6 denotes the minimum zone for the found polygon which contains six points. The minimum zone for this polygon coincides with the initial solution for the set which makes the last one to be the minimum zone for the set. The size of the polygon can be small, medium or in the worst case the polygon can expand up to the convex hull for the set.

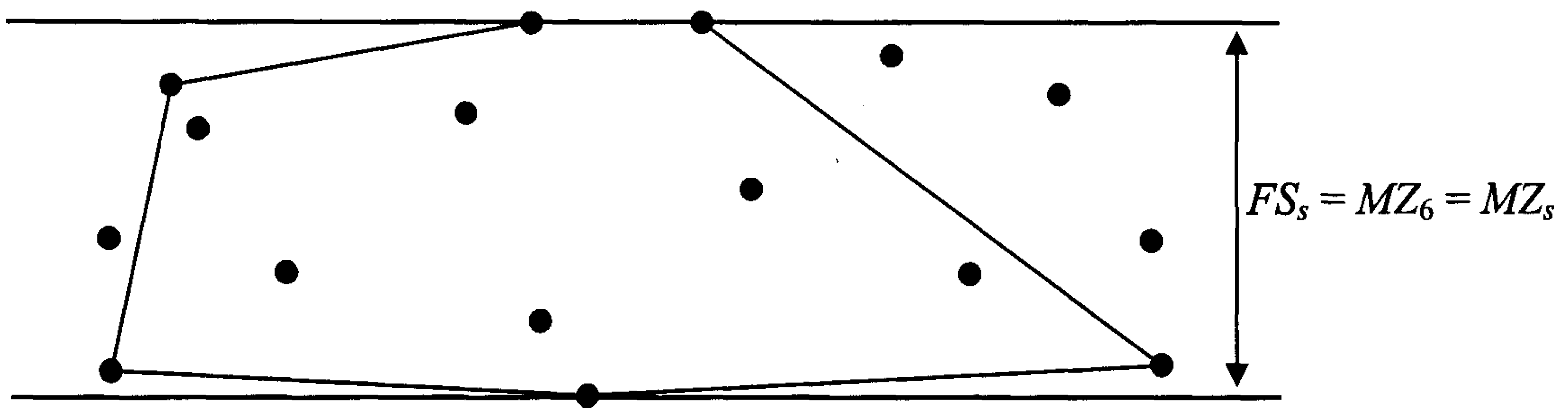


Figure 2.17 Initial solution is found to be the minimum zone with polygon expansion.

2) The second possibility is the initial solution to be only a feasible solution for the set and another one to be the minimum zone (route 2 in Figure 2.7). When the initial solution is not the minimum zone for the starting triangle, the polygon starts to grow. The polygon stops growing when there are no new furthest points outward and inward from the minimum edge. This implies the minimum edge is an edge of the convex hull and its opposite vertex from the polygon is also its opposite vertex from the convex hull. Because the expansion cannot continue the minimum edge and its opposite vertex define

a new feasible solution for the set. In this case the minimum edge does not coincide with the reference edge and the new feasible solution for the set (*FS_s'*) is smaller than the initial solution (*FS_s*), being given by the minimum edge of the polygon. In Figure 2.18 the dashed parallel lines represent the initial solution (*FS_s*) and the continuous parallel lines the new smaller feasible solution (*FS_s'*). The initial solution is only a feasible solution for the polygon and the new feasible solution for the set is the minimum zone for the polygon. The new feasible solution satisfy the condition of being the minimum zone for the subset and a feasible solution for the set, which guarantees that is the minimum zone for the entire set.

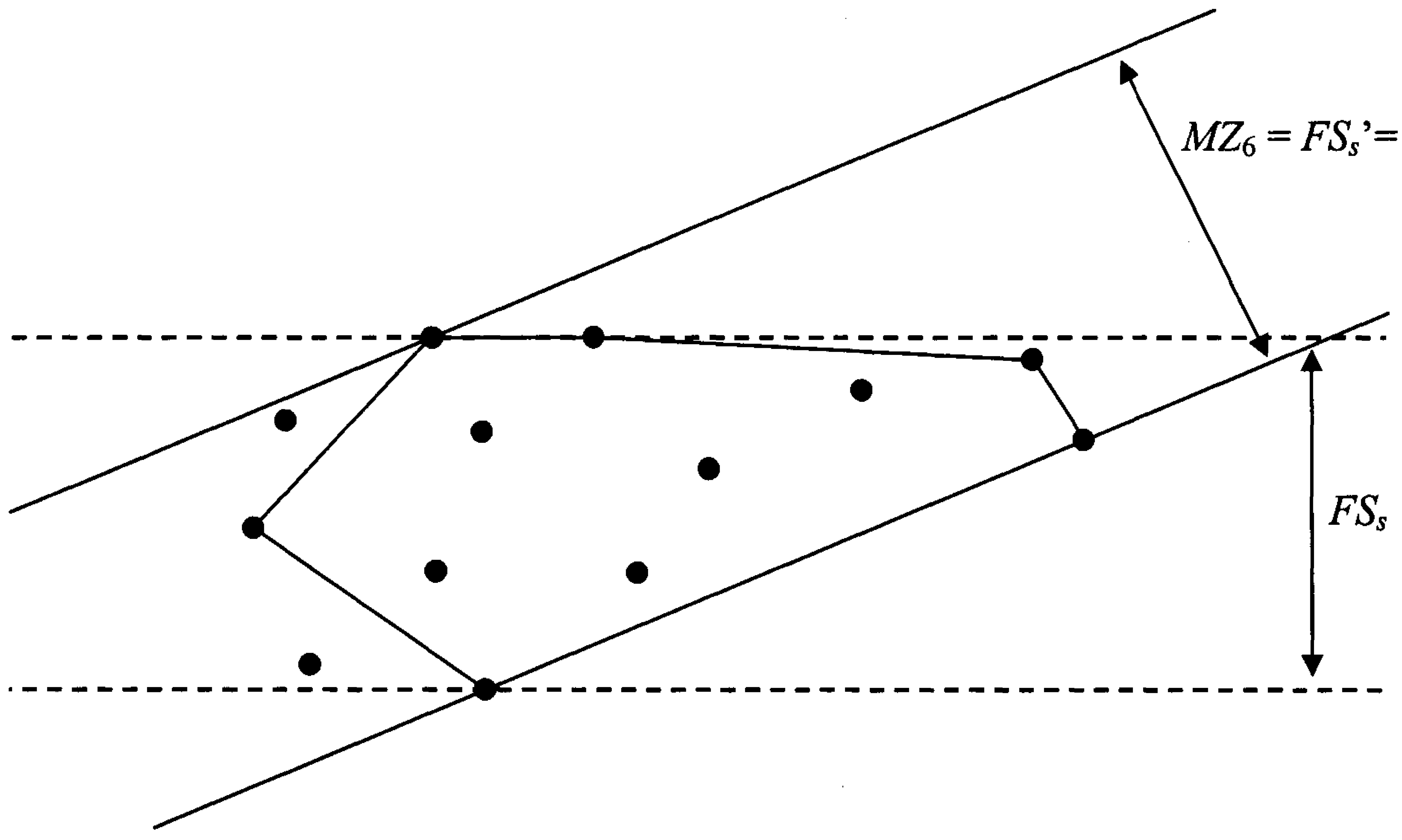


Figure 2.18 Initial solution is not the minimum zone.

CHAPTER 3

COMPUTATIONAL COMPLEXITY ANALYSIS

3.1 Background

Computational complexity is the branch of the theory of computation. It studies the resources, or cost, of the computation required to solve a given problem. This cost is usually measured in terms of abstract parameters such as time, which represents the number of steps it takes to solve a problem, and space represented by the quantity of information storage required. The time complexity is the most common used in the analysis of an algorithm. This is done by counting the key operations executed by the algorithm. It shows a general functional dependence of computation time upon problem size, how fast the computation time grows with the problem size. Only the leading term of the growing function is significant for finding the asymptotic efficiency of an algorithm. The leading term denotes the *rate of growth* or *order of growth*. The lower-order terms and the constant coefficients are ignored, since they are less significant than the rate of growth in determining computational efficiency for large inputs. An algorithm is considered more efficient than another one if its time has a lower order of growth. This may not be true for small inputs due to constant factors and lower-order terms. The symbols used to describe the asymptotic computing time of an algorithm are "Θ"(read "theta of"), "O"(read "big oh of"), "Ω"(read "omega of") [5].

The Θ-notation asymptotically bounds a function from above and below (Figure 3.1(a)). Θ($g(n)$) denotes the set of all functions $f(n)$ such that there exist positive constants c_1, c_2, and n_0 with:

$$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \quad \textit{for all } n \geq n_0 \tag{3.1}$$

A function $f(n)$ belongs to the set Θ($g(n)$) if there exist positive constants c_1 and c_2 such that it can be enclosed between $c_1g(n)$ and $c_2g(n)$, for sufficiently large n. $g(n)$ is an asymptotically tight bound for $f(n)$.

O-notation gives an upper bound on a function, to within a constant factor (Figure 3.1(b)). $O(g(n))$ denotes the set of all functions $f(n)$ such that there exist positive constants c and n_0 with:

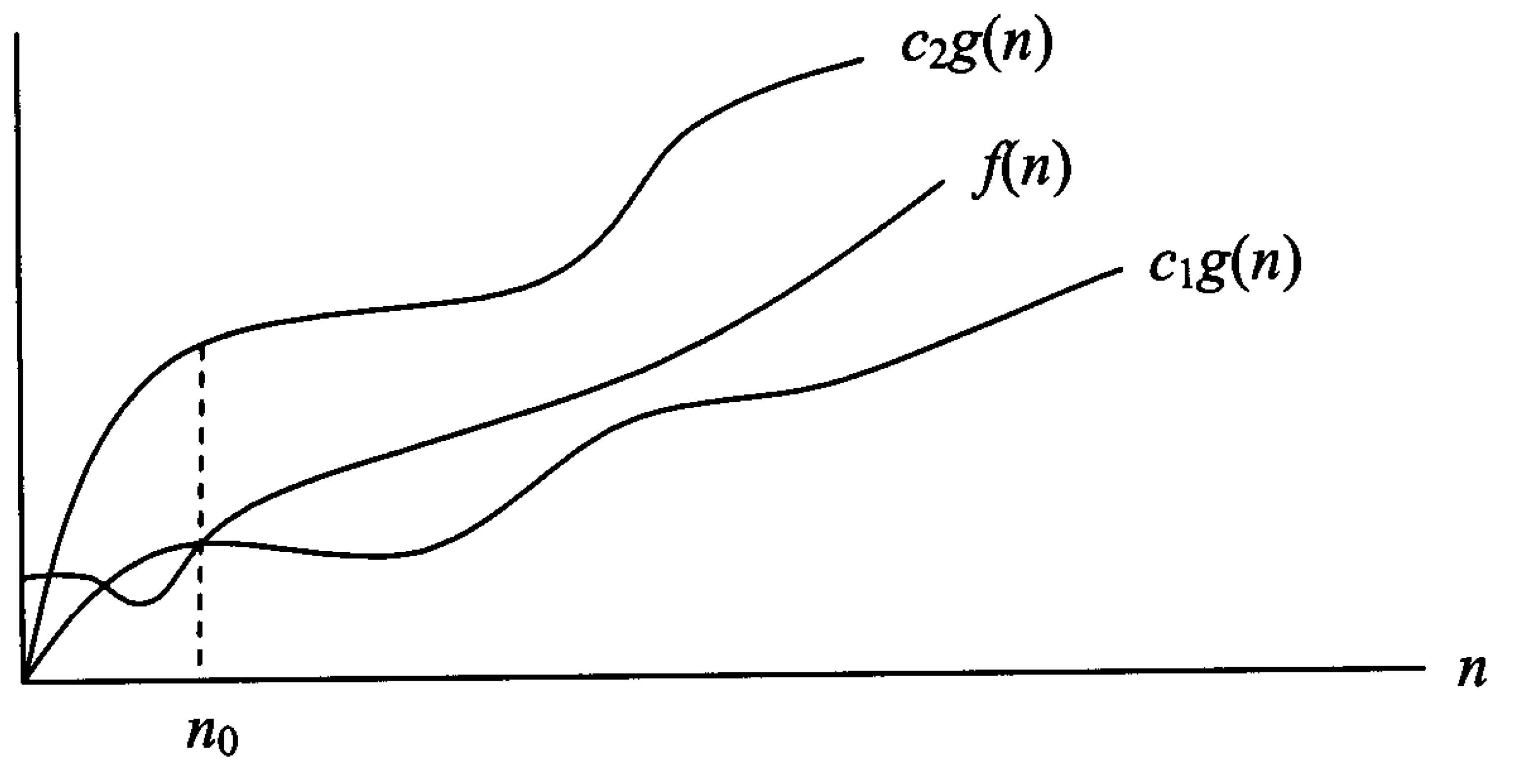
$$0 \leq f(n) \leq cg(n) \quad \textit{for all } n \geq n_0 \tag{3.2}$$

A function belongs to the set $O(g(n))$ if there are positive constants n_0 and c such that for all $n \geq n_0$, the value of $f(n)$ always lies on or below $cg(n)$.

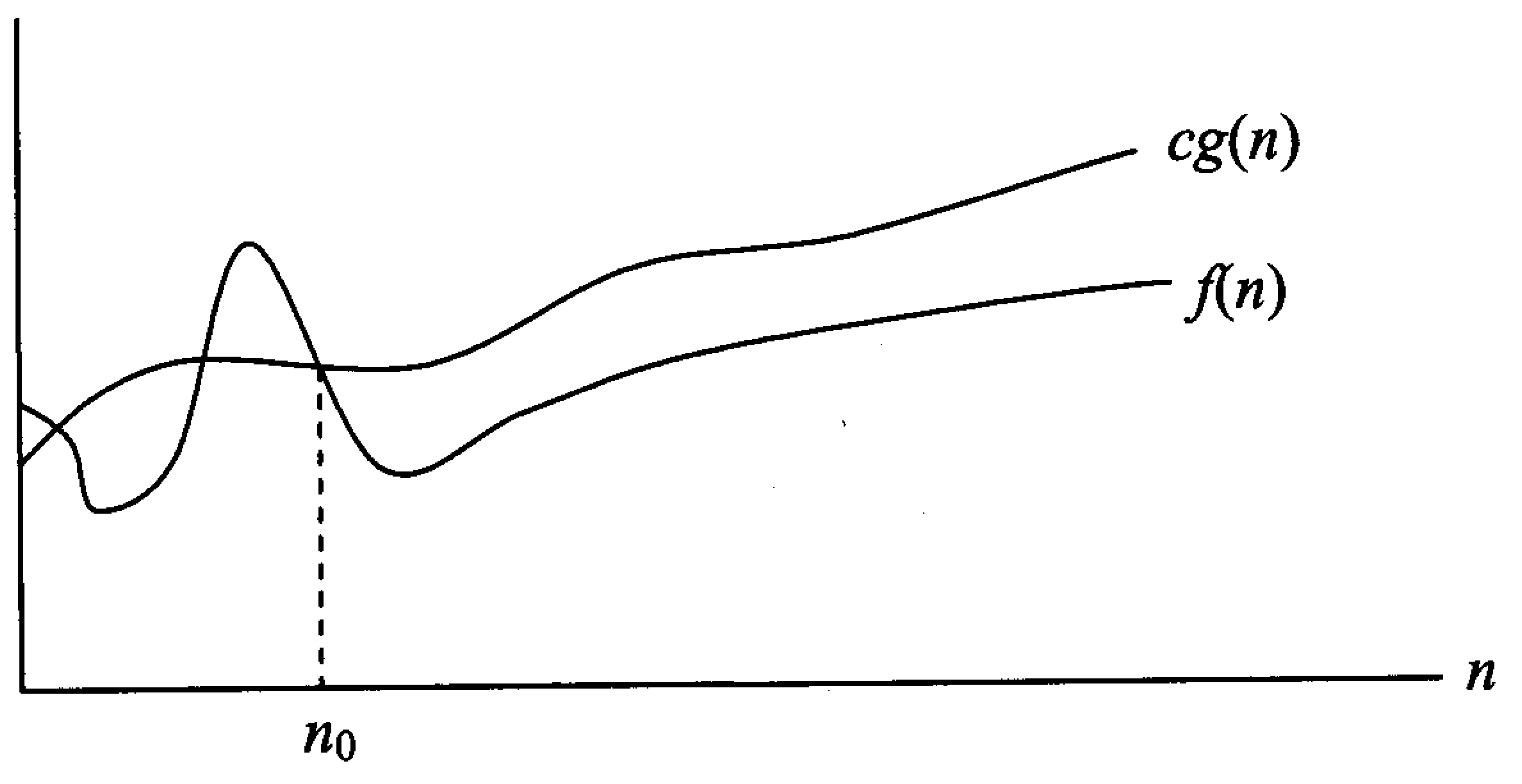
Ω-notation provides an asymptotic lower bound for a function (Figure 3.1(c)). Ω($g(n)$) denotes the set of all functions $f(n)$ such that there exist positive constants c and n_0 with:

$$0 \leq cg(n) \leq f(n) \quad \textit{for all } n \geq n_0 \tag{3.3}$$

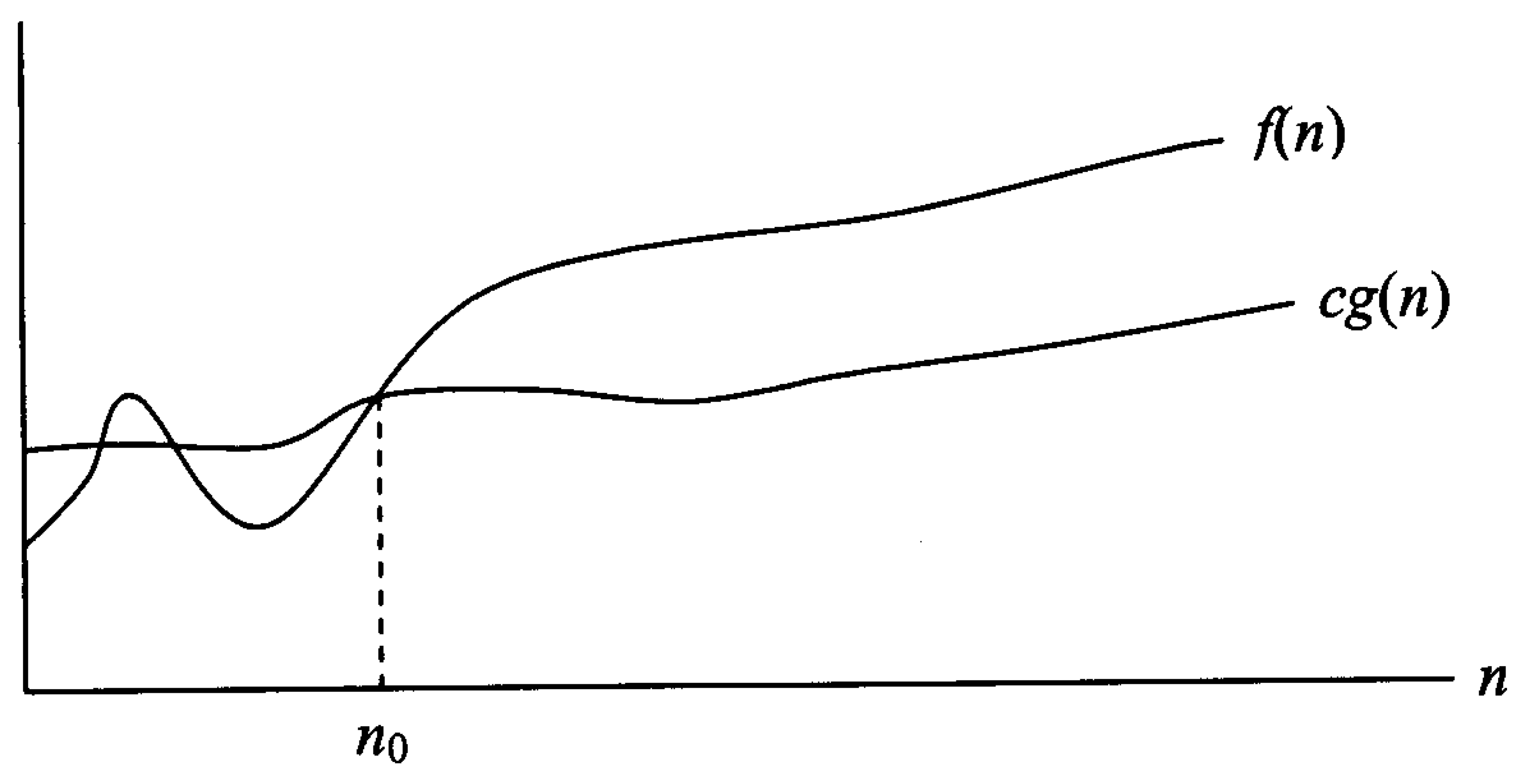
A function belongs to the set Ω($g(n)$) if there are positive constants n_0 and c such that for all $n \geq n_0$, the value of $f(n)$ always lies on or above $cg(n)$.



(a)



(b)



(c)

Figure 3.1 Graphic examples of the asymptotic notations: (a) Θ; (b) O; (c) Ω.

Using *O*-notation, the computing time of an algorithm can be described by inspecting the algorithm's overall structure. The analysis of the present algorithm is based on the number of iterations, operations required to solve the problem.

3.2 Complexity Function Derivation

The time used by a computation is the sum of the times of the individual operations being executed. The complexity of the algorithm is given by the highest complexity of all the modules of the algorithms (Figure 2.7). The complexity of the first module (*linear least squares line*) is $O(n)$, where n is the number of points in the set. The coordinates of the points are substituted in the Equations (1.3) and (1.4) and the parameters of the line are found. The second module (*initial solution*) calculates the distances from all the points to the least squares line in order to find the furthest two points. The third point is the first point touched by one of the enclosing parallel lines when they rotate around the previous two points they are fitted through. Therefore, the smallest angle between each initial line and the lines connecting the two initial points with all the other points is selected. Because all the points are involved and tested the complexity is $O(n)$. Another module is *minimum edge.* For each edge the furthest vertex is found, so for each edge are tested (h-2) points. The symbol h represents the number of points and edges of the polygon. There are (h-1) edges to test, which implies (h-1)*(h-2) iterations. The module is executed once for the three starting points and each time the polygon is updated which implies (h-2) times, so its complexity is $O(h^3)$. The modules *furthest point outward from min. edge* and *furthest point inward from min. edge* test (n-h)

points. The number of tested points can be reduced if at each step the interior points of the polygon are deleted. Because of the repetitions ((h-3) times), their complexity is $O(h(n-h))$ which can be approximated with $O(nh)$. The complexity of *update polygon* module is $O(h^2)$. The number of operations for finding the position of the new point in the polygon is (h-1) and for inserting the new point in the polygon in the worst case is (h-2) and number of repetitions is (h-3). The total complexity of the algorithm is the summation of all the module complexities:

$$\textit{Total complexity}: \quad O(n)+O(n)+O(h^3)+O(nh)+O(h^2) \tag{3.4}$$

Worst case complexity of an algorithm is the maximum of a measure of performance of the algorithm over all problem instances of a given size [5, 20]. In the present algorithm the worst case arises when the subset or polygon grows a lot and h is very close or equal to n. They are equal ($h = n$) when all the points are on the polygon. In this case the initial solution is not a good starting solution and the polygon grows until all the points are on the polygon. Substituting $h = n$ in Equation (3.4), the worst case complexity is:

$$O(n)+O(n)+O(n^3)+O(n^2)+O(n^2)=O(n^3) \tag{3.5}$$

Similarly with the worst case, the best case complexity of the algorithm is the function defined by the minimum number of steps taken on any instance of size n. The best case complexity is $O(n)$. This happens when the least squares line is a good approximation of the points and initial solution is found directly to be the final solution. The polygon is not expanding at all because the initial solution satisfies the condition for minimum zone for the starting three points. In this situation the size of the polygon is minimum ($h = 3$). Therefore the functions *furthest point outward from min. edge*, *furthest point inward from min. edge* and *update polygon* are not executed.

3.3 Expected Computational Complexity

Expected or average case complexity gives an estimate of the observed behavior of the algorithm [20]. To analyze the expected case performance, only the expected value of h needs to be computed. The expected size of the convex hull for points uniformly distributed in a circle is $O(n^{1/3})$. The convex hull for points uniformly distributed in any convex polygon has $O(\log n)$ expected size [5, 20]. The polygon obtained in the proposed method is smaller than the convex hull for the entire set of points. Only in the worst case when the polygon expands to the maximum they are equal.

In almost all the cases the polygon is smaller than the convex hull for the set of points. The size of the convex hull is expected to be $O(\log n)$ for uniform distribution of points in a convex polygon. The size of the polygon in the proposed method is expected to be less than $O(\log n)$, being smaller than the size of the convex hull. Thus, substituting $h = \log n$ in Equation (3.4), the expected complexity becomes:

$$O(n) + O(n) + O(\log^3 n) + O(n \log n) + O(\log^2 n) = O(n \log n) \tag{3.6}$$

This is the expected complexity of the algorithm if the obtained polygon is equal to the convex hull for the set. This is a rare case, the expected size of the polygon is less than $O(\log n)$ and the expected complexity of the algorithm is less than $O(n \log n)$. This is also confirmed by the experimental results.

CHAPTER 4

IMPLEMENTATION RESULTS AND DISCUSSION

The two main advantages of the proposed method are the accuracy of the obtained results and a less expected computational complexity. The accuracy is assured by the computational geometric approach. The expanded polygon is a convex polygon and usually is only a small part of the convex hull for the set of points. The expected computational complexity is less than the complexities of the existing methods based on computational geometry which can guarantee the global solution. Those techniques construct the convex hull and then they find the minimum zone for the convex hull which is identical with the minimum zone for the set. In the presented algorithm, the obtained polygon is only a part of the convex hull and it enlarges only as much is required in order to find the final solution. The expansion stops when the necessary part is obtained.

4.1 Accuracy

The computer program was written in C++ based on the presented algorithm. The reported data points in the literature for straightness error were used to prove the accuracy of the proposed method. There are seven sets containing different number of points. The smallest set of points contains 5 points and the largest one contains 25 points. It was considered that there is no error in the measurement. The obtained results are compared in Table 4.1 with the previously reported results. The second column of the table contains the current results. There are two values presented for each data set in this column of the table. The first value contains number of digits equal with the previously published result which has the smallest number of digits for the same set of points. The second value presented in the bracket contains the same number of digits as the published result with the largest number of digits for that set of points. The results show the present method is accurate and converges to the global solution even if the initial solution is not the final one. Using computational geometric technique based on the minimum zone condition guarantees precise results. The results match the best of the previously published results. The obtained results are the same with the results from convex hull based techniques [4, 6], and they are better than some other methods results [16, 19]. For the third data set Traband et al. [4] published a smaller result than the others, but Cheraghi et al. [8] showed that is a programming implementation error or a typing error. As discussed in the introduction, the control line rotation scheme method introduced by Huang et al. [19] cannot guarantee the global solution. For the third data set the control line rotation scheme result has a bigger value than the other methods. Also Sharma et al. [16] published a larger value for the sixth data set than the other methods. For these particular data sets all the techniques present satisfactory results.

Table 4.1 Comparison of current and published accuracy results.

Data Set (No. of points)	Current Result (mm)	Published Results (mm)
1 (5 points)	0.002667 (0.0026667)	0.0026666[6]; 0.002666[7,10,14]; 0.002667[11,15];
2 (5 points)	2.1213	2.1213 [4,7,8,9,11]; 2.1214[12,16]
3 (10 points)	0.86 (0.8578577)	0.8479[4]; 0.8578[7,8,12]; 0.8578577[10,11]; 0.8579[16]; 0.88[19]
4 (20 points)	0.1646	0.1645[7]; 0.1646[4,8,11,12]; 0.1647[16]
5 (15 points)	0.0052 (0.005185658)	0.005186[4,8,9,11]; 0.0051856[6]; 0.005185[7]; 0.0052[12,16]; 0.005185658[17]
6 (25 points)	0.0013 (0.00131129)	0.001311[4,7,8,9,10]; 0.0013112[6] 0.0013113[11]; 0.0013[12]; 0.001317[16]; 0.00131129[17];
7 (10 points)	5.5 (5.493)	5.493[7,8]; 5.5[19];

4.2 Computational Efficiency

The expected size of the convex hull for a set of points is $O(\log n)$ for uniform distribution of points in a polygon and $O(n^{1/3})$ for distribution of points in a circle [5, 20]. For uniform sampling within any bounded domain, the convex hull of a random set tends to assume the shape of the boundary of the domain. For a polygon, points accumulating in the corners cause the resulting convex hull to have few vertices. Because the circle has no corners, the expected number of convex hull vertices is comparatively high.

The simulated data points do not correspond to straightness data. The purpose of the simulated data points is to allow a comparison between the size of the obtained polygon and the size of the convex hull. Therefore, the points were uniformly, randomly distributed in a unit diameter circle, unit square, and three rectangles with 1×2, 1×5 and 1×10 ratios centered at the origin (0, 0) in the Cartesian coordinate system in the Euclidean plane as shown in Figure 4.1. The circle diameter and the square side length is 1 unit. The rectangles width is 1 unit and the lengths are 2, 5 and 10 units respectively. For a better visualization, the shapes in Figure 4.1 were enlarged. The unit diameter of the circle and the unit length side of the square in this figure corresponds to 1.4 cm. The rectangles widths corresponds to 1.4 cm and the lengths corresponds to 2.8 cm, 7 cm and 14 cm respectively. The points were generated with the *unifrnd* function in Matlab. This function returns a matrix of random numbers chosen from the continuous uniform distribution on the specified interval. To avoid the repetitions of the coordinates x and y of the points and for a better precision, the points were generated with sixteen decimals. The number of points in the sets varies from ten to one million (10, 50, 100, 500, …, 1000000) for each distribution shape. For each distribution shape and for each size set eleven sets were processed and the results were averaged. The results contain the final size of the constructed polygon needed to find the minimum zone for each set of points.

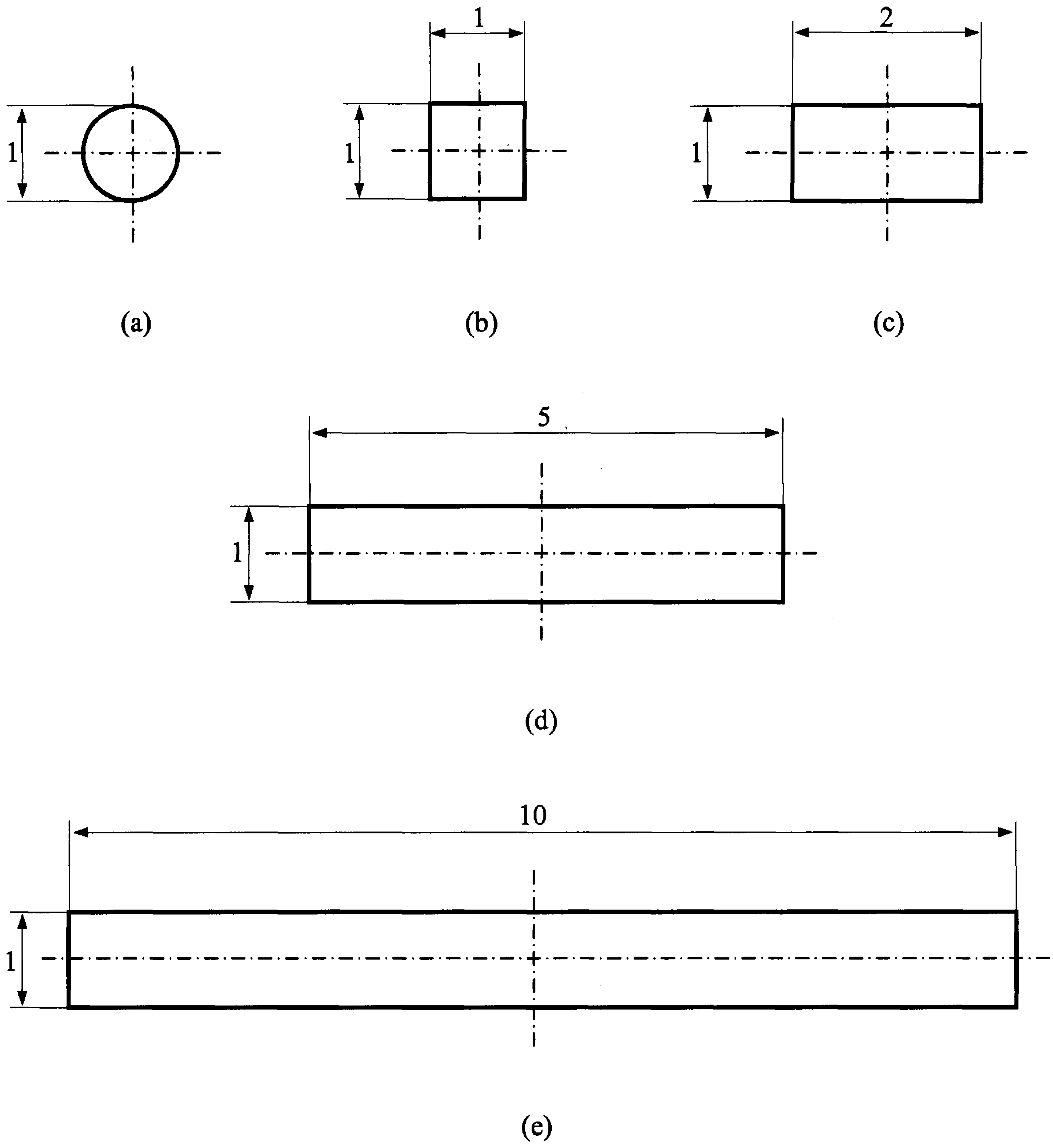


Figure 4.1 The distribution shape of points (1 unit ≡ 1.4 cm): (a) unit diameter circle; (b) unit square; (c) rectangle 1×2; (d) rectangle 1×5; (e) rectangle 1×10.

The expected size of the convex hull for a set of points was shown to be $O(\log n)$ for uniform distribution of points in a polygon and $O(n^{1/3})$ for distribution in a circle [5, 20]. Therefore, the convex hull for the tested sets of points was constructed to compare the obtained size of the polygon with the expected size of the convex hull. For the construction of the convex hull for the sets of points the *qhull* method was used. This algorithm starts by finding two points either with the greatest and least y-coordinates, either the points with the greatest and least x-coordinates. The points are connected with a line and the point from the set which is the furthest point from the line is found. These three points form a triangle and the process continue by finding the furthest points from each edge of the triangle. At each step the interior points are deleted. The polygon increases until there is no point outside of each edge and obviously outside the polygon. The final polygon is the convex hull for the set of points. A source code is available and can be downloaded from *The Geometry Center Home Page.* This technique is very efficient having a $O(n^2)$ complexity in the worst case when all the points are on the boundary of the convex hull and no points are thrown away and a $O(n \log n)$ complexity on the average [20].

The number of points of the obtained polygon is represented by h and the number of points of the convex hull for a set of points is represented by H. The sizes of the polygons and convex hulls depend on the distribution of points. Even for the same distribution shape of points and for the same set size they vary from set to set. Due to this variation of sizes of the polygons and convex hulls, eleven sets of points were generated from each set size. The sizes of the obtained polygons and convex hulls were averaged over the eleven sets employed, being notated h_{avg} and H_{avg}.

4.2.1 Distribution of Points in a Unit Diameter Circle

For the circle distribution of points, a circle with the diameter equal to 1 unit centered at (0, 0) was chosen. The points were generated randomly and uniformly distributed in the circle. Therefore the minimum value of x and y coordinates is -0.5 and maximum is 0.5. The sets of points contain [10 50 100 500 1,000 5,000 10,000 50,000 100,000 500,000 1,000,000] points. From each of these sizes were generated eleven sets of points. Figure 4.2 shows the plots of four sets of points distributed in the mentioned circle. For a better observation of the points, the size of the circle in Figure 4.2 was enlarged seven times with respect to the real size. These plotted sets of points were chosen to be the middle sizes sets [500 1,000 5,000 10,000] due to the density of the points. The smaller sets and larger sets were not plotted because, for this particular scale, in the smaller sets [10 50 100] the points are sparse, while in the large sets [50,000 100,000 500,000 1,000,000] the points are too dense and the circle appears as a solid. Eleven sets of points were generated from each size and the resulting sizes of the polygons and convex hulls were averaged. All the obtained results are presented in Table 4.2. The first column of the table contains the number of points (n) in each set. The second and third columns present the sizes of the obtained polygons and convex hulls respectively. The sizes of the polygons (h) and convex hulls (H) were arranged in an ascending order for each number of input points (n). The fourth column contains the ratios between the convex hulls and the obtained polygons for each set of points. The shaded cells of the table contains the minimum and maximum obtained sizes of the polygons and convex hulls for each set size.

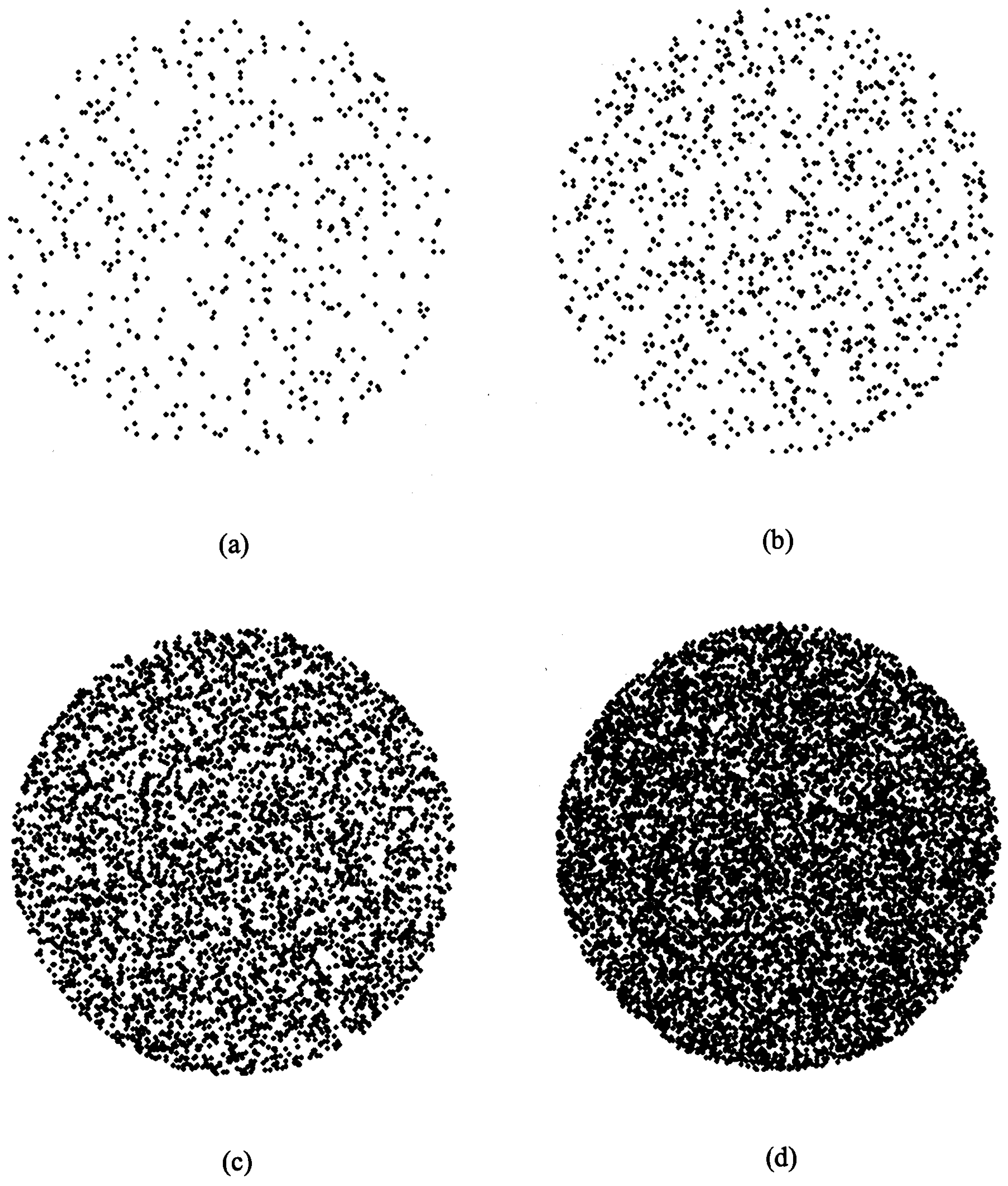


Figure 4.2 Different number of points distributed in unit diameter circle (1 unit ≡ 7 cm): (a) 500 points; (b) 1,000 points; (c) 5,000 points; (d) 10,000 points.

Table 4.2 The results obtained from the processed sets for circle distribution.

n	h	H	H/h
10	3	5	1.667
	4	5	1.250
	4	5	1.250
	4	5	1.250
	4	6	1.500
	5	6	1.200
	5	6	1.200
	5	6	1.200
	6	7	1.167
	6	7	1.167
	7	8	1.143
5×10	7	10	1.429
	7	10	1.429
	7	10	1.429
	8	10	1.250
	8	10	1.250
	8	11	1.375
	8	12	1.500
	8	13	1.625
	9	14	1.556
	9	14	1.556
	11	14	1.273
10^2	8	13	1.625
	9	13	1.444
	9	14	1.556
	10	14	1.400
	10	14	1.400
	11	16	1.455
	11	17	1.545
	11	17	1.545
	11	17	1.545
	12	18	1.500
	14	18	1.286
5×10^2	14	20	1.429
	14	22	1.571
	16	25	1.563
	16	26	1.625
	18	27	1.500
	18	28	1.556
	18	28	1.556
	19	28	1.474
	19	28	1.474
	20	28	1.400
	20	29	1.450
10^3	17	27	1.588
	17	31	1.824
	18	32	1.778
	18	32	1.778
	20	33	1.650
	20	34	1.700
	21	34	1.619
	22	34	1.545
	22	35	1.591
	23	35	1.522
	24	39	1.625
5×10^3	31	55	1.774
	32	56	1.750
	32	56	1.750
	32	56	1.750
	33	58	1.758
	34	58	1.706
	34	59	1.735
	35	59	1.686
	36	61	1.694
	36	62	1.722
	37	67	1.811
10^4	32	67	2.094
	34	69	2.029
	35	71	2.029
	36	72	2.000
	36	72	2.000
	38	72	1.895
	39	73	1.872
	42	75	1.786
	42	75	1.786
	42	75	1.786
	46	78	1.696
5×10^4	62	120	1.935
	63	121	1.921
	64	121	1.891
	64	122	1.906
	65	123	1.892
	66	124	1.879
	66	125	1.894
	67	127	1.896
	68	132	1.941
	68	135	1.985
	72	135	1.875
10^5	72	145	2.014
	73	152	2.082
	75	153	2.040
	75	156	2.080
	76	157	2.066
	76	157	2.066
	77	157	2.039
	77	158	2.052
	77	161	2.091
	80	162	2.025
	81	167	2.062
5×10^5	110	258	2.345
	110	263	2.391
	123	265	2.154
	129	267	2.070
	135	267	1.978
	136	268	1.971
	139	274	1.971
	139	275	1.978
	140	277	1.979
	142	279	1.965
	144	292	2.028
10^6	133	326	2.451
	134	330	2.463
	139	330	2.374
	153	331	2.163
	155	334	2.155
	157	335	2.134
	161	336	2.087
	162	338	2.086
	167	343	2.054
	169	350	2.071
	170	350	2.059

Table 4.3 contains the averaged values for the obtained polygons and convex hulls for each set size. The fourth column of the table includes the ratios between the averaged sizes of the convex hulls and polygons. The fifth column shows the average of the ratios between the convex hulls and polygons for each set of points. It can be seen that the size of the polygon and the size of the convex hull increase with the number of points. Also their ratio grows with the number of points. This is also graphically shown in the Figure 4.3 where the ratio between the sizes of the convex hulls and obtained polygons increases with the number of points. This implies that the polygon has a smaller order of growth than the one of the convex hull. The size of the polygon is observed to be half the size of the convex hull for large sets of points.

Table 4.3 Polygons and convex hulls sizes for points distributed in a unit diameter circle.

Circle				
n	h_{avg}	H_{avg}	$H_{\text{avg}} / h_{\text{avg}}$	$(H / h)_{\text{avg}}$
10	4.818	6.000	1.245	1.289
50	8.182	11.636	1.422	1.427
100	10.545	15.545	1.474	1.504
500	17.454	26.273	1.505	1.517
1,000	20.182	33.273	1.649	1.668
5,000	33.818	58.818	1.739	1.742
10,000	38.364	72.636	1.893	1.909
50,000	65.909	125.909	1.910	1.914
100,000	76.273	156.818	2.056	2.058
500,000	131.545	271.364	2.063	2.084
1,000,000	154.545	336.636	2.178	2.196

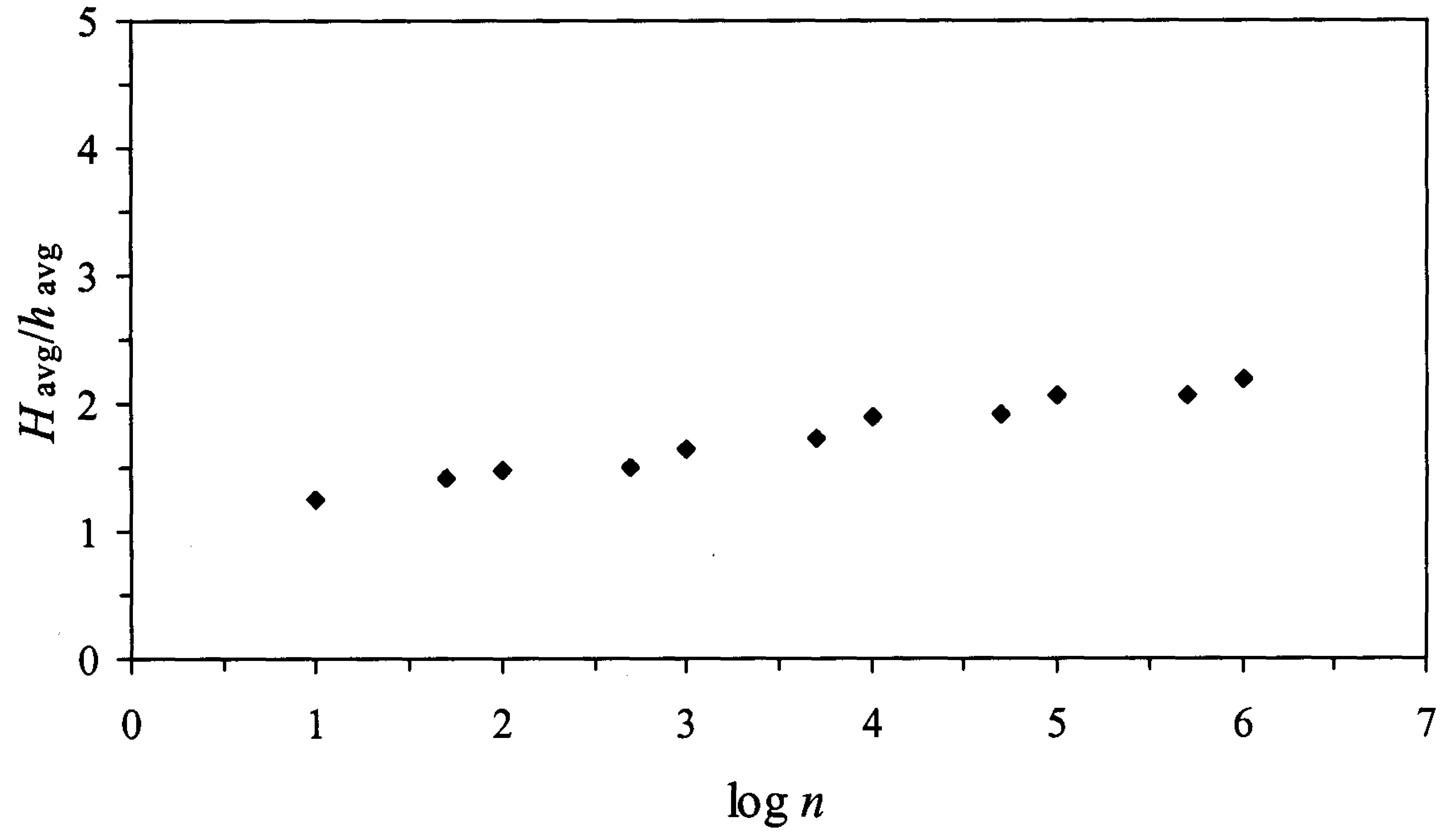


Figure 4.3 The ratios between the sizes of convex hulls and polygons for points distributed in a unit diameter circle.

It was mentioned at the beginning of this chapter that the expected size of the convex hull for points uniformly distributed in a circle is $O(n^{1/3})$. In Figure 4.4, the convex hull sizes and polygon sizes are plotted against the expected sizes of the convex hull. The relationship between the obtained and expected sizes of the convex hulls is represented by straight line which means the order of growth of the obtained convex hull size is the same with the expected one. The R^2 value in the figure is the square of the correlation coefficient. The correlation coefficient, R, gives a measure of the reliability of the linear relationship between the x and y values. A value of R = 1 indicates an exact linear relationship between x and y and values of R close to 1 indicate excellent linear reliability. If the correlation coefficient is relatively far away from 1, the prediction based on the linear relationship is less reliable. In this case the value of the correlation coefficient indicates an excellent linear relationship between the expected and obtained sizes of the convex hull which implies that the obtained sizes of convex hull (*H*) are indeed order of ($n^{1/3}$) for circle distribution of points. In Table 4.3 and Figure 4.3 was shown that the obtained polygon has a lower order of growth than the convex hull. Therefore the expected size of the obtained polygon is less than $O(n^{1/3})$ for circle distribution of points.

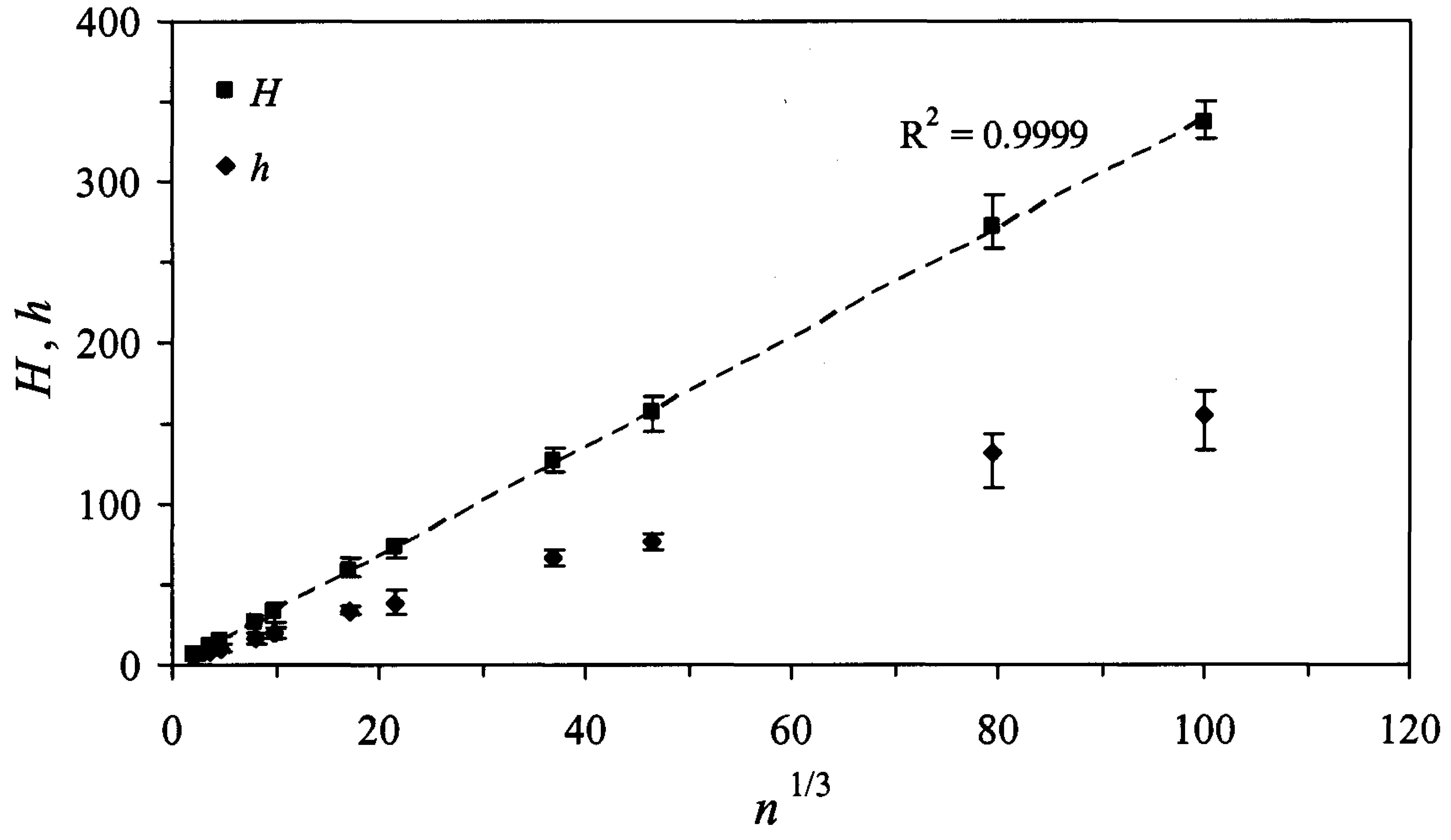


Figure 4.4 The sizes of polygons and convex hulls for points distributed in a unit diameter circle.

4.2.2 Distribution of Points in a Unit Square

The square has the side length equal to 1 unit and it is also centered at the origin of the *x*-*y* axes. The points were generated randomly, uniformly distributed in the square. Therefore the minimum value of *x* and *y* coordinates of the points is -0.5 and maximum is 0.5. As in the circle case, the sets contain [10 50 100 500 1,000 5,000 10,000 50,000 100,000 500,000 1,000,000] points. Four data sets containing [500 1,000 5,000 10,000] points are presented grafically in Figure 4.5. These sets of points were enlarged seven times in the figure.

The results obtained from all the sets of points are presented in Table 4.4. As in the circle distribution shape of points, the first column of the table presents the sets sizes. The second and third columns of the table contain the sizes of the obtained polygons and convex hulls and the fourth column includes their ratios values. The shaded cells in the table represent the minimum and maximum values of the sizes of the polygons and convex hulls for each set size.

Making a comparison with the results obtained from the circle distribution of points, it can be seen that the polygons and the convex hulls in the case of circle for the same number of input points are a lot bigger, especially for the big sets of points where the difference is up to ten times.

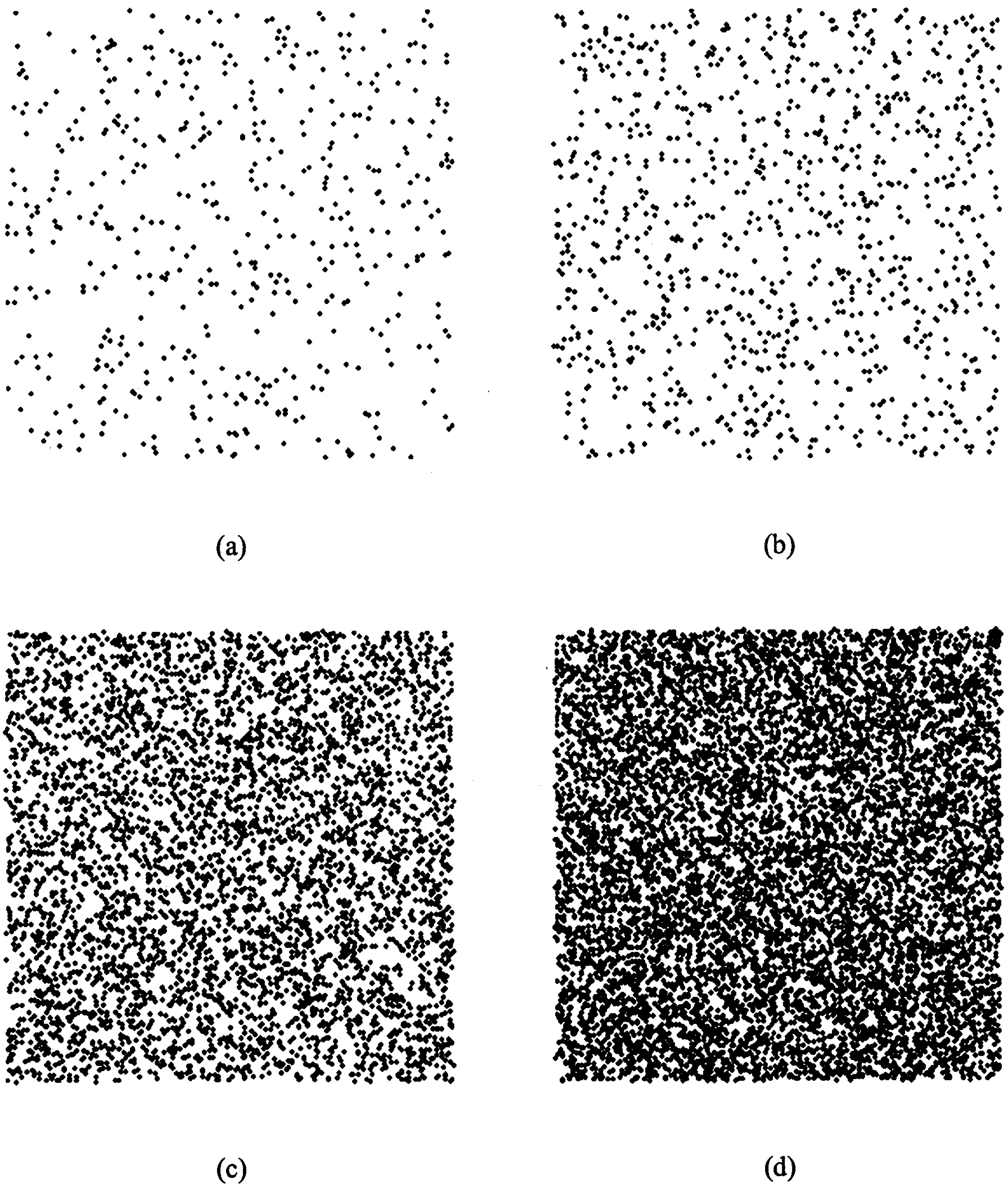


Figure 4.5 Different number of points distributed in unit square (1 unit ≡ 7 cm):

(a) 500 points; (b) 1,000 points; (c) 5,000 points; (d) 10,000 points.

Table 4.4 The results obtained from the processed sets for square distribution.

n	h	H	H/h
10	3	4	1.333
	3	4	1.333
	4	5	1.250
	4	5	1.250
	4	5	1.250
	4	6	1.500
	5	6	1.200
	5	6	1.200
	5	6	1.200
	5	7	1.400
	5	7	1.400
5×10	6	9	1.500
	6	9	1.500
	7	10	1.429
	7	10	1.429
	7	10	1.429
	7	10	1.429
	8	11	1.375
	8	11	1.375
	9	11	1.222
	9	12	1.333
	9	13	1.444
10^2	5	8	1.600
	6	10	1.667
	7	10	1.429
	7	10	1.429
	7	12	1.714
	8	12	1.500
	9	12	1.333
	9	13	1.444
	9	13	1.444
	9	13	1.444
	11	14	1.273
5×10^2	8	12	1.500
	8	16	2.000
	8	16	2.000
	9	16	1.778
	9	17	1.889
	9	18	2.000
	11	18	1.636
	11	18	1.636
	12	18	1.500
	12	18	1.500
	13	19	1.462
10^3	7	13	1.857
	10	15	1.500
	10	16	1.600
	10	17	1.700
	11	18	1.636
	11	18	1.636
	11	18	1.636
	11	19	1.727
	12	20	1.667
	13	20	1.538
	13	22	1.692
5×10^3	11	20	1.818
	11	21	1.909
	11	21	1.909
	11	22	2.000
	11	22	2.000
	12	23	1.917
	12	23	1.917
	13	23	1.769
	13	23	1.769
	14	26	1.857
	14	26	1.857
10^4	12	19	1.583
	12	20	1.667
	12	22	1.833
	12	24	2.000
	13	24	1.846
	15	24	1.600
	15	26	1.733
	15	28	1.867
	15	28	1.867
	16	30	1.875
	17	32	1.882
5×10^4	13	23	1.769
	14	25	1.786
	14	25	1.786
	15	27	1.800
	15	27	1.800
	16	28	1.750
	16	28	1.750
	17	28	1.647
	17	30	1.765
	18	30	1.667
	19	34	1.789
10^5	13	26	2.000
	15	26	1.733
	15	28	1.867
	15	30	2.000
	15	30	2.000
	15	31	2.067
	16	31	1.938
	17	32	1.882
	17	32	1.882
	17	33	1.941
	18	35	1.944
5×10^5	12	23	1.917
	12	27	2.250
	13	31	2.385
	15	32	2.133
	16	33	2.063
	16	33	2.063
	18	34	1.889
	18	37	2.056
	18	37	2.056
	20	39	1.950
	22	41	1.864
10^6	12	30	2.500
	13	33	2.538
	15	35	2.333
	15	36	2.400
	16	37	2.313
	17	39	2.294
	18	40	2.222
	19	40	2.105
	19	43	2.263
	19	44	2.316
	21	44	2.095

In Table 4.5, as in the circle case the sizes of the polygon and convex hull increase with the number of points. Their ratio grows with the number of points which implies the convex hull grows more than the polygon for more points. The size of the polygon is half the size of the convex hull for large sets of points.

Table 4.5 Polygons and convex hulls sizes for points distributed in a unit square.

Square				
n	h_{avg}	H_{avg}	H_{avg} / h_{avg}	$(H / h)_{avg}$
10	4.273	5.545	1.298	1.326
50	7.545	10.545	1.397	1.416
100	7.909	11.545	1.460	1.493
500	10.000	16.909	1.691	1.726
1,000	10.818	17.818	1.647	1.677
5,000	12.091	22.727	1.880	1.895
10,000	14.000	25.182	1.799	1.810
50,000	15.818	27.727	1.753	1.763
100,000	15.727	30.364	1.931	1.940
500,000	16.364	33.364	2.039	2.071
1,000,000	16.727	38.273	2.288	2.327

In Figure 4.6 the ratios between the convex hulls and the polygons were plotted against the number of points from the sets. The convex hull has a higher order of growth than the obtained polygon due to the increase in their ratio with the number of points.

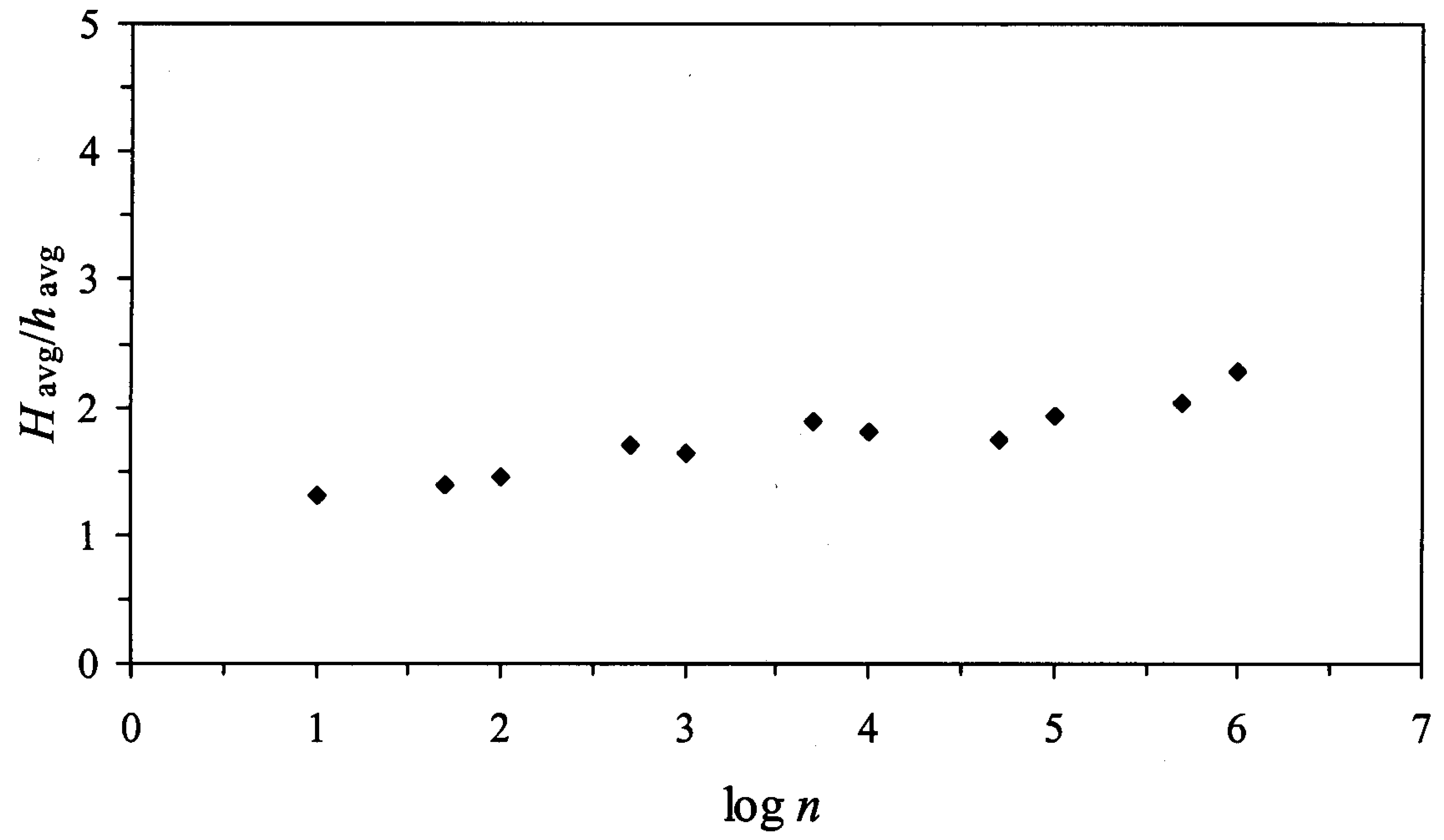


Figure 4.6 The ratios between the sizes of convex hulls and polygons for points distributed in a unit square.

The expected size of the convex hull for points uniformly distributed in a square was shown to be $O(\log n)$. In Figure 4.7, similar to the circle distribution of points, the obtained convex hulls sizes and polygons sizes are plotted against the expected sizes of the convex hulls. The obtained sizes of the convex hulls are order of (log n) shown by the straight line relationship between the obtained and expected sizes of the convex hulls.

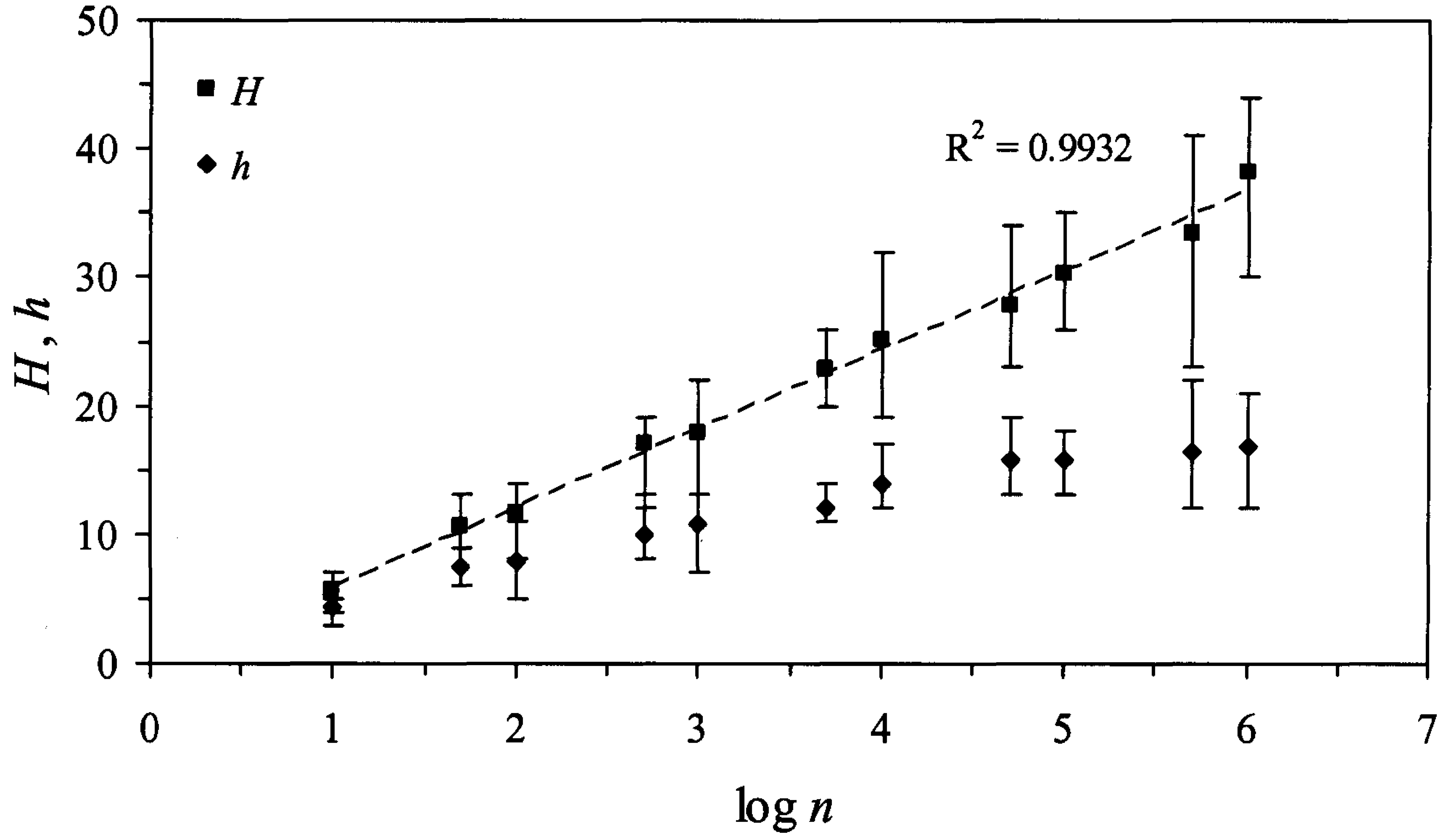


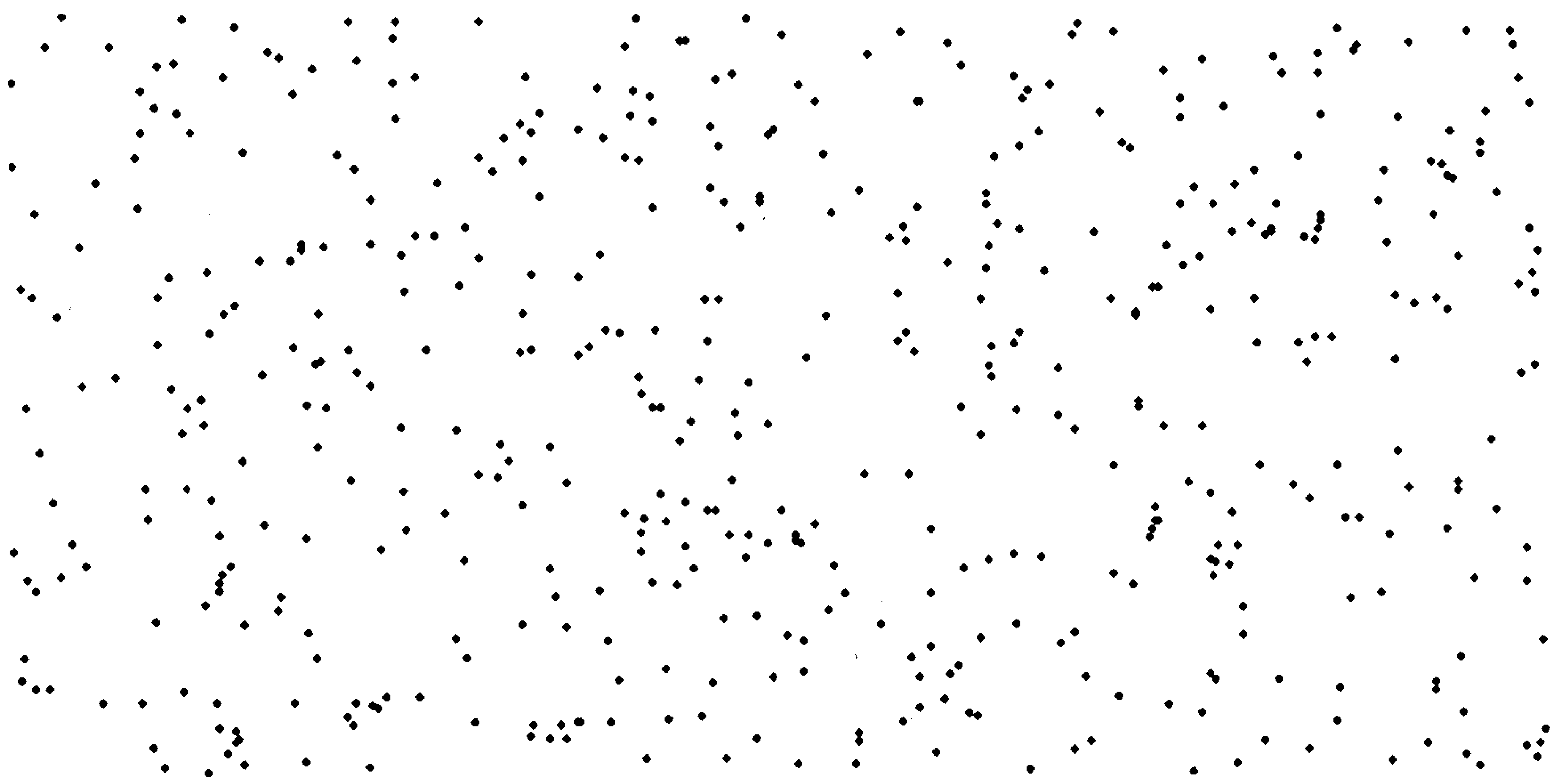
Figure 4.7 The sizes of polygons and convex hulls for points distributed in a unit square.

From Table 4.5 and Figure 4.6, the order of growth of the obtained polygon is less than the order of growth of the convex hull. Therefore the expected size for the polygon is less than $O(\log n)$ for distribution of points in a square.

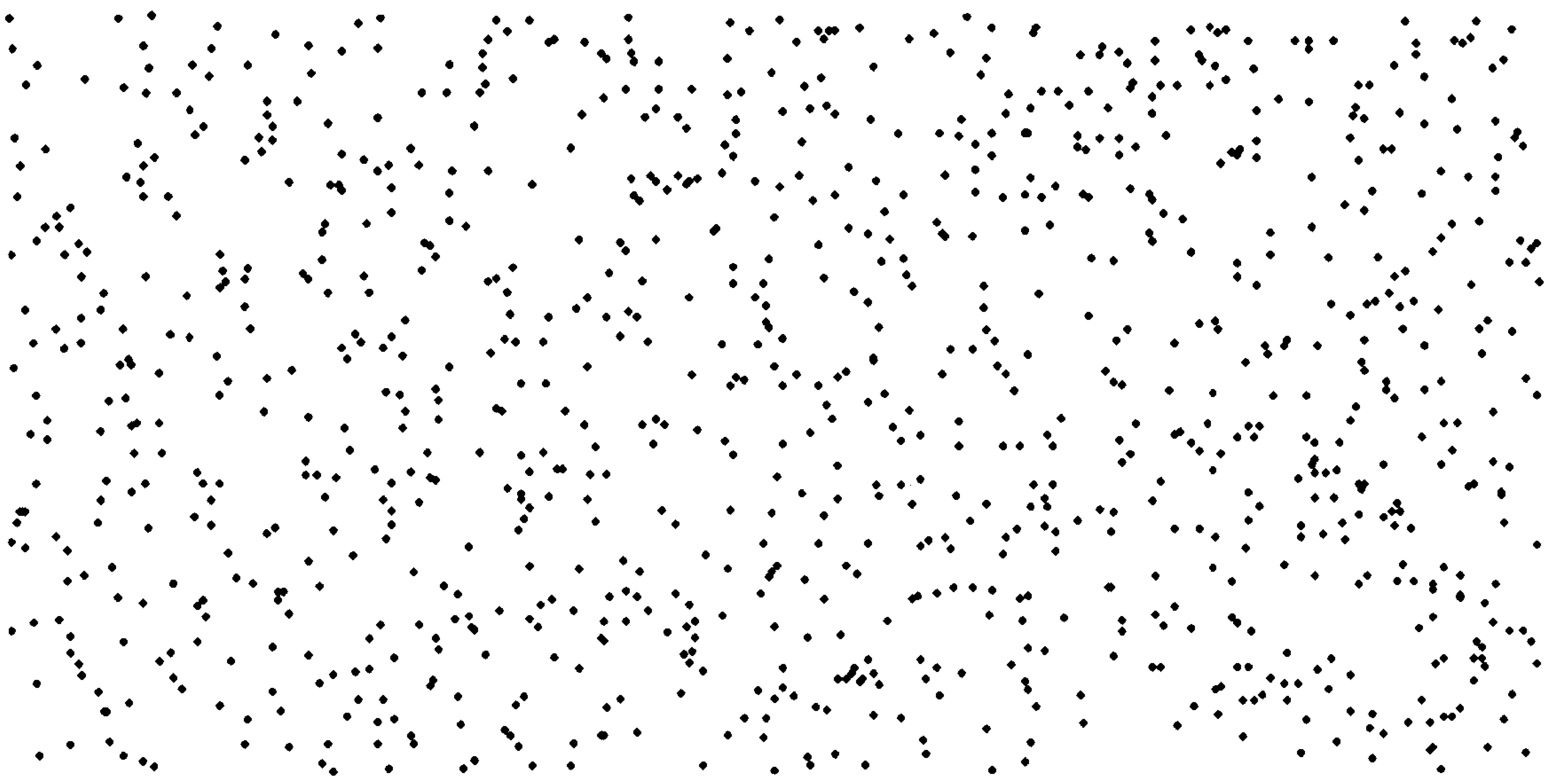
4.2.3 Distribution of Points in a Rectangle 1×2

The rectangle 1×2 has the width equal to 1 unit, the length equal to 2 units and it is also centered at the origin of the x-y axes. The points were generated randomly and uniformly distributed in the rectangle. Therefore the minimum and maximum value of x is

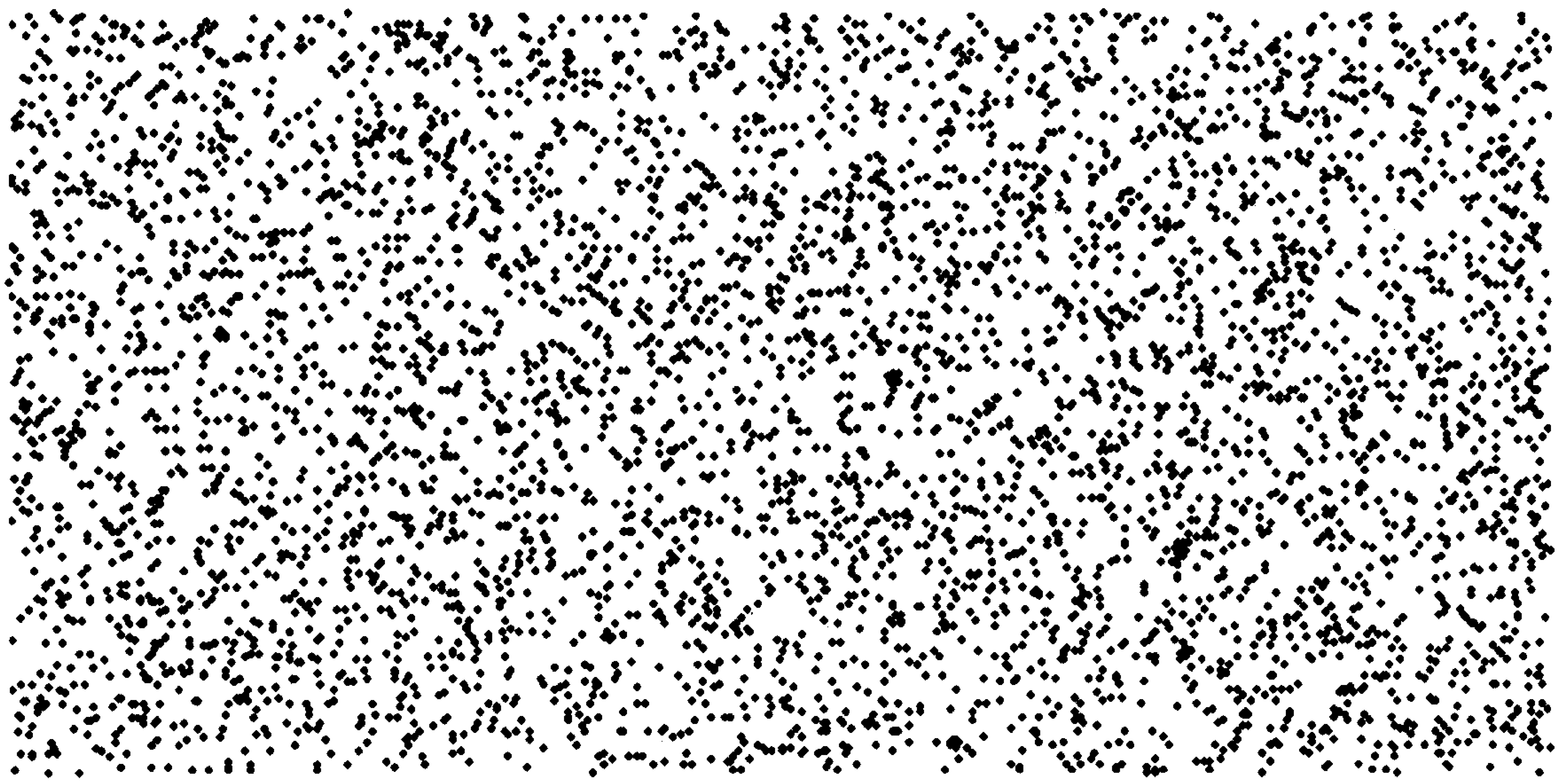
± 1 and the minimum and maximum value of y coordinate is ± 0.5. Figure 4.6 presents exemples of points distributed in the rectangle 1×2, expanded 7.5 times.



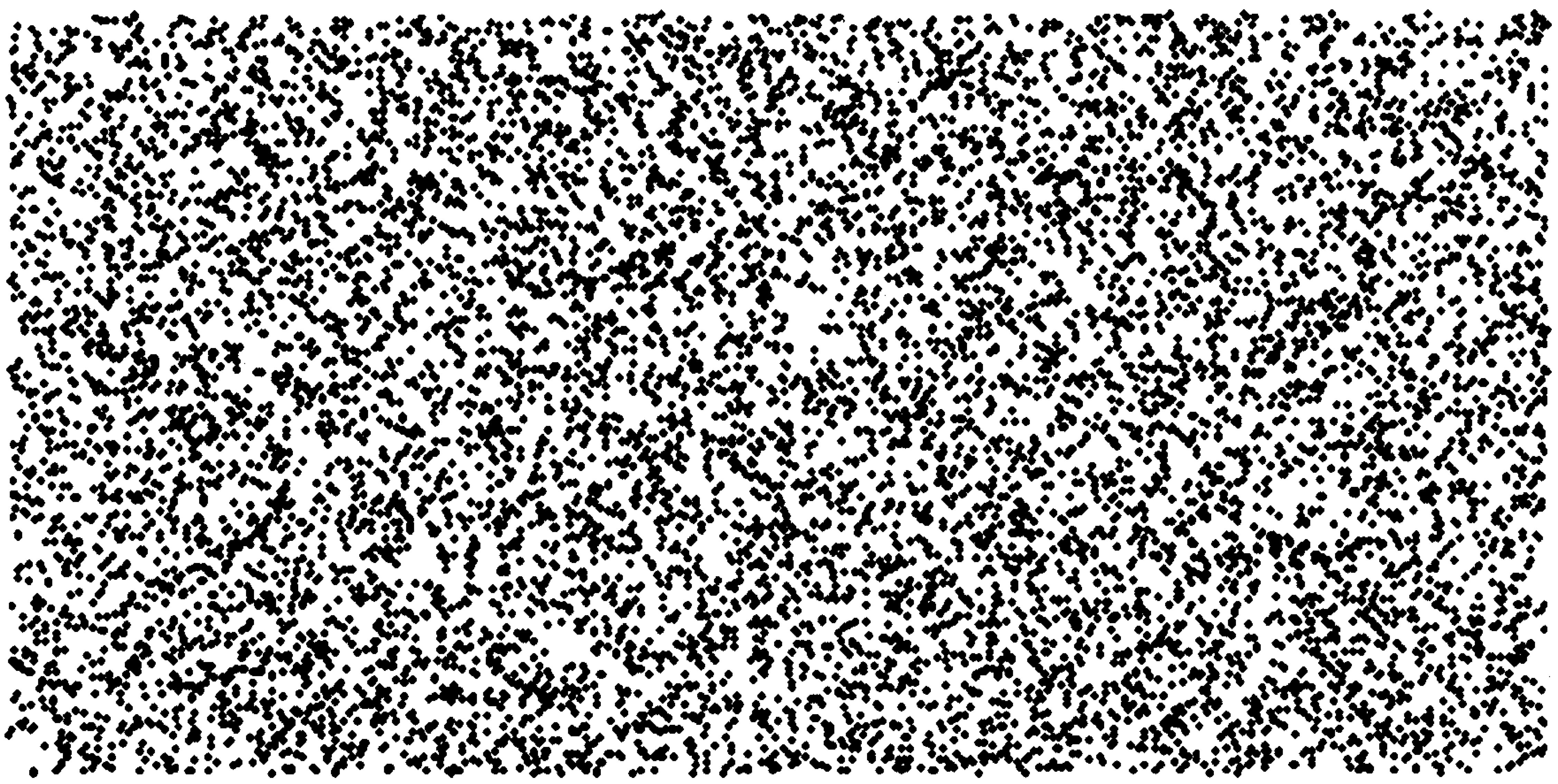
(a)



(b)



(c)



(d)

Figure 4.8 Different number of points distributed in rectangle 1×2 (1 unit ≡ 7.5 cm):

(a) 500 points; (b) 1,000 points; (c) 5,000 points; (d) 10,000 points.

Table 4.6 includes the sizes of the polygons and convex hulls obtained from the processed sets of points and their average sizes. There are no significant differences between the sizes of the convex hulls in this case and in the square distribution of points. A significant difference there exists in the sizes of the obtained polygons in the two shape distributions. In the rectangle 1×2 case the obtained polygon is up to two times smaller than the obtained polygon in the square distribution of points. This is due to the least square line which is a better approximation of the points distributed in rectangle. If the initial solution is better than in the square case, the constructed polygon needed to find the final solution is smaller.

Table 4.6 The results obtained from the processed sets for rectangle 1×2 distribution.

n	h	H	H/h
10	3	4	1.333
	3	5	1.667
	3	6	2.000
	3	6	2.000
	4	6	1.500
	4	6	1.500
	4	6	1.500
	4	6	1.500
	4	6	1.500
	4	7	1.750
	5	7	1.400
5×10	4	7	1.750
	5	9	1.800
	5	10	2.000
	5	10	2.000
	5	11	2.200
	5	11	2.200
	5	12	2.400
	6	13	2.167
	7	13	1.857
	7	13	1.857
	8	14	1.750
10^2	3	9	3.000
	4	9	2.250
	4	9	2.250
	4	10	2.500
	4	11	2.750
	4	11	2.750
	4	11	2.750
	5	12	2.400
	6	13	2.167
	6	14	2.333
	6	15	2.500
5×10^2	4	15	3.750
	4	15	3.750
	5	15	3.000
	5	16	3.200
	6	16	2.667
	6	16	2.667
	7	16	2.286
	7	17	2.429
	7	17	2.429
	8	17	2.125
	8	21	2.625
10^3	3	14	4.667
	4	14	3.500
	5	16	3.200
	5	17	3.400
	6	17	2.833
	6	18	3.000
	6	19	3.167
	8	19	2.375
	8	19	2.375
	8	21	2.625
	9	21	2.333
5×10^3	4	20	5.000
	6	20	3.333
	6	20	3.333
	6	21	3.500
	7	22	3.143
	7	22	3.143
	8	23	2.875
	8	25	3.125
	9	25	2.778
	9	25	2.778
	10	26	2.600
10^4	4	19	4.750
	5	22	4.400
	7	23	3.286
	7	25	3.571
	8	25	3.125
	9	25	2.778
	9	26	2.889
	9	29	3.222
	10	30	3.000
	11	32	2.909
	11	35	3.182
5×10^4	7	24	3.429
	7	25	3.571
	7	26	3.714
	7	26	3.714
	9	27	3.000
	10	29	2.900
	10	30	3.000
	10	30	3.000
	10	30	3.000
	11	31	2.818
	12	34	2.833
10^5	6	26	4.333
	7	26	3.714
	8	28	3.500
	9	29	3.222
	9	29	3.222
	9	31	3.444
	9	32	3.556
	9	32	3.556
	9	33	3.667
	10	34	3.400
	11	34	3.091
5×10^5	7	23	3.286
	7	27	3.857
	8	31	3.875
	8	32	4.000
	8	33	4.125
	9	33	3.667
	10	34	3.400
	11	36	3.273
	11	37	3.364
	11	39	3.545
	11	41	3.727
10^6	6	30	5.000
	6	33	5.500
	7	35	5.000
	7	36	5.143
	9	37	4.111
	9	39	4.333
	9	40	4.444
	10	40	4.000
	10	43	4.300
	11	44	4.000
	12	44	3.667

In Table 4.7 and Figure 4.9, it is shown that the order of growth of the convex hull is larger than the one for the polygon. The size of the polygon is four times smaller than the size of the convex hull for large sets of points

Table 4.7 Polygons and convex hulls sizes for points distributed in a rectangle 1×2.

Rectangle 1×2				
n	h_{avg}	H_{avg}	H_{avg} / h_{avg}	$(H / h)_{avg}$
10	3.727	5.909	1.585	1.632
50	5.636	11.182	1.984	2.040
100	4.545	11.273	2.480	2.594
500	6.091	16.454	2.701	2.858
1,000	6.182	17.727	2.868	3.196
5,000	7.273	22.636	3.112	3.299
10,000	8.182	26.454	3.233	3.420
50,000	9.091	28.364	3.120	3.216
100,000	8.727	30.364	3.479	3.534
500,000	9.182	33.273	3.624	3.685
1,000,000	8.727	38.273	4.385	4.537

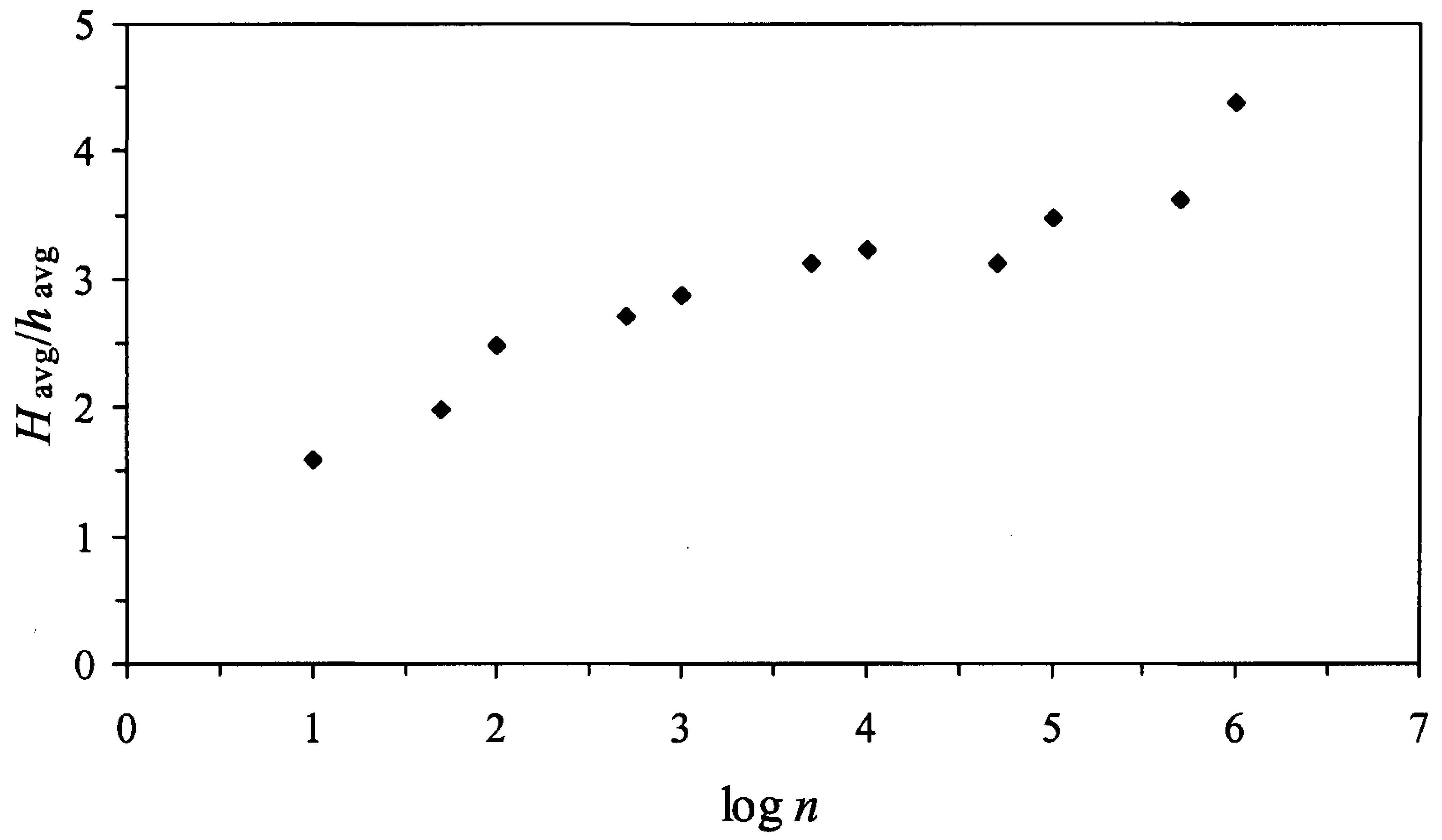


Figure 4.9 The ratios between the sizes of convex hulls and polygons for points distributed in a rectangle 1×2.

In Figure 4.10, similar to the previous shapes, the sizes of the convex hull, and the polygon, are plotted against the expected sizes of the convex hull. The expected size of the convex hull for points uniformly distributed in a rectangle is $O(\log n)$. The obtained size of the convex hull is $O(\log n)$ shown by the value of the correlation coefficient which is very close to 1. The expected size of the polygon is less than $O(\log n)$ having a smaller order of growth than the convex hull.

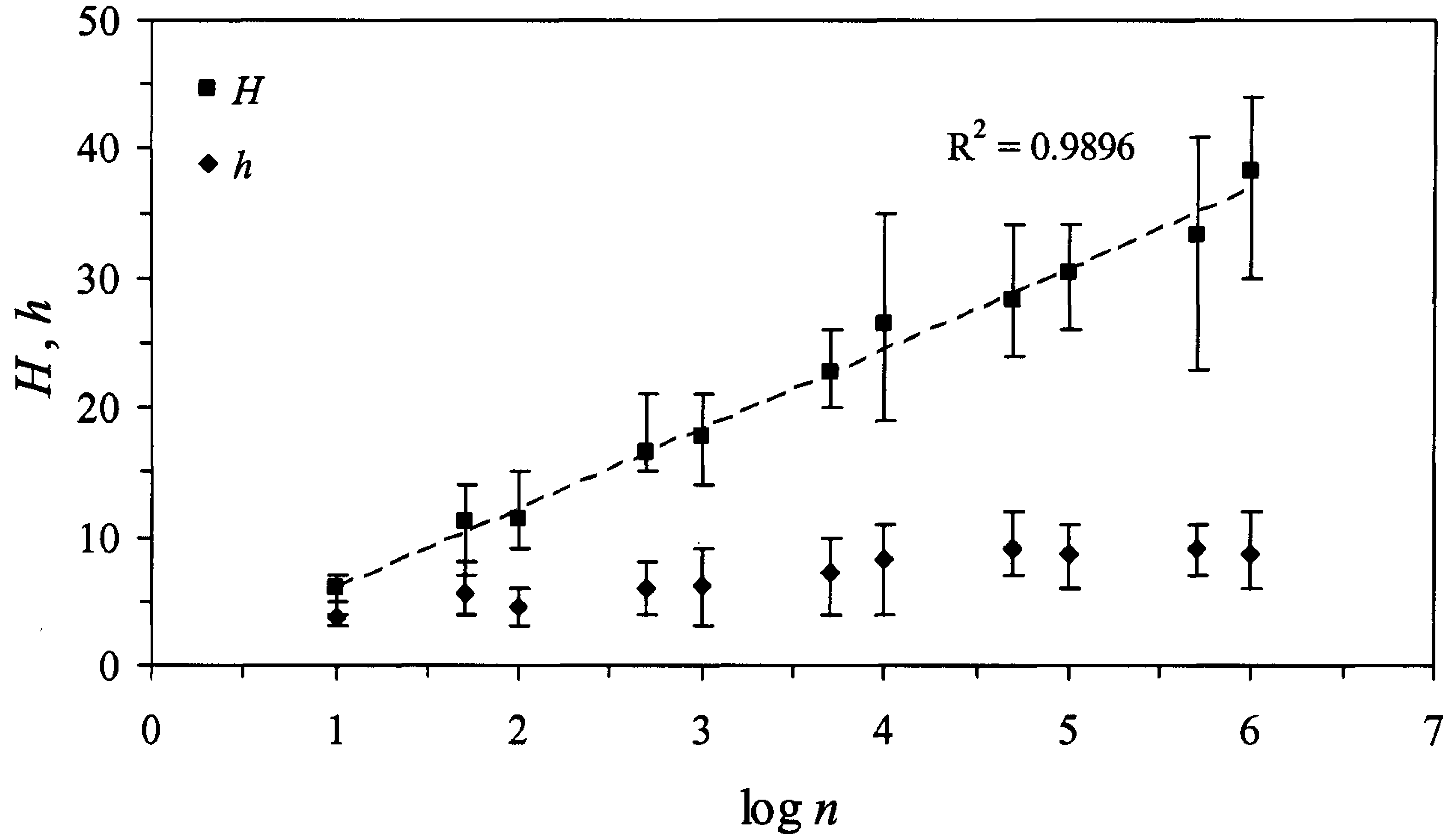


Figure 4.10 The sizes of polygons and convex hulls for points distributed in a rectangle 1×2.

4.2.4 Distribution of Points in a Rectangle 1×5

The rectangle 1×5 is centered at the origin of the *x*-*y* axes and has the width equal to 1 unit and the length equal to 5 units. Therefore the minimum and maximum values of *x* are ± 2.5 and the minimum and maximum value of *y* coordinate are ± 0.5. Figure 4.11 presents points distributed in the rectangle 1×5, enlarged three times for a better visualization. Table 4.8 contains the results for all the processed sets of points.

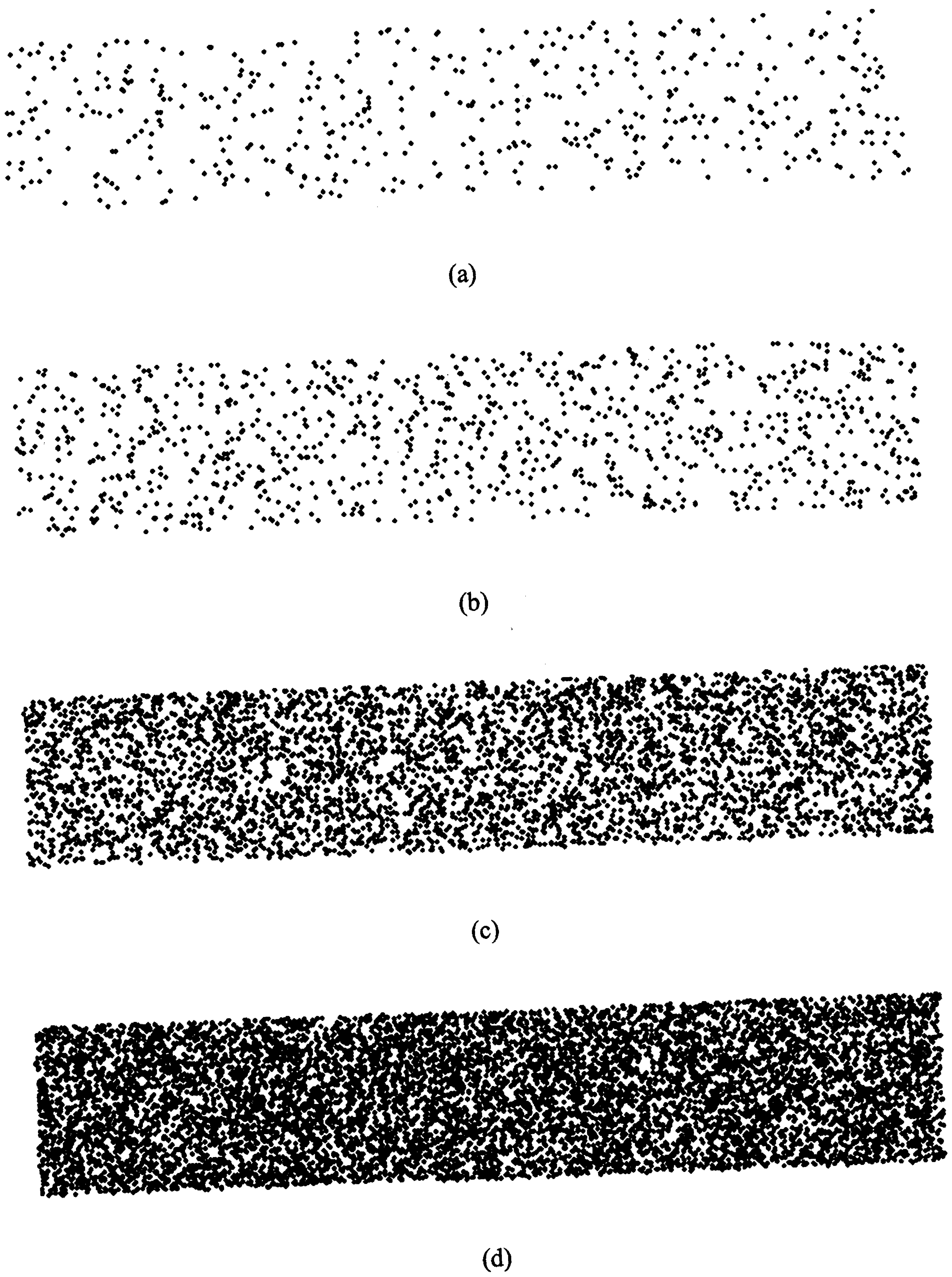


Figure 4.11 Different number of points distributed in rectangle 1×5 (1 unit ≡ 3 cm):

(a) 500 points; (b) 1,000 points; (c) 5,000 points; (d) 10,000 points.

Table 4.8 The results obtained from the processed sets for rectangle 1×5 distribution.

n	h	H	H/h
10	3	4	1.333
	3	6	2.000
	3	6	2.000
	3	6	2.000
	4	6	1.500
	4	6	1.500
	5	7	1.400
	5	7	1.400
	5	7	1.400
	5	7	1.400
	6	8	1.333
5×10	3	7	2.333
	3	9	3.000
	3	9	3.000
	3	9	3.000
	4	9	2.250
	5	9	1.800
	5	10	2.000
	5	10	2.000
	6	12	2.000
	6	12	2.000
	8	14	1.750
10^2	3	7	2.333
	3	10	3.333
	3	10	3.333
	3	11	3.667
	4	11	2.750
	5	11	2.200
	5	12	2.400
	5	13	2.600
	5	14	2.800
	5	15	3.000
	6	16	2.667
5×10^2	3	13	4.333
	5	16	3.200
	5	16	3.200
	5	16	3.200
	5	16	3.200
	6	17	2.833
	6	18	3.000
	7	18	2.571
	7	18	2.571
	8	19	2.375
	10	19	1.900
10^3	3	17	5.667
	5	19	3.800
	5	19	3.800
	6	19	3.167
	6	20	3.333
	6	20	3.333
	6	20	3.333
	7	21	3.000
	7	22	3.143
	7	23	3.286
	8	24	3.000
5×10^3	3	17	5.667
	5	18	3.600
	6	19	3.167
	6	19	3.167
	7	21	3.000
	8	22	2.750
	8	22	2.750
	8	23	2.875
	9	23	2.556
	9	23	2.556
	10	25	2.500
10^4	5	19	3.800
	5	20	4.000
	6	21	3.500
	6	22	3.667
	7	22	3.143
	7	23	3.286
	8	24	3.000
	8	24	3.000
	9	25	2.778
	9	27	3.000
	9	27	3.000
5×10^4	5	26	5.200
	6	26	4.333
	6	27	4.500
	6	28	4.667
	7	29	4.143
	7	29	4.143
	7	29	4.143
	8	30	3.750
	8	31	3.875
	10	31	3.100
	10	32	3.200
10^5	5	23	4.600
	5	26	5.200
	6	27	4.500
	6	29	4.833
	7	30	4.286
	8	31	3.875
	8	31	3.875
	8	31	3.875
	9	34	3.778
	10	34	3.400
	10	36	3.600
5×10^5	6	29	4.833
	8	33	4.125
	9	34	3.778
	10	34	3.400
	10	34	3.400
	10	34	3.400
	10	36	3.600
	11	37	3.364
	11	38	3.455
	11	38	3.455
	11	39	3.545
10^6	6	30	5.000
	6	30	5.000
	8	31	3.875
	9	32	3.556
	9	35	3.889
	9	37	4.111
	10	37	3.700
	10	39	3.900
	10	39	3.900
	11	42	3.818
	13	43	3.308

From Table 4.9 the size of the polygon is four times smaller than the size of the convex hull for many points. As in the previous distribution shapes, the expected size of the obtained polygon is smaller than the expected size of the convex hull. Table 4.9 and Figure 4.12 show that their ratio increases.

Table 4.9 Polygons and convex hulls sizes for points distributed in a rectangle 1×5.

Rectangle 1×5				
n	h_{avg}	H_{avg}	$H_{\text{avg}} / h_{\text{avg}}$	$(H / h)_{\text{avg}}$
10	4.182	6.364	1.522	1.606
50	4.636	10.000	2.157	2.453
100	4.273	11.818	2.766	2.902
500	6.091	16.909	2.776	3.022
1,000	6.000	20.364	3.394	3.574
5,000	7.182	21.091	2.937	3.274
10,000	7.182	23.091	3.215	3.335
50,000	7.273	28.909	3.975	4.134
100,000	7.454	30.182	4.049	4.308
500,000	9.727	35.091	3.607	3.760
1,000,000	9.182	35.909	3.911	4.054

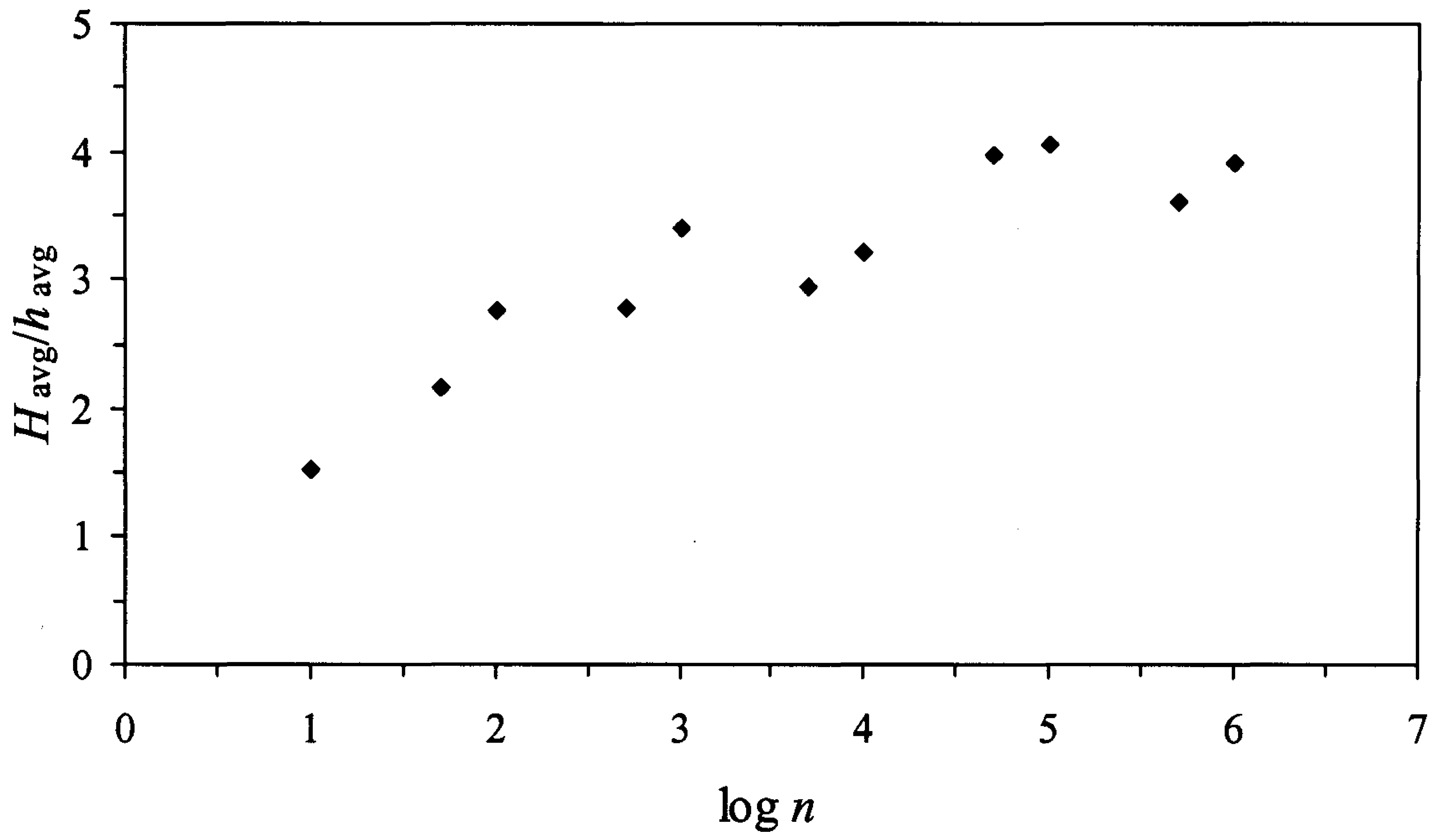


Figure 4.12 The ratios between the sizes of convex hulls and polygons for points distributed in a rectangle 1×5.

The expected size of the convex hull for points uniformly distributed in a rectangle is $O(\log n)$. Figure 4.13 shows the obtained size of the convex hull is $O(\log n)$ and therefore the expected size of the polygon is less than $O(\log n)$.

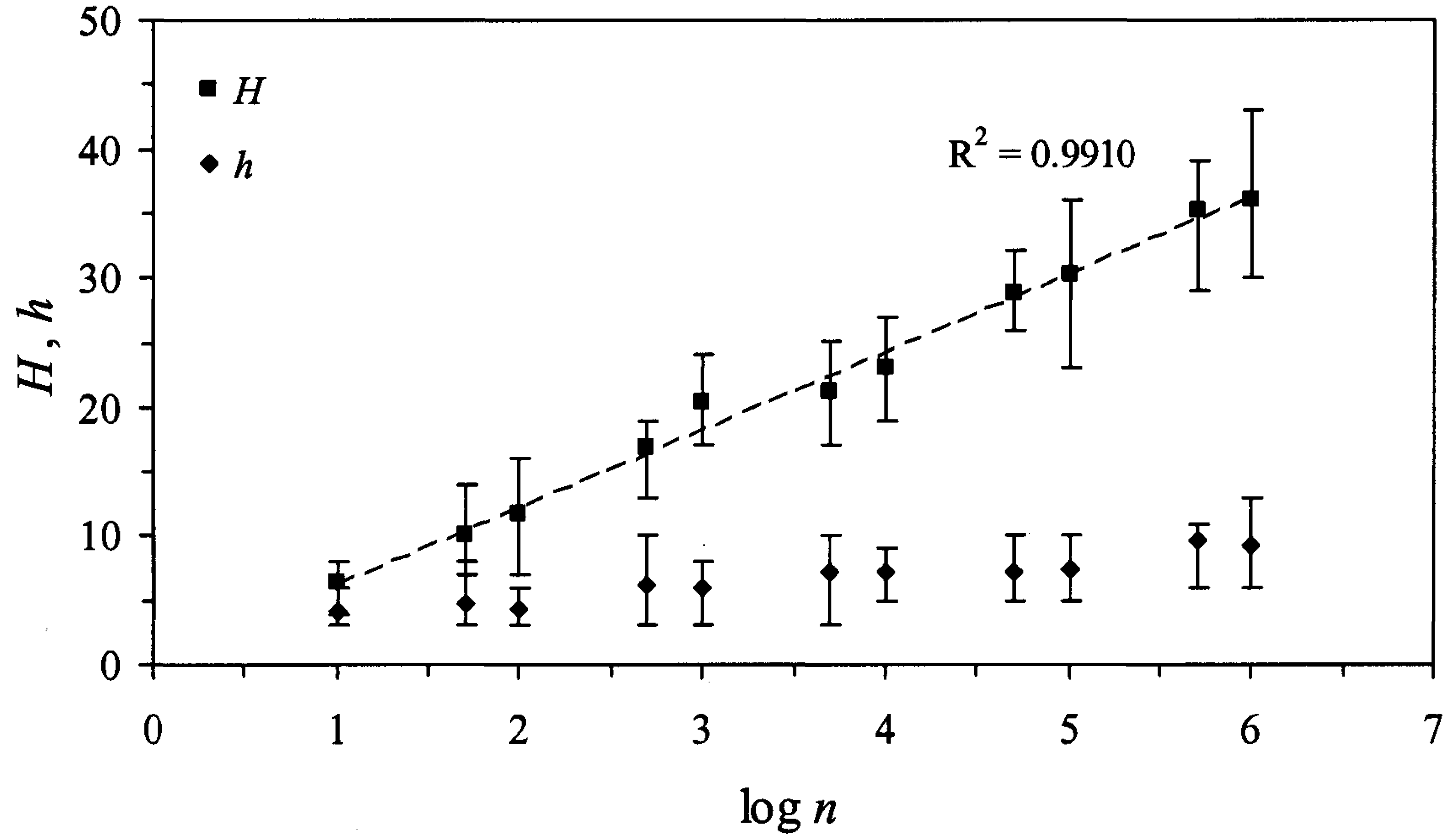
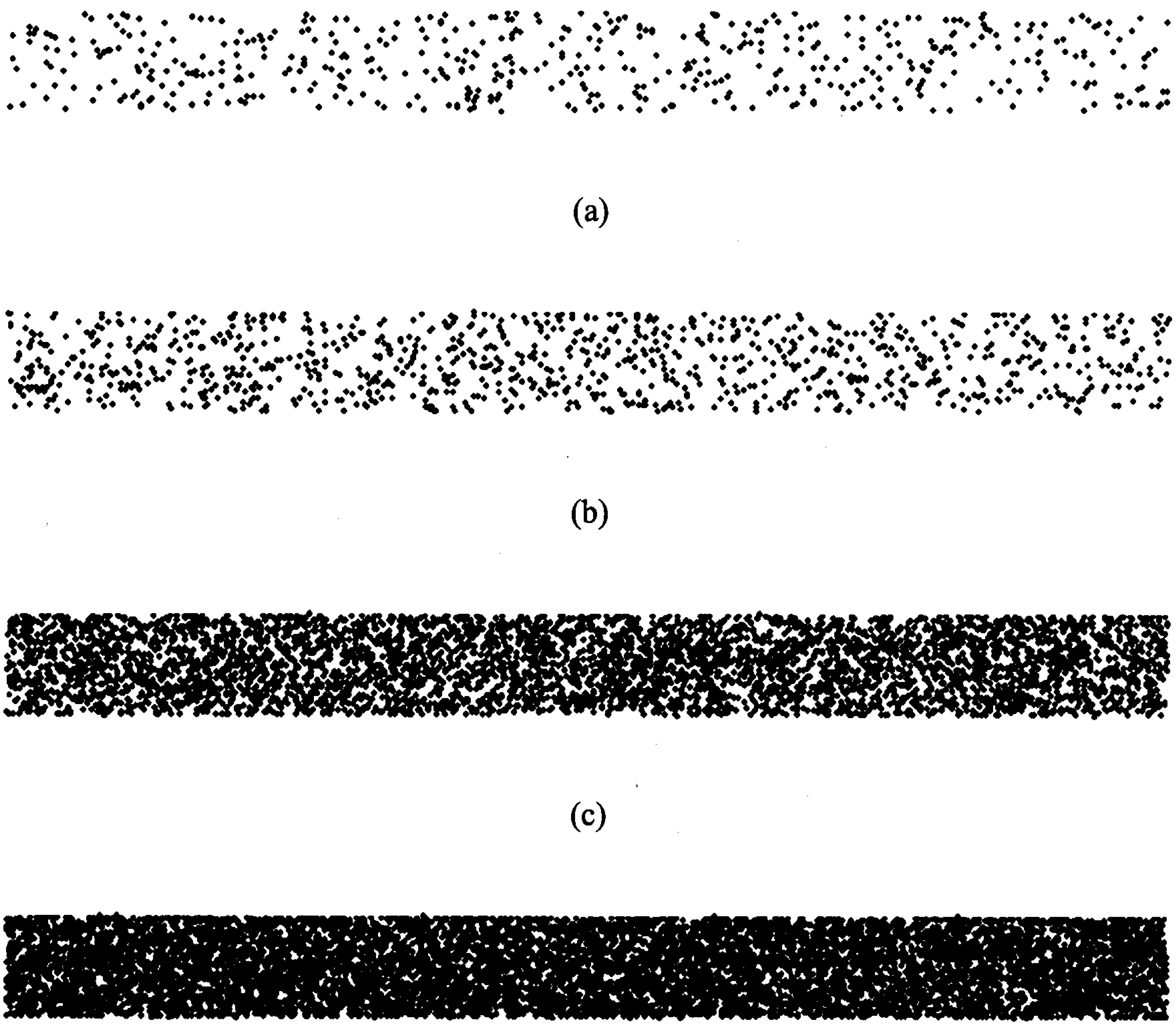


Figure 4.13 The sizes of polygons and convex hulls for points distributed in a rectangle 1×5.

4.2.5 Distribution of Points in a Rectangle 1×10

The rectangle 1×10 centered at the origin of the coordinate axes has the width equal to 1 unit and the length equal to 10 units. The minimum and maximum values of x are ± 5 and the minimum and maximum values of y coordinate are ± 0.5. Figure 4.14 illustrates points distributed in the rectangle 1×10. The distribution shape is enlarged 1.5 times in the figure.



(a)

(b)

(c)

(d)

Figure 4.14 Different number of points distributed in rectangle 1×10 (1 unit ≡ 1.5 cm): (a) 500 points; (b) 1,000 points; (c) 5,000 points; (d) 10,000 points.

Table 4.10 includes the obtained values of the sizes of the polygons and the convex hulls and the ratio between them for each set of points. As in the previous distribution shapes the shaded cells contain the minimum and maximum sizes of the polygons and convex hulls from each set size.

Table 4.10 The results obtained from the processed sets for rectangle 1×10 distribution.

n	h	H	H/h
10	3	4	1.333
	3	4	1.333
	3	5	1.667
	3	5	1.667
	3	6	2.000
	3	6	2.000
	3	6	2.000
	3	6	2.000
	4	6	1.500
	4	7	1.750
	5	8	1.600
5×10	3	8	2.667
	4	8	2.000
	4	9	2.250
	4	10	2.500
	4	10	2.500
	5	10	2.000
	5	10	2.000
	5	10	2.000
	5	11	2.200
	6	11	1.833
	6	11	1.833
10^2	3	9	3.000
	4	10	2.500
	5	10	2.000
	5	11	2.200
	5	12	2.400
	5	12	2.400
	6	12	2.000
	6	13	2.167
	6	14	2.333
	7	14	2.000
	9	14	1.556
$5×10^2$	3	13	4.333
	3	13	4.333
	3	14	4.667
	5	14	2.800
	5	14	2.800
	5	14	2.800
	5	16	3.200
	5	16	3.200
	7	17	2.429
	8	18	2.250
	8	19	2.375
10^3	3	17	5.667
	3	18	6.000
	4	18	4.500
	5	18	3.600
	6	19	3.167
	6	19	3.167
	7	19	2.714
	8	20	2.500
	9	20	2.222
	10	21	2.100
	10	24	2.400
$5×10^3$	3	17	5.667
	6	18	3.000
	6	20	3.333
	6	20	3.333
	6	20	3.333
	7	21	3.000
	8	22	2.750
	8	22	2.750
	8	22	2.750
	8	23	2.875
	9	24	2.667
10^4	3	21	7.000
	5	21	4.200
	6	22	3.667
	7	22	3.143
	7	23	3.286
	7	23	3.286
	7	24	3.429
	8	25	3.125
	9	25	2.778
	10	27	2.700
	12	28	2.333
$5×10^4$	6	22	3.667
	7	24	3.429
	8	25	3.125
	8	26	3.250
	8	26	3.250
	9	27	3.000
	9	28	3.111
	9	29	3.222
	10	31	3.100
	10	33	3.300
	11	34	3.091
10^5	6	26	4.333
	8	28	3.500
	8	30	3.750
	8	30	3.750
	8	31	3.875
	8	33	4.125
	9	33	3.667
	9	34	3.778
	9	34	3.778
	10	35	3.500
	11	37	3.364
$5×10^5$	3	27	9.000
	6	32	5.333
	6	33	5.500
	7	33	4.714
	7	35	5.000
	8	35	4.375
	9	36	4.000
	9	36	4.000
	9	37	4.111
	11	37	3.364
	11	41	3.727
10^6	7	31	4.429
	8	32	4.000
	8	33	4.125
	8	35	4.375
	9	35	3.889
	9	37	4.111
	9	38	4.222
	10	38	3.800
	11	39	3.545
	11	39	3.545
	13	46	3.538

In Table 4.11, as in the other two rectangles distributions of points, the size of the obtained polygon is four times smaller than the size of the convex hull for more points. In this table as well as in Figure 4.15 it can be observed that the expected size of the polygon has a less order of growth than the one of the convex hull.

Table 4.11 Polygons and convex hulls sizes for points distributed in a rectangle 1×10.

Rectangle 1×10				
n	h_{avg}	H_{avg}	H_{avg} / h_{avg}	$(H / h)_{avg}$
10	3.364	5.727	1.703	1.753
50	4.636	9.818	2.118	2.218
100	5.545	11.909	2.147	2.264
500	5.182	15.273	2.947	3.313
1,000	6.454	19.364	3.000	3.508
5,000	6.818	20.818	3.053	3.343
10,000	7.364	23.727	3.222	3.575
50,000	8.636	27.727	3.210	3.274
100,000	8.545	31.909	3.734	3.799
500,000	7.818	34.727	4.442	5.011
1,000,000	9.364	36.636	3.913	3.998

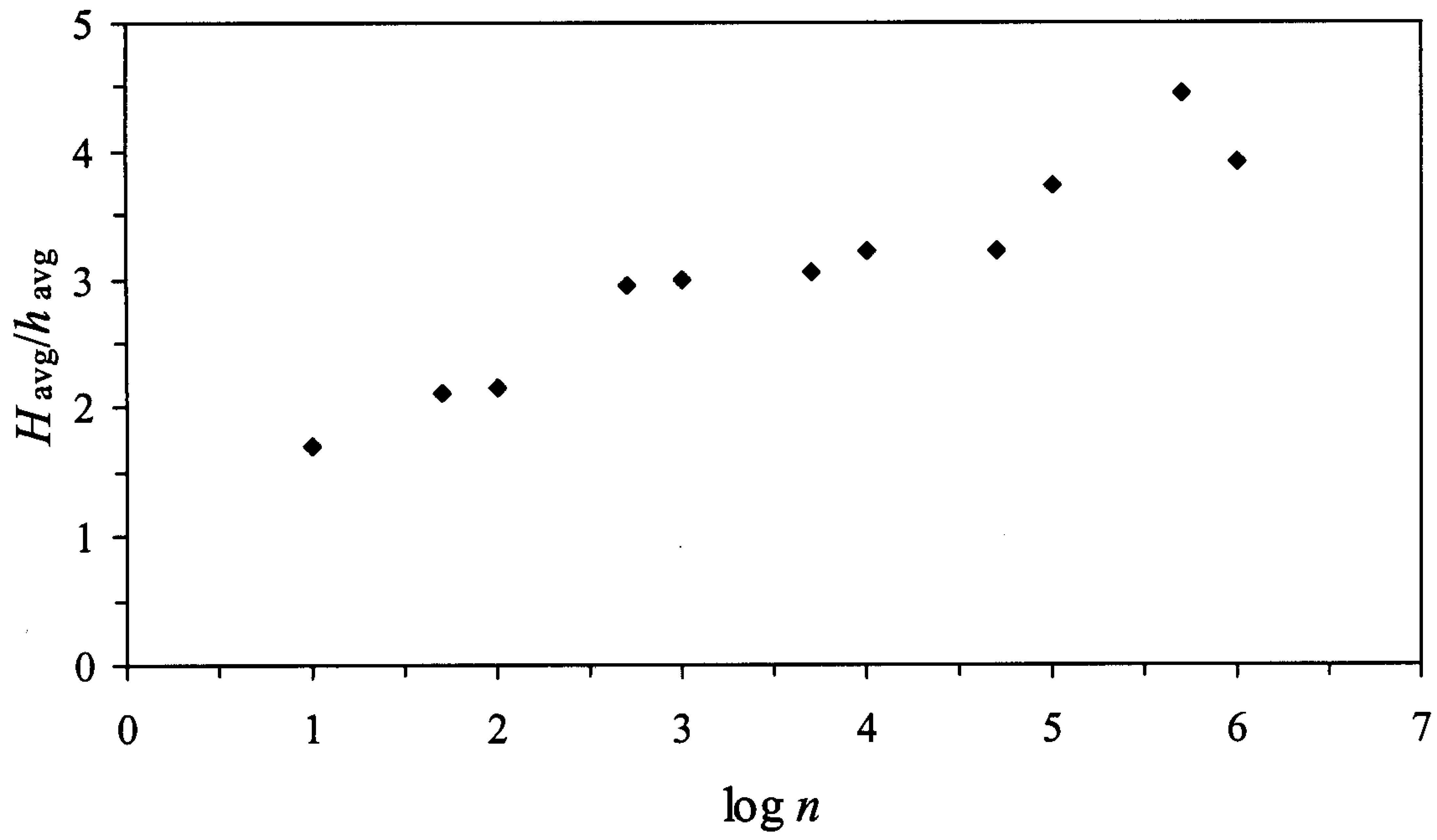


Figure 4.15 The ratios between the sizes of convex hulls and polygons for points distributed in a rectangle 1×10.

The expected size of the convex hull is $O(\log n)$. The obtained sizes of the convex hull are also $O(\log n)$ as shown in Figure 4.16 and the obtained sizes of the polygon are less than $O(\log n)$.

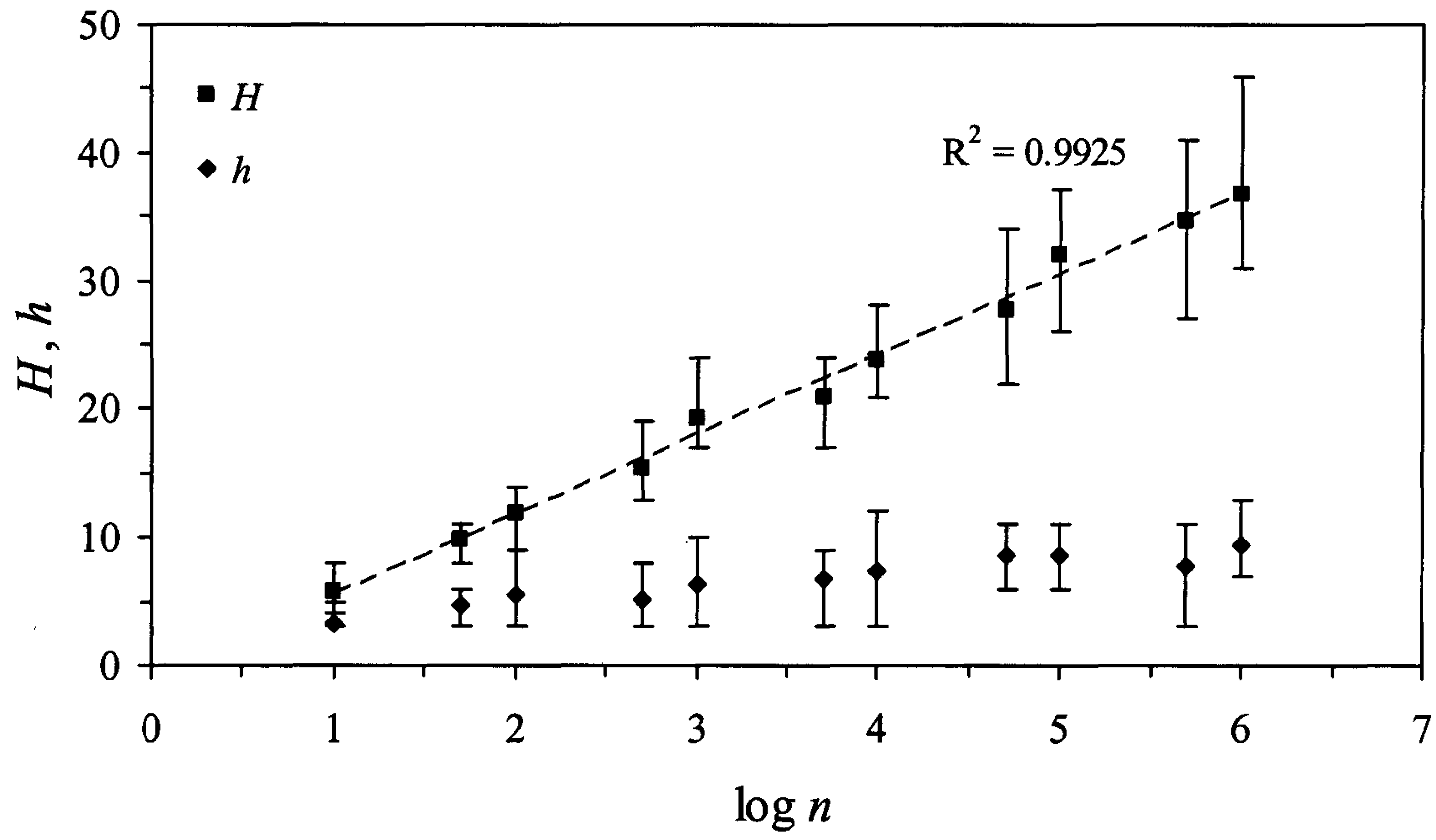


Figure 4.16 The sizes of polygons and convex hulls for points distributed in a rectangle 1×10.

Table 4.12 includes the ratios between the convex hulls sizes and the polygons sizes for each distribution shape. It can be seen that the ratio between the sizes of the polygons increases with the number of points. The shaded cells contain the minimum and maximum values of the ratios between the sizes of the polygons and the convex hulls for each distribution of points. In the circle and square distributions of points, the polygon size is half the convex hull size for large number of points. In the rectangles distributions of points, the polygon size is up to four times smaller than the convex hull size. This difference is due to the least squares line and shape distribution of the points. The least squares line can have any direction for points distributed in a circle and it is not a good representation of the points. For rectangles distributions of points, the least square line is a better approximation of the points and implicitly defines a better initial solution.

Table 4.12 Comparison of the sizes of polygons and convex hulls.

H_{avg} / h_{avg}					
n	Circle	Square	Rectangle 1×2	Rectangle 1×5	Rectangle 1×10
10	1.245	1.298	1.585	1.522	1.703
50	1.422	1.397	1.984	2.157	2.118
100	1.474	1.460	2.480	2.766	2.147
500	1.505	1.691	2.701	2.776	2.947
1,000	1.649	1.647	2.868	3.394	3.000
5,000	1.739	1.880	3.112	2.937	3.053
10,000	1.893	1.799	3.233	3.215	3.222
50,000	1.910	1.753	3.120	3.975	3.210
100,000	2.056	1.931	3.479	4.049	3.734
500,000	2.063	2.039	3.624	3.607	4.442
1,000,000	2.178	2.288	4.385	3.911	3.913

Between rectangles distributions of points there is no significant change in the ratios of the polygons and convex hulls. The least squares line has almost the same behavior even if the rectangle is shorter or longer. Both the polygon and the convex hull grow with the number of points but the growth is not proportional. The polygon size in rectangles cases, for small sets for example, is 1.5 times smaller than the convex hull size while for the big sets is 4 times smaller. The changes in the ratios of sizes show the polygon has a lower order of growth than the convex hull.

The convex hulls sizes and polygons sizes were plotted against the expected sizes of the convex hulls for all the distribution shapes of points. The obtained convex hulls sizes (H) are indeed order of ($n^{1/3}$) for circle and order of ($\log n$) for polygon distributions of points. The relationship between the obtained and expected sizes of the convex hull is represented by straight lines, which means the obtained convex hull size has the same order as the expected one. The ratios between the obtained polygon sizes and the convex hull sizes increase with the number of points which implies the order of growth of the polygon is less than the one of the convex hull. Therefore the expected size of the polygon is less than $O(n^{1/3})$ for uniform distribution of points in a circle and less than $O(\log n)$ for uniform distribution of points in a convex polygon. The present algorithm is an output sensitive algorithm which means that depends on the size of the output. The output in this case is the size of the polygon or the number of points on the polygon (h). If the size of the expected polygon is less than $O(n^{1/3})$ for circle distribution of points, the expected computational complexity for the proposed algorithm is less than $O(n^{4/3})$ for the same distribution of points. The expected size of the polygon is less than $O(\log n)$ for polygons distributions of points and therefore the expected complexity of the present algorithm is less than $O(n \log n)$.

CHAPTER 5

CONCLUSIONS

A fast and accurate method for minimum zone straightness evaluation was introduced in this thesis. The proposed method is a computational geometric technique and is based on minimum zone for the smallest subset and the polygon expansion principles. This method guarantees the minimum zone straightness for any set of points. It is not dependent on the number of points, the type of point's distribution (uniform, non-uniform) or the geometry (circle, square, rectangle).

The algorithm starts by finding an initial solution. In the next step the algorithm verifies if the initial solution is the global solution based on the minimum zone for the three points which define the initial solution. The algorithm stops if the initial solution for the set is the minimum zone for the initial three points and guarantees that it is also the minimum zone for the set. If the initial solution does not coincide with the minimum zone for the three points, the algorithm expands the subset of vertex points which define a convex polygon. New points are added one by one at the starting three points based on the polygon expansion principle. This principle consists of finding the furthest points outward or inward from that edge of the polygon, which defines one of the minimum zone lines for the polygon. The polygon expansion based on the furthest points from the minimum edge ensures the polygon is a convex polygon and all its vertices are also vertices of the convex hull for the set of points. At each addition of new points, the algorithm checks if the initial solution coincides with the minimum zone for that subset. If a subset satisfying this condition is found, the algorithm stops and guarantees the initial solution is the minimum zone for the set. The algorithm also stops when the minimum zone for the subset coincides with a new feasible solution for the set. The new feasible solution for the set is guaranteed to be the minimum zone for the set. In this case, the initial solution is found to be only a feasible solution for the set. The algorithm converges when the feasible solution for the set of points is the minimum zone for a subset or when a minimum zone for a subset of points is a feasible solution for the set.

The present algorithm is an output sensitive algorithm. Its expected computational complexity depends on the size of the subset of points or the size of the constructed polygon. The polygon stops growing when of the minimum zone conditions discussed previously is satisfied. The polygon is part of the convex hull for the set of points. Only in the worst case the polygon is expanded to the maximum and coincides with the convex hull. Therefore, the expected size of the polygon is smaller than the expected size of the convex hull. The expected size of the convex hull was shown to be $O(\log n)$ for points uniformly distributed in a polygon and $O(n^{1/3})$ for points uniformly distributed in a circle. Thus, the expected size of the polygon is less than $O(\log n)$ for points uniformly distributed in a convex polygon and less than $O(n^{1/3})$ for points uniformly distributed in a circle. From simulated data, in Chapter 4 it is observed that the polygon generated from the present method is up to two times smaller than the convex hull for points uniformly distributed in a circle and a square and up to four times smaller for points uniformly distributed in a rectangle.

Because the expected computational complexity of the algorithm depends on the size of the polygon, the algorithm has a complexity less than $O(n \log n)$ for points uniformly distributed in a convex polygon and less than $O(n^{4/3})$ for points uniformly distributed in a circle. The existing methods based on the convex hull method have a $O(n \log n)$ computational complexity. If the initial solution is the best solution the algorithm presents even a better complexity, which is $O(n)$ for the best case. Therefore the present method is not only accurate but also efficient. The accuracy and efficiency of the present method were confirmed using data sets reported in the literature and simulated data.

An interesting direction for future research can be to find the best initial solution. The polygon expansion depends on the initial solution. The expansion can be a big expansion in the worst case, a moderate one or, in the best case, no expansion at all. The least squares line is not a good representation of the points all the time. Implicitly the initial solution obtained from the least squares line is not a very good initial solution. Having the best initial solution ensures that the found subset is the smallest possible subset from that set of points. The smallest size of the polygon from all the possible polygons also reduces the expected computational complexity of the algorithm to or close to $O(n)$. Due to the similarities between straightness and flatness, another direction for future research can be the expansion of the present method to flatness.

REFERENCES

[1] ASME Y14.5M, 1994, Dimensioning and tolerancing, *The American Society of Mechanical Engineers*, New York.

[2] Lin, S. S., Varghese, P., Zhang, C. and Wang H. –P. B., 1995, "A Comparative Analysis of CMM Form-Fitting Algorithms," *Manufacturing Review*, Vol. 8, No. 1, pp. 47-58.

[3] Chetwynd, D. G., 1985, "Applications of Linear Programming to Engineering Metrology," *Proceedings of the Institute of Mechanical Engineers*, Vol. 199, No. B2, pp. 93-100.

[4] Traband, M. T., Joshi, S., Wysk, R. A. and Cavalier, T. M., 1989, "Evaluation of Straightness and Flatness Tolerances Using the Minimum Zone," ASME *Manufacturing Review*, Vol. 2, No. 3, pp. 189-195.

[5] Cormen, T. H., Leiserson, C. E., Rivest, R. L. and Stein, C., 2001, "Introduction to Algorithms," *The MIT Press*, Cambridge, Massachusetts London, England.

[6] Samuel, G. L. and Shunmugam, M. S., 1999, "Evaluation of Straightness and Flatness Error Using Computational Geometric Techniques," *Computer Aided Design*, Vol. 31, No. 13, pp. 829-843.

[7] Chen, M. C. and Fan, S. K. S., 2002, "Tolerance Evaluation of Minimum Zone Straightness Using Non-Linear Programming Techniques: A Spreadsheet Approach," *Computers and Industrial Engineering*, Vol. 43, No. 3, pp. 437-453.

[8] Cheraghi, S. H., Lim, H. S. and Motavalli, S., 1996, "Straightness and Flatness Tolerance Evaluation: An Optimization Approach," *Precision Engineering*, Vol. 18, No. 1, pp. 30-37.

[9] Wang, Y., 1992, "Minimum Zone Evaluation of Form Tolerances," *Manufacturing Review*, Vol. 5, No. 3, pp. 213-220.

[10] Carr, K. and Ferreira, P., 1995, "Verification of Form Tolerances. Part I: Basic Issues, Flatness, and Straightness," *Precision Engineering*, Vol. 17, No. 2, pp. 131-143.

[11] Endrias, D. H. and Feng, H. Y., 2003, "Minimum-Zone Form Tolerance Evaluation Using Rigid-Body Coordinate Transformation," *Journal of Computing and Information Science in Engineering*, Vol. 3, pp. 31-38.

[12] Weber, T., Motavalli, S., Fallahi, B. and Cheraghi, S. H., 2002, "A Unified Approach to Form Error Evaluation," *Precision Engineering*, Vol. 26, No. 3, pp. 269-278.

[13] Kanada, T. and Suzuki, S., 1993, "Application of Several Computing Techniques for Minimum Zone Straightness," *Precision Engineering*, Vol. 15, No. 4, pp. 274-280.

[14] Shunmugam, M. S., 1986, "On Assessment of Geometric Errors," *International Journal of Production Research*, Vol. 24, No. 2, pp. 413-425.

[15] Shunmugam, M. S., 1987, "New Approach for Evaluating Form Errors of Engineering Surfaces," *Computer Aided Design*, Vol. 19, No. 7, pp. 368-374.

[16] Sharma, R., Rajagopal, K. and Anand, S., 2000, "A Genetic Algorithm Based Approach for Robust Evaluation of Form Tolerances," *Journal of Manufacturing Systems*, Vol. 19, No. 1, pp. 46-57.

[17] Wen, X. and Song, A., 2003, "An Improved Genetic Algorithm for Planar and Spatial Straightness Error Evaluation," *International Journal of Machine Tools and Manufacture*, Vol. 43, No. 11, pp. 1157-1162.

[18] Suen, D. S. and Chang, C. N., 1997, "Application of Neural Network Interval Regression Method for Minimum Zone Straightness and Flatness," *Precision Engineering*, Vol. 20, No. 3, pp. 196-207.

[19] Huang, S. T., Fan, K. C. and Wu, J. H., 1993, "A New Minimum Zone Method for Evaluating Straightness Errors," *Precision Engineering*, Vol. 15, No. 3, pp. 158-164.

[20] Preparata, F. P. and Shamos, M. I., 1985, "Computational Geometry: An Introduction," *Springer-Verlag,* New York.

[21] Barber, C. B., Dobkin, D. P. and Huhdanpaa, H., 1996, "The Quickhull Algorithm for Convex Hulls," ACM *Transactions on Mathematical Software*, Vol. 22, No. 4, pp. 469-483.