SYMBOLIC DIMENSIONING IN COMPUTER-AIDED DESIGN

by

ROBERT ALLAN LIGHT

SB, May 1979, Massachusetts Institute of Technology

SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE
DEGREE OF

MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

(February 1980)

Signature of Author [illegible]
Department of Mechanical Engineering
February, 1980

Certified by [illegible]
Thesis Supervisor

Accepted by ______________________
Chairman, Department Committee

SYMBOLIC DIMENSIONING IN COMPUTER-AIDED DESIGN

by

ROBERT ALLAN LIGHT

Submitted to the Department of Mechanical Engineering on February, 1980 in partial fulfillment of the requirements for the degree of Master of Science.

ABSTRACT

Symbolic dimensioning permits the modification of part geometry by means of altering the numeric value of the explicit dimensional constraints.

A symbolic dimensioning CAD system was developed which utilized an iterative numerical procedure for determining part geometry from the values of the dimensional constraints. The part geometry was limited to 2-D lines, arcs and circles. The allowable dimensional constraints included horizontal, vertical, linear, angular and radial dimensions. In addition, geometric constraints such as area were incorporated. The dimensional constraints were treated as mathematical equations relating the points in the part. The solution to the system of equations was the part geometry.

An important feature of this system was the capability of sketch input of the part geometry. The dimensional constraints are interactively specified and the geometry modified by alteration of the numeric values of the explicit dimensions.

The mathematical procedures permit the detection of over- and under-dimensioning as well as which dimensions are redundant and which pieces of geometry are unconstrained.

An algorithm was developed which selects the minimum set of equations and unknowns required for solution given a change in dimensions. This reduces the number of equations to which the numerical methods must be applied.

Thesis Supervisor: David C. Gossard
Title: Associate Professor

ACKNOWLEDGEMENTS

There are many people and organizations that I would like to express my appreciation. Without their help, much of this research could not have been accomplished:

To Professor David Gossard for his open ear and guidance throughout this project;

To Doug Wilson for his valuable assistance in the Joint Computer Facility;

To Philip Meyfarth for his support in the many facets of this project;

To James Hurt of the Deere & Company Technical Center for his invaluable mathematical insight;

To the Computervision Corporation for their financial support;

To the Deere & Company Technical Center whose text processing and graphics facilities were used to create this document;

Finally, to my loving parents who were responsible for me to begin with.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

1.0 INTRODUCTION

Systems for computer-aided mechanical design utilize a shape model to represent a mechanical component. The shape model is the computer's representation of the physical geometry of a part. This model is created and manipulated for such purposes as drafting, structural analysis and tool path generation for numerically controlled machines.

Because the design process is iterative in nature, the designer must modify the topology, the specific geometry, or the dimensioning scheme of the shape model many times during the design cycle. The effectiveness of a CAD system depends directly upon the degree to which the shape model represents the corresponding component and the ease with which the shape model can be created and modified.

The objective of this research effort was to develop more flexible methods for the definition and modification of shape models. The method investigated is called "symbolic dimensioning". Symbolic dimensioning is an interactive, graphic procedure which utilizes the dimensioning scheme, in conjunction with the numeric values of the explicit dimensions, to define the geometry of a shape model.

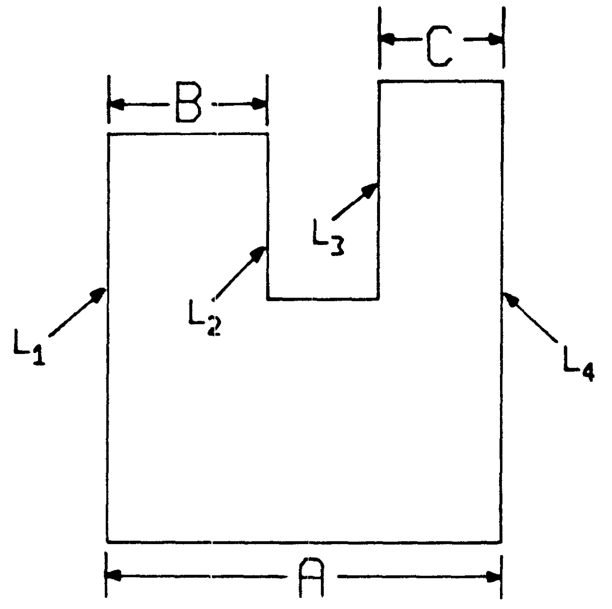
The system incorporates two major functional properties which would facilitate the design process. It allows the user to specify the initial geometry in an approximate fashion. A sketch-like input procedure is used to define the topology of the geometric shape without concern for the exact placement or orientation of the graphic entities. Secondly, after the shape is sketched, dimensional constraints are specified interactively. The desired numeric values of the explicit dimensions are supplied, and the corresponding change in geometry is computed using an iterative numerical method.

The features of sketch-like input and interactive shape modification were developed using a 2-D prototype CAD system (hereafter referred to as the "DIMENSION" system). The application of symbolic dimensioning in the areas of tolerance analysis, tolerance synthesis, dimensional associativity and 3-D design were also explored.

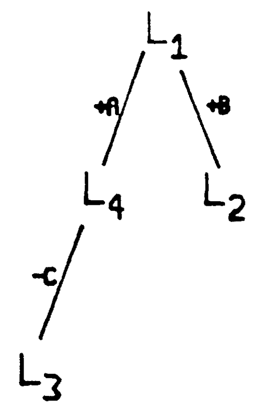
2.0 RELATED WORK

The concept of using explicit dimensions as symbolic variables which, when assigned a value, define the geometry of the shape model, was described by A. G. Requicha [12] and A. M. Gopin [2]. In these separate works, the relationship between dimensions was described by a dimensional tree and the shape models restricted the class of graphic entities to rectilinear line segments in 2-D or orthogonal planes in 3-D. Dimensional trees consisted of nodes and branches where the nodes represented line segments (or planes in 3-D) and the branches represented the dimensional symbol. A separate tree was constructed for each orthogonal direction. Using this procedure, the dimensioned part in Figure 1a would be represented by the tree in Figure 1b (only the horizontal dimensions and tree are shown). The negative sign on the branch from L_4 to L_3 signifies that L_3 is measured to the left of L_4.

Over- or under-dimensioning could be determined by detecting cycles in the dimensional tree. Thus, if a dimension D is added from L_2 to L_4, as in Figure 2a, the graph of Figure 2b results. The complete cycle from L_1 to L_2 to L_4 and back to L_1 indicates a redundancy in dimensions.

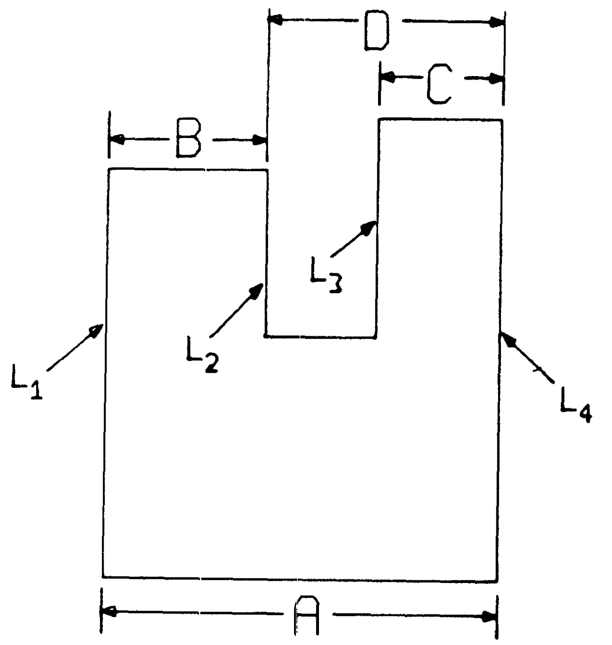


(a)

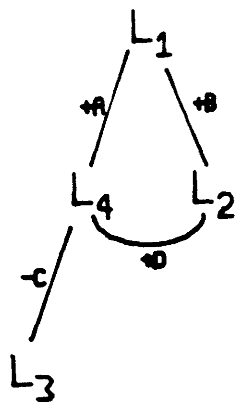


(b)

Figure 1, Symbolic dimensioning using a tree structure.



(a)



(b)

Figure 2, Redundancy detected using cycles in a tree.

Dimensional trees are a useful notation for symbolic dimensioning and detection of redundancies. However, this theory is now limited to shape models consisting of recti-linear segments. The extension to oblique segments, arcs and circles has not been investigated.

In Computervision Corporation's PEP [1] (Parametric Element Processor), and COMVAR, of University of Berlin, the idea of using the dimensional constraints to define the geom-etry of a shape was realized by requiring the user to write an APT-like program which constructs the shape model using the values of a set of input parameters. These systems incorporate some of the concepts of symbolic dimensioning, in that the input parameters control the resultant geometry. A major shortcoming of these approaches is that they do not permit general interactive procedures for graphic input and geometric modification.

The work of R. C. Hillyard and I. C. Braid [6] and [7] laid an important foundation for this work. These papers developed a theory by which variations in the locations of individual points of a shape can be related to variations in dimensional constraints. For example, consider two points P_1 and P_2 in two-space which are to be constrained by a linear dimension. The location of each point is varied by a small amount represented by vectors d_1 and d_2 as in Figure 3. If dimension A is considered a vector quantity $\vec{A}$ from P_1 to P_2, then the variation dA of dimension A can be approximated by equation 1.

$$dA = \frac{(d_2 - d_1) \cdot \vec{A}}{|\vec{A}|} \qquad (1)$$

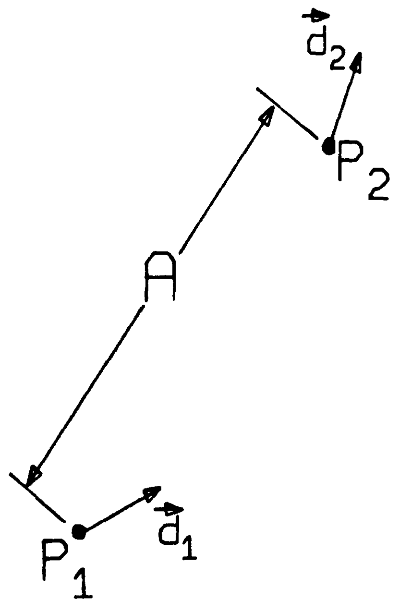


Figure 3, Variation of two points.

Variations of other dimensional constraints can be similarly defined. When the variations are small, the relationships between the variations in dimensions to the variations in point positions are linear. These linear equations can be solved to determine the variations in geometry which correspond to a given variation in dimension. This relationship can be described by equation 2;

$$\underline{dx} = \bar{\underline{r}}^{-1} \underline{dA} \tag{2}$$

where the vector, $\underline{dx}$ represents the variation in geometry, the matrix $\bar{\underline{r}}$ is termed the "Rigidity" matrix and the vector $\underline{dA}$ represents the variation in dimensions. An iterative procedure can be implemented to modify the geometry such that the geometry is compatible with the dimensions.

In the author's Bachelor's Thesis [9], a prototype symbolic dimensioning procedure was developed. This algorithm assumes that each dimensional constraint could be described as a mathematical equation involving the coordinates of the points in the shape. The geometry of the shape model is the solution to all the dimensional "equations".

These equations are generally non-linear and must be solved by a numerical method such as the Newton-Raphson method. The Rigidity matrix, $\bar{\underline{r}}$ and the variation in dimensions, $\underline{dA}$ in equation 2 are identical to the Jacobian matrix and the residual vector in the Newton-Raphson method. The approach taken by Hillyard and Braid is basically a special case of the more general development in this work. The present approach can be more easily extended to include general geometric constraints such as area, mass, or inertia, and higher order graphic entities such as arcs, circles and 3-D surfaces.

3.0 GENERALIZED DIMENSIONAL CONSTRAINTS

The concept of generalized dimensional constraints follows directly from conventional engineering practice for dimensioning mechanical drawings. Dimensions in a mechanical drawing may be thought of as constraints on the permissible locations of the portions of geometry to which they refer. A correctly dimensioned mechanical drawing is described as a drawing, were the position of each piece of geometry (points, lines, arcs, circles, etc.) may be determined from the dimensional information contained in the drawing.

3.1 Types of Constraints

The term "dimensional information" used above, includes three types of constraints. The first type is the conventional explicit dimension. It constrains, for example, two points to be a specified distance apart. The second type of information contained in a drawing can be termed "implicit constraints". Implicit constraints include the specification that lines are horizontal, vertical, parallel or perpendicular.

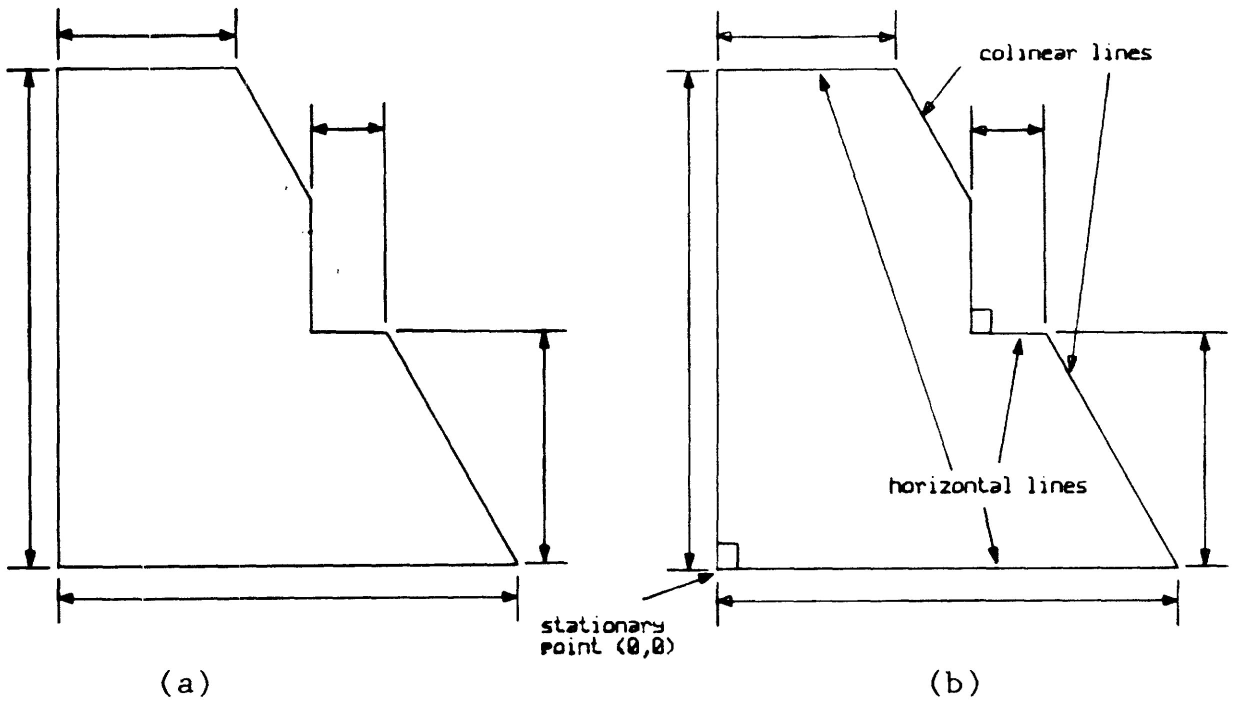


Figure 4, Implicit dimensions; (a) original drawing; (b) drawing with implicit dimensions shown.

The part shape in Figure 4a would be considered fully dimensioned. The implicit constraints, added in Figure 4b, are generally inferred by the person viewing the part shape. Implicit dimensions are as important as the explicit dimensions in determining the position of each entity of the part shape. Thus, explicit and implicit dimensions are simply subsets of the larger class of generalized dimensional constraints.

In addition to these two classes of constraints, there exists a third class of "geometric constraints". These are constraints on such geometrical properties as mass, surface area, inertia and center of mass. As with implicit and explicit dimensional constraints, geometric constraints determine a spatial relationship of one geometric entity to a set of other entities.

3.2 Over-dimensioning and Under-dimensioning

With this description of generalized dimensional constraints, a qualitative discussion of the problem of over- or under-dimensioning can be presented. As discussed above, a properly dimensioned mechanical drawing is defined as one where the positions of all the graphic entities can be determined from the complete set of general dimensional constraints. The case of an over-dimensioned drawing can be detected when the position of a graphic entity is specified by more than one set of constraints. For example, in Figure 5, the horizontal position of line 3 can be determined from the dimensional sets AC and BCD.

The case of under-dimensioning can be detected when there is no complete set of constraints which will determine the position of a graphic entity. In the case of 2-D rectilinear shapes, Requicha [12] detected the case of over-dimensioning by looking for a node (which represents the position of an

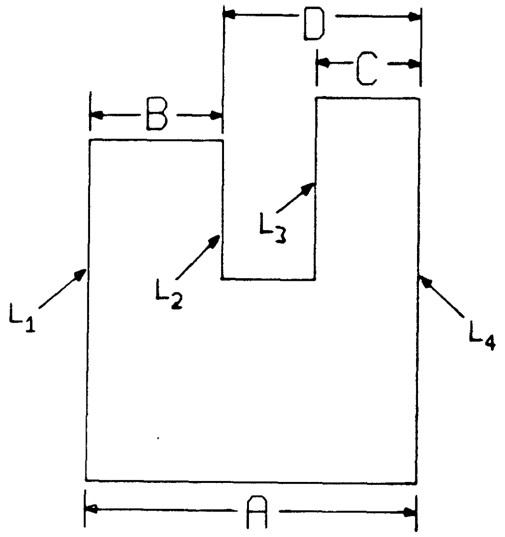


Figure 5, Condition of over-dimensioning.

entity) in a graph that was accessible from two different paths. The under-dimensioned case was detected by looking for a node which was not accessible from any path.

The approach used in this work for detecting over- and under-dimensioning is discussed in section 5.2.

4.0 THE SHAPE MODEL

A model for the geometrical shape is required to delineate the entities that the generalized dimensions are to constrain. In this paper, only geometries of two dimensions were considered. The primitive geometrical entity constrained was the two-dimensional point. All higher order geometric entities such as arcs or circles were defined with respect to their defining points (verticies, points of tangency, center points, etc.).

The shape model, therefore, consists of a collection of points in two-space. The geometry of the shape is defined by specifying the position of the points in the part shape. The generalized dimensional constraints described in the previous section are mathematical equations which state some relationship between the x,y coordinates of the part shape. The total set of dimensional information must be sufficient to define the coordinate values of every point in the part.

5.0 MATHEMATICAL DESCRIPTION OF GENERALIZED DIMENSIONS

The preceeding discussion has delineated two assumptions. First, the only geometric entity in the shape model is the two-dimensional point. Secondly, the generalized dimensional constraints define the position of each point in two-space. The objective of this section is to provide a mathematical framework for a more quantitative discussion of dimensions and their relationship to geometry.

The basic theory was discussed in the author's Bachelor's thesis [9] but is included, here, for completeness. In order to illustrate the approach taken here, an example will be used.

A triangle, as in Figure 6 is defined by the coordinates of its three points, which can be written as a six component vector, $\underline{X}$.

$$\underline{X} = \{X_1, Y_1, X_2, Y_2, X_3, Y_3\}^T \qquad (3)$$

These points are constrained by three dimensions: a linear dimension, A, from P_1 to P_2; a linear dimension, C, from P_1 to P_3; and a vertical dimension, B, from P_2 to P_3.

If line P_1P_3 is constrained to remain horizontal, all the points in the shape are defined relative to each other and the shape is constrained from solid body rotation. To constrain the shape from solid body translation, it is necessary to fix the position of one point. Therefore, if P_1 is positioned at the origin (0,0), each point is uniquely positioned in the X-Y plane and the shape is sufficiently dimensioned.

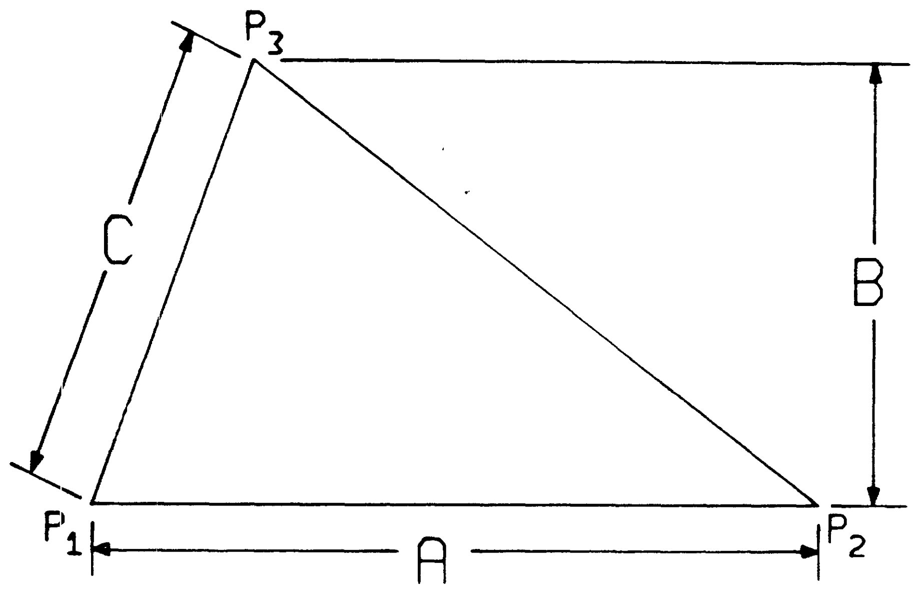


Figure 6, Example shape.

The constraints represented by dimensions A, B, and C can be described by the following equations;

$$(X_1-X_2)^2 + (Y_1-Y_2)^2 = A^2 \tag{4}$$

$$Y_3-Y_2 = B \tag{5}$$

$$(X_1-X_3)^2 + (Y_1-Y_3)^2 = C^2 \tag{6}$$

To constrain the shape from solid body translation;

$$X_1 = 0 \tag{7}$$

$$Y_1 = 0 \tag{8}$$

To constrain line segment P_1-P_2 to remain horizontal;

$$Y_2-Y_1 = 0 \tag{9}$$

We have six equations (4-9) and six unknowns ($X_1, Y_1, X_2, Y_2, X_3, Y_3$). If there were more equations than twice the number of points, the shape would be over-dimensioned. If there were fewer equations than twice the number of points, the shape would be under-dimensioned.

The purpose of the symbolic dimensioning algorithm is to determine the positions of the three points given the values of the dimensions A, B, and C. The coordinates of the points can be seen to satisfy the following set of equations;

$$f_1 = (X_1-X_2)^2 + (Y_1-Y_2)^2 - A^2 = 0 \tag{10}$$

$$f_2 = Y_3 - Y_2 - B = 0 \tag{11}$$

$$f_3 = (X_1-X_3)^2 + (Y_1-Y_3)^2 - C = 0 \tag{12}$$

$$f_4 = X_1 = 0 \tag{13}$$

$$f_5 = Y_1 = 0 \tag{14}$$

$$f_6 = Y_2 - Y_1 = 0 \tag{15}$$

The determination of the points' location requires that the above set of simultaneous non-linear equations be solved. The solution procedure used in this application was the Newton-Raphson method. (See Appendix A.2 for a discussion of the Newton-Raphson method.) The Jacobian matrix (or Rigidity matrix as termed by Hillyard [5]) and residual vector for the equations above are shown in the following matrix equation;

$$\begin{bmatrix} 2(X_1-X_2) & 2(Y_1-Y_2) & -2(X_1-X_2) & -2(Y_1-Y_2) & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \\ 2(X_1-X_3) & 2(Y_1-Y_3) & 0 & 0 & -2(X_1-X_3) & -2(Y_1-Y_3) \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} dx_1 \\ dy_1 \\ dx_2 \\ dy_2 \\ dx_3 \\ dy_3 \end{bmatrix} = \begin{bmatrix} -f_1 \\ -f_2 \\ -f_3 \\ -f_4 \\ -f_5 \\ -f_6 \end{bmatrix}.$$

where $X_1, Y_1, X_2, Y_2, X_3, Y_3$, represent the current estimate of the point coordinates.

Let the current estimate be represented by X_n;

$$\underline{X}_n = \{X_1, Y_1, X_2, Y_2, X_3, Y_3\}_n \tag{16}$$

The values of the residuals f_1, f_2, f_3, f_4, f_5, and f_6, as defined by equations (10-15) are evaluated using the current estimate, $\underline{X}_n$. The Jacobian matrix is also evaluated using the current estimate.

When the numeric value of a dimension is altered, the residuals are no longer zero. The solution to the vector equation 17 produces the vector $\underline{dx}$ which is the displacements of all the points in the shape.

$$\underline{dx} = \underline{\bar{J}}^{-1} \underline{R} \tag{17}$$

where $\underline{\bar{J}}$ is the Jacobian, and $\underline{R}$ is the vector of negative residuals. Thus, a new shape estimate is provided by summing the current estimate with the coordinate displacements;

$$\underline{X}_{n+1} = \underline{X}_n + \underline{dx} \tag{18}$$

Equation 17 may then be evaluated at the new estimate, $\underline{X}_{n+1}$. This iteration continues until the residuals are sufficiently small, at which time the geometry is consistant with the dimensions.

5.1 Non-uniqueness of Solution

An important property is that there are many mathematically valid solutions to a set of non-linear equations. For example, all the shapes shown in Figure 7 would satisfy the system of equations (10-15) and therefore are valid representations of the same set of dimensional constraints - A, B, and C.

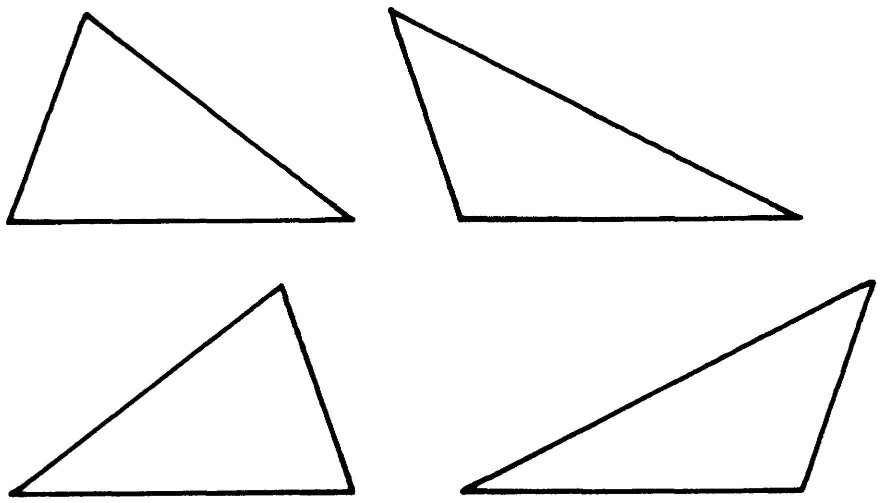


Figure 7, Different solutions.

If the initial estimate, $\underline{X}_i$, is sufficiently close to the desired final solution of the system of equations, the iteration will converge to the desired shape. (i.e.-the Newton-Raphson method will converge to a shape with the same general topology as that of the initial estimate.) This is the case with the application of the Newton-Raphson method in the symbolic dimensioning algorithm: Originally, the points are positioned so that the topology of the part is given. The residuals may not be zero, but the current shape results in a good initial estimate which approximates the desired geometry.

5.2 Detection of Redundant Dimensioning

The requirement that there be a number of equations equal to twice the number of points is a necessary but not a sufficient condition for a valid dimensioning scheme. Referring to the example of the triangle;

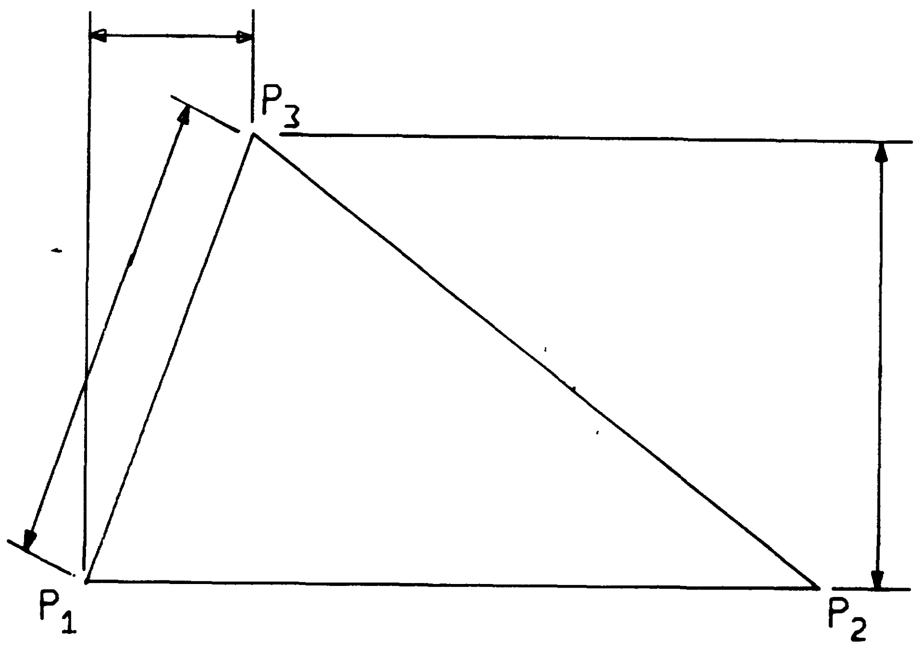


Figure 8, Redundant dimensioning scheme.

Upon counting the number of equations (there are three implied dimensions as in the previous example) we find that the number of equations equals twice the number of points in the shape. Therefore, the triangle has the correct number of dimensions. When the Newton-Raphson method is implemented with this example, it would be determined that the Jacobian was singular. This condition indicates that at least two of the constraints are not independent and that at least one of the dimensions is redundant. Note that in Figure 8, point 3 is over-constrained and point 2 is under-constrained.

From linear algebra it is known that for an nxm matrix, A;

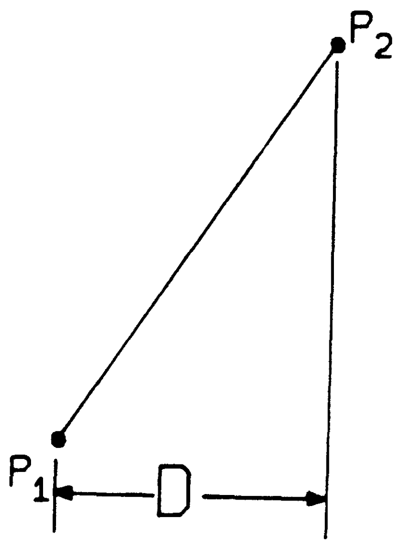
$$\text{Rank}(A) \leq \min(n,m) \tag{19}$$

Therefore, if $n \neq m$ the matrix A is always singular. In the Newton-Raphson method, if the number of equations does not equal the number of unknowns, the Jacobian is not square and is necessarily singular. Therefore, the condition that the Jacobian be non-singular is a necessary and sufficient condition for a valid dimensioning scheme.

5.3 Elementary Dimensional Constraints

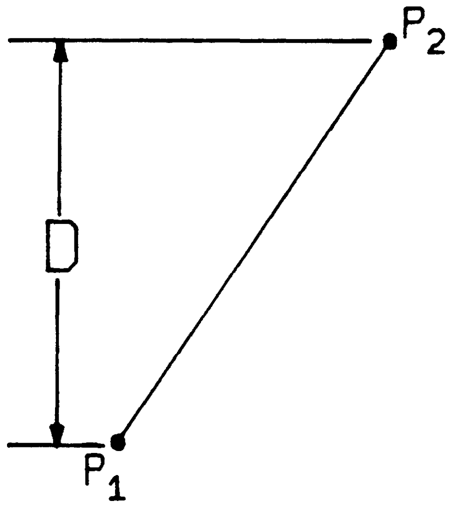
Up to this point, a limited number of dimensional constraints and their corresponding equations have been identified. The purpose for this section is to present a more complete set of elementary constraints on geometry and their respective constraint equations. The term "elementary" denotes that each constraint may be defined by a single equation. "Compound" dimensional constraints require two or more equations. Their relationship to higher order entities will be discussed in section 5.5.

5.3.1 Horizontal dimension



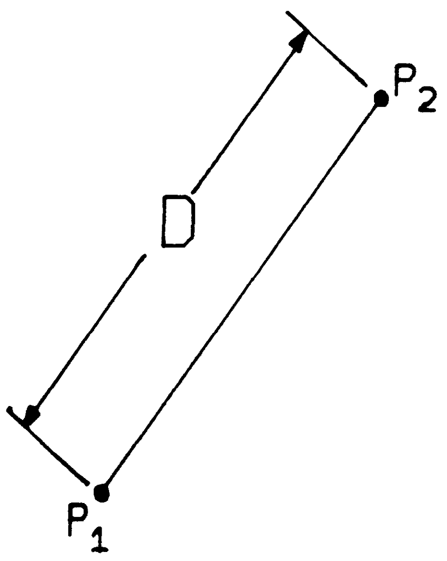
$$X_2 - X_1 - D = 0 \tag{20}$$

5.3.2 Vertical Dimension



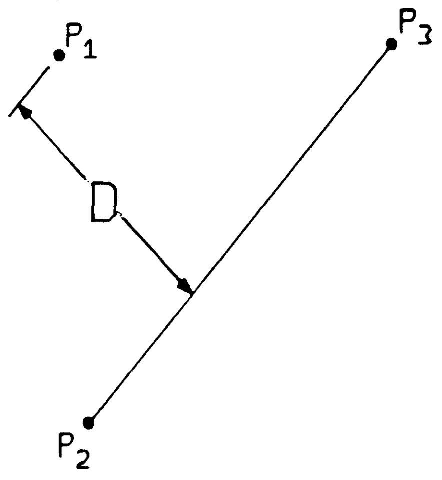
$$Y_2 - Y_1 - D = 0 \tag{21}$$

5.3.3 Linear dimension



$$(X_1-X_2)^2 + (Y_1-Y_2)^2 - D^2 = 0 \tag{22}$$

5.3.4 Distance from a point to a line



To constrain the distance from a point to a line, two vectors must be defined: the unit vector $\hat{U}$ from P_2 to P_3 and the vector $\vec{V}$ from P_2 to P_1. The distance, D, is then the cross product of $\hat{U}$ with $\vec{V}$.

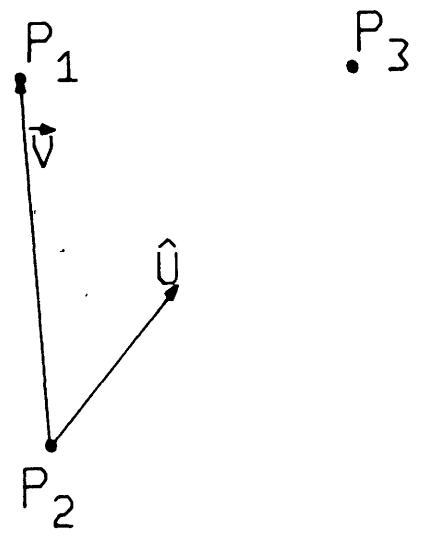


Figure 9, Cross product.

First, define the unit vector $\hat{U}$ from P_2 to P_3;

$$\hat{U} = \frac{(X_3-X_2)}{|P_2P_3|}\,\hat{i} + \frac{(Y_3-Y_2)}{|P_2P_3|}\,\hat{j} \tag{23}$$

The vector $\vec{V}$ from P_2 to P_1 can be written as;

$$\vec{V} = (X_2-X_1)\hat{i} + (Y_2-Y_1)\hat{j} \tag{24}$$

The equation for the distance from a point to a line becomes;

$$f = \hat{U} \times \vec{V} - D = 0 \tag{25}$$

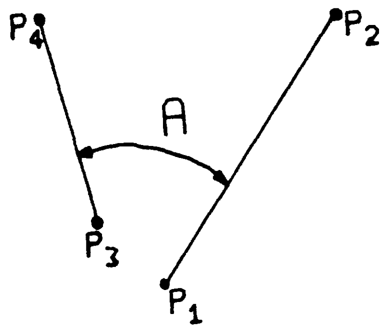
Or;

$$f = U_x(Y_2-Y_1) - U_y(X_2-X_1) - D = 0 \tag{26}$$

where U_x and U_y are defined as in equation 27.

$$U_x = \frac{X_3-X_2}{|P_2P_3|} \qquad\qquad U_y = \frac{Y_3-Y_2}{|P_2P_3|} \tag{27}$$

5.3.5 Angular dimension



The constraint equation for an angular dimension between vector P_1P_3 and vector P_3P_4 depends upon the value of the actual angle between the vectors. These equations are derived by evaluating the cross product or the dot product between the two vectors.

For $0<A<\frac{\pi}{4}$ and $\frac{3\pi}{4}<A<\pi$;

$$\frac{(X_2-X_1)(Y_4-Y_3) - (Y_2-Y_1)(X_4-X_3)}{((X_2-X_1)^2+(Y_2-Y_1)^2)^{1/2}((X_4-X_3)^2+(Y_4-Y_3)^2)^{1/2}} - SIN(A)=0 \qquad (28)$$

For $\frac{\pi}{4}<A<\frac{3\pi}{4}$;

$$\frac{(X_2-X_1)(X_4-X_3) + (Y_2-Y_1)(Y_4-Y_3)}{((X_2-X_1)^2+(Y_2-Y_1)^2)^{1/2}((X_4-X_3)^2+(Y_4-Y_3)^2)^{1/2}} - COS(A)=0 \qquad (29)$$

The two different equations are necessary due to the difference in accuracy between the sine and the cosine functions in the corresponding ranges.

A variational approach for constraining an angle was first presented by Hillyard [5] and was incorporated as a part of the prototype program developed as part of the author's Bachelor's thesis [9]. As a result of considerations such as stability, convergence, and uniformity of approach, it was concluded that the present formulation described by equations 28 and 29, was more effective than the variational approach.

5.3.6 Equal linear distances

This constraint specifies that the distance between a pair of points is equal to the distance between a second pair of points.

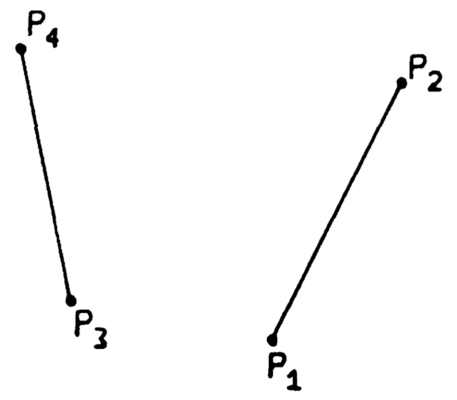


Figure 10, Equal linear distance constraint.

This constraint does not assign a value to the distances but merely constrains the two distances to be equal. This constraint will be important in defining radial measure.

$$(X_2-X_1)^2 + (Y_2-Y_1)^2 - (X_4-X_3)^2 - (Y_4-Y_3)^2 = 0 \qquad (30)$$

Using an analogous formulation, the same type of constraint may be defined from any of the elementary constraints previously defined. For example, equal vertical displacement between P_1-P_2 and P_3-P_4 may be constrained by equation 31.

$$Y_4-Y_3-(Y_2-Y_1) = \emptyset \qquad (31)$$

5.4 Compound Dimensional Constraints

Compound dimensional constraints are simply combinations of two or more elementary constraints. Therefore, a compound constraint is represented by two or more equations relating the x,y, coordinates of the points to be constrained. For example, if a constraint is to be defined for a distance, D, between two (parallel) lines, two elementary equations must be used. The first constraint could be an angular dimension between $\overline{P_1P_2}$ and $\overline{P_3P_4}$, whose value is zero. The second constraint is the distance from point, P_3, to the line $\overline{P_1P_2}$. The value of this dimension would be the distance, D, between the two parallel lines.

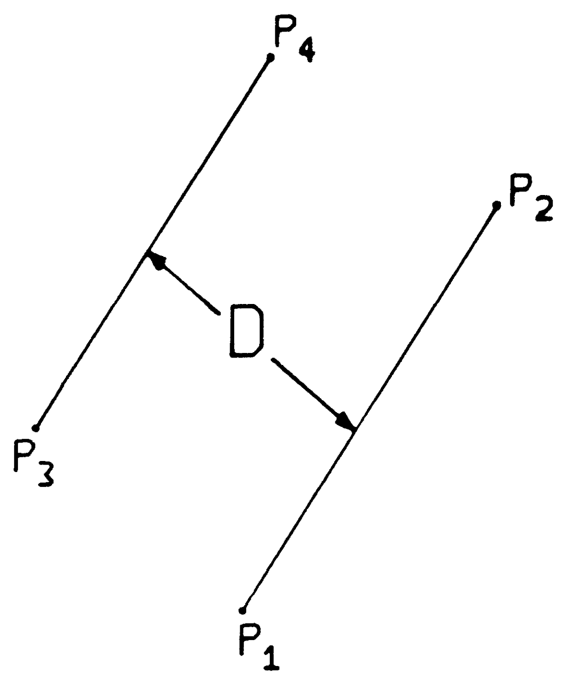


Figure 11, Distance between two lines.

An alternative scheme would be to constrain the distance from P_3 to line P_1P_2. In order to constrain P_3P_4 and P_1P_2 to be parallel, a constraint could be used which constrains the distance from P_3 to line P_1P_2 to be "equal" to the distance from P_4 to line P_1P_2. The constraint equation can be formed by using the notation of equation 27 and the method described in section 5.3.6. The result is given below.

$$U_x(Y_3-Y_1) - U_y(X_3-X_1) - [U_x(Y_4-Y_1) - U_y(X_4-X_1)] = 0 \qquad (32)$$

Or;

$$U_x(Y_3-Y_4) - U_y(X_3-X_4) = 0 \tag{33}$$

Note that this constraint (equation 33) is equivalent to the vanishing of the cross product between $\overline{P_1P_2}$ and $\overline{P_3P_4}$. This is the form of the angle constraint where the angle vanishes. Therefore, this method of constraint is equivalent to the method discussed above. The second method is actually used in the DIMENSION system due to the reduced amount of computation.

5.5 Extensions to Higher Order Graphic Entites

The mathematical procedures presented in the previous section operate on a shape model which includes simple points as the basic entities to be constrained. A point may be considered a center of a circle, a vertex between two lines, or a point of tangency.

In order to extend the basic approach to include arcs, circles or any other entity, a set of points must be identified which define the position, orientation and size of the entity. These points must then be constrained in a manner which is compatible with current dimensioning practices. It is important to realize that the shape model is not being altered in order to include arcs and circles. Points are still the only entity in the shape model. We are simply defining arcs and circles in terms of their defining points.

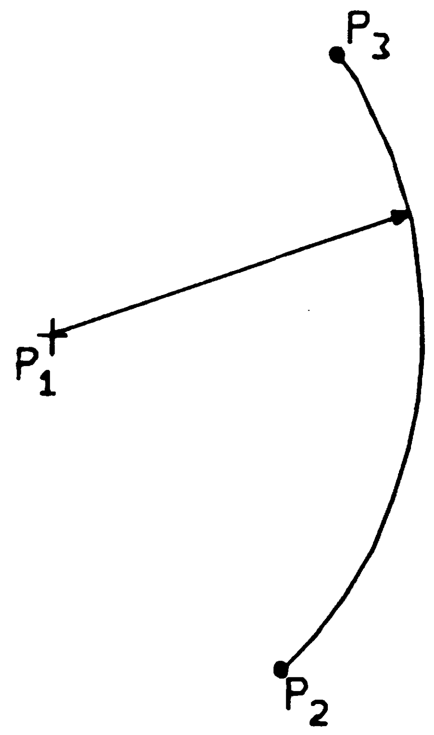


Figure 12, Defining points of a circular arc.

The three points in Figure 12 define an arc sufficiently. In order to constrain the points in the radial direction, the distance between P_1 and P_2 is constrained to be equal to the distance from P_1 and P_3. If the radius is to be specified, the linear distance between either P_1 and P_2 or between P_1 and P_3 is constrained.

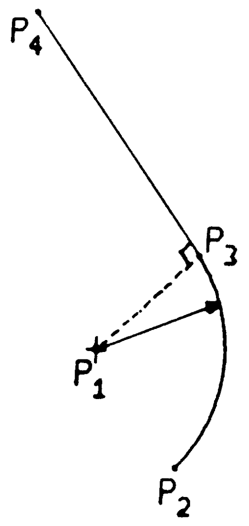


Figure 13, Tangency between line and arc.

If the arc is to be tangent to a line, as in Figure 13, an angular constraint is used between $\overline{P_3P_4}$ and $\overline{P_3P_1}$. The value of this angle is set to 90°.

This procedure constrains the three points according to the users requirements. For display purposes, the three points are simply marked as belonging to an arc. When the plotting program draws the shape contour, an arc is generated between P_2 and P_3 about P_1.

The arc can be positioned in two-space by constraining any of its defining points. The dimensioning scheme of Figure 14 cannot be accomplished with only P_1, P_2, and P_3 defined. The horizontal dimension to the extremum of the arc does not constrain a defining point.

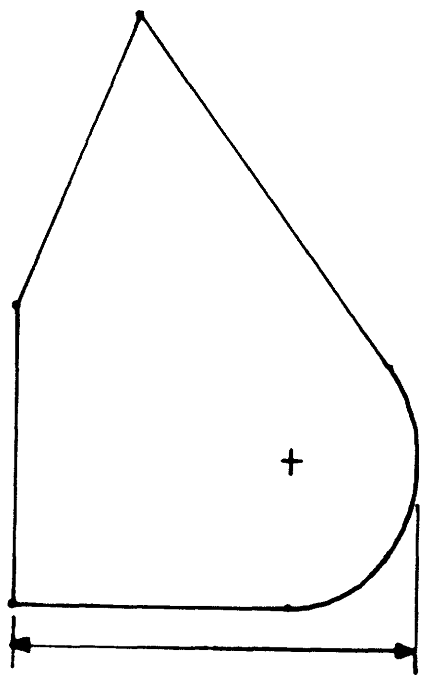


Figure 14, Dimensioning to the extremum of an arc.

In order to dimension to an extremum of an arc, an additional point must be used. This point will be constrained to be at the extremum of the arc relative to the desired dimension.

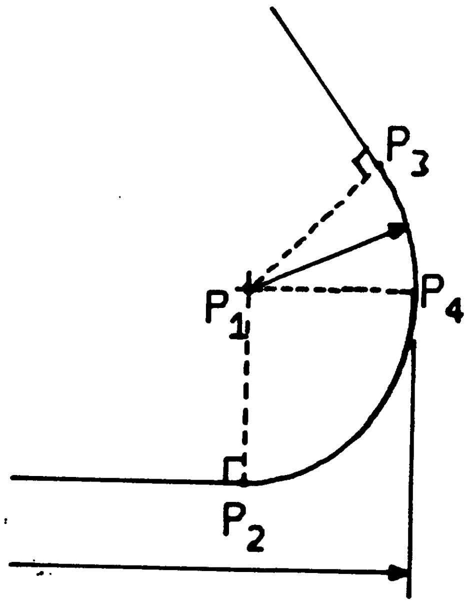


Figure 15, Dimensioning to an arc.

Since two unknowns have been added (the x,y, coordinates of P_4), two additional constraints must be supplied. The first is that the distance between P_1 and P_4 be equal to the distance between P_1 and P_3 (or P_1 and P_2). If a horizontal dimension to the arc is desired, the second constraint is that the "line" between P_1 and P_4 be horizontal (i.e.- $Y_4-Y_1 = 0$). If a vertical dimension to an arc is needed, this second constraint between P_4 and P_1 must be $X_4-X_1 = 0$ or a vertical "line" constraint.

If a linear dimension, as in Figure 16, between a point P_5 and an extremum point, P_4, is desired, the second constraint must be a parallel constraint between $\overline{P_1P_4}$ and $\overline{P_1P_5}$ (i.e.- angle between $\overline{P_1P_4}$ and $\overline{P_1P_5}$ is zero). This would constrain P_5,P_1 and P_4 to be colinear.

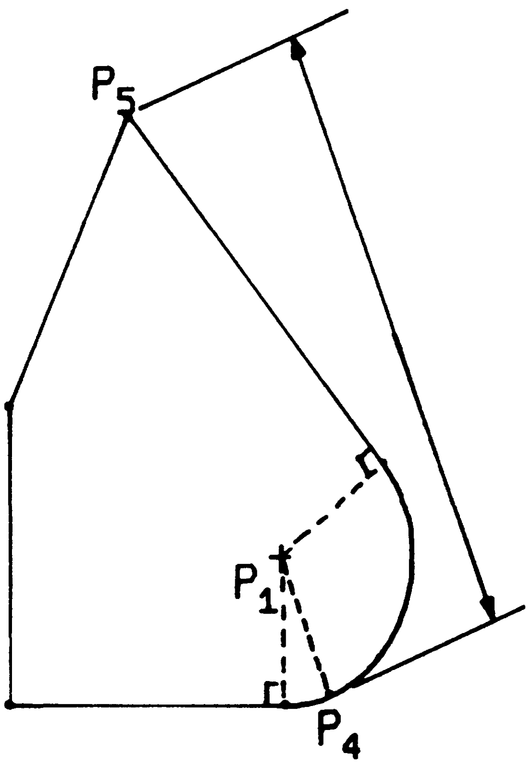


Figure 16, Linear dimension from a point to an arc.

A circle may be incorporated by simply positioning the center of the circle and specifying the radius. The radius is used only to draw the actual circle around the center point. If a dimension is desired to a point on the circle, a method analogous to that just described for the arc could be used.

5.6 Geometric Constraints

General geometric constraints, such as area, may also be defined. The area of the shade triangle in Figure 17, can be evaluated from the cross product of two vectors as in Equation 34.

$$A = (X_1Y_2 - Y_1X_2)/2 \tag{34}$$

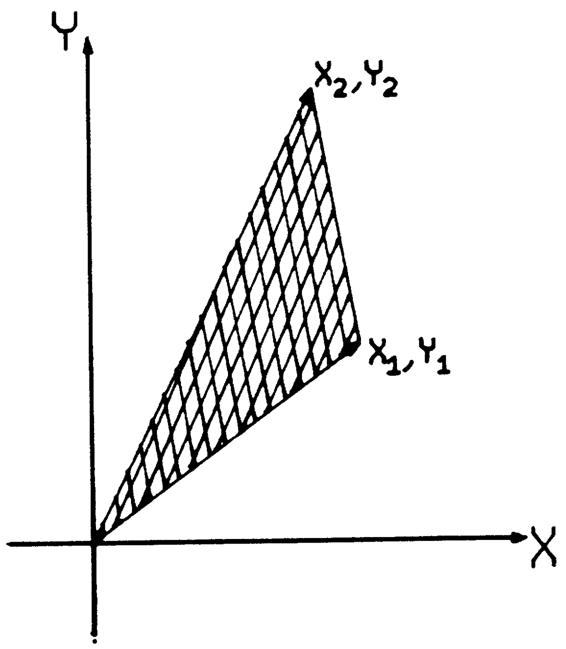


Figure 17, Area between two vectors.

For any N-sided polygon (Figure 18), the area is defined by equation 35. The inner summation is positive if one moves around the contour in the counter-clockwise direction. If one moves in a clockwise direction, the summation is negative. Therefore, the area of a part shape containing a number of polygons, with some polygons as "holes", as in Figure 18, can also be determined by equation 35 by traversing any polygon in the direction which keeps the inside of the bounded region to the left.

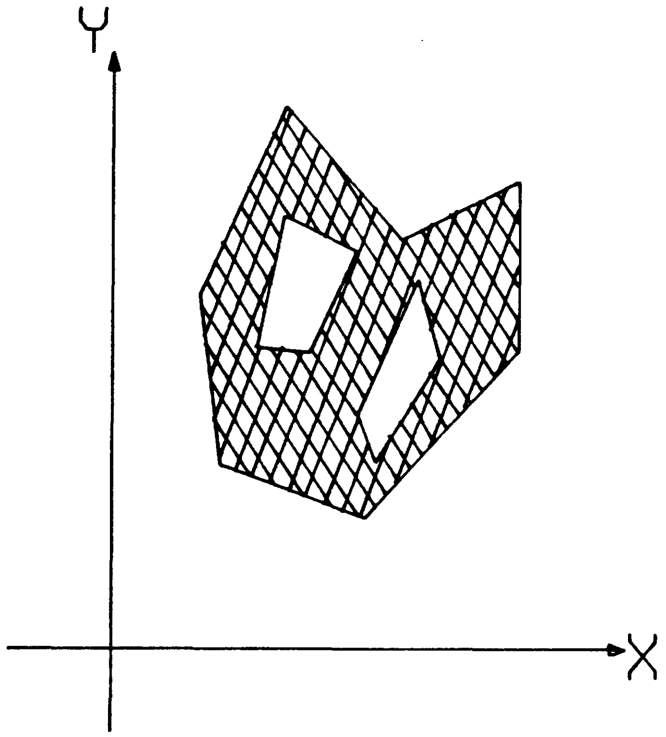


Figure 18, General polygon with "holes".

Equation 35 represents a constraint on the area of the part shape. In a similar manner, the equation for the inertia tensor and center of mass may be defined by equations 36-42.

$$A = \frac{1}{2}\sum_j \left(\sum_i (X_i Y_{i+1} - Y_i X_{i+1})\right) \qquad (35)$$

$$M_x = \frac{1}{6}\sum_j \left(\sum_i (X_i Y_{i+1} - Y_i X_{i+1})(Y_i + Y_{i+1})\right) \qquad (36)$$

$$M_y = \frac{1}{6}\sum_j \left(\sum_i (X_i Y_{i+1} - Y_i X_{i+1})(X_i + X_{i+1})\right) \qquad (37)$$

$$I_x = \frac{1}{12}\sum_j \left(\sum_i (X_i Y_{i+1} - Y_i X_{i+1})(Y_i^2 + Y_i Y_{i+1} + Y_{i+1}^2)\right) \qquad (38)$$

$$I_y = \frac{1}{12}\sum_j \left(\sum_i (X_i Y_{i+1} - Y_i X_{i+1})(X_i^2 + X_i X_{i+1} + X_{i+1}^2)\right) \qquad (39)$$

$$I_{xy} = \frac{1}{12}\sum_j \left(\sum_i (X_i Y_{i+1} - Y_i X_{i+1})(X_i Y_i + \frac{1}{2}(X_i Y_{i+1} + Y_i X_{i+1}) + X_{i+1} Y_{i+1})\right) \qquad (40)$$

$$X_c = M_x / A \qquad (41)$$

$$Y_c = M_y / A \qquad (42)$$

where $j=1,\ldots,$number of loops
$i=1,\ldots,$number of points

The use of Equation 35, in the Newton-Raphson method is analogous to the specification of any other mathematical constraint on the part geometry. A single row in the Jacobian is defined by the partial derivative of the area equation with respect to each coordinate of the shape. The residual is the difference between the desired area and the present area of the shape. Since the area is not defined when any two polygons interfere, checks must be made within each iteration to insure the integrety of the part shape.

The area constraint, equation 35, is not applicable to a part shape which contains circular arcs. Even though an arc may be approximated by a polygon, the points in an arc (the verticies of the approximating polygon) do not explicitly appear as unknowns in the numerical procedures.

Given a part shape, as in Figure 19, the area of a polygon with verticies at points 1, 2, 3, and 4 can be constrained by equation 35. This area is represented by the shaded region in Figure 20.

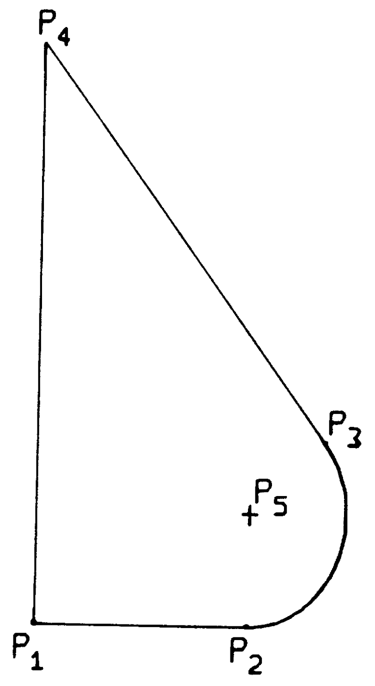


Figure 19, Part shape with an arc.

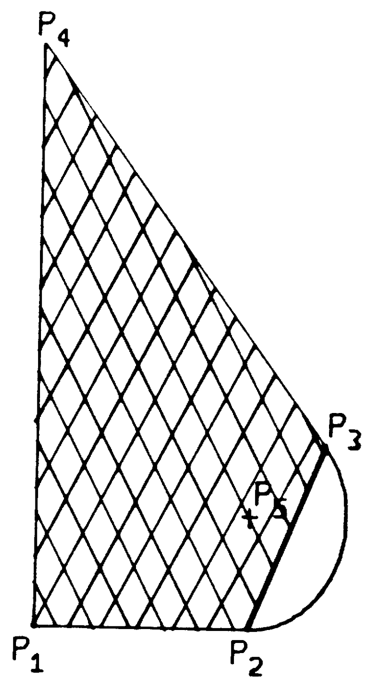


Figure 20, Area constrained by equation 35.

An additional term must be added to equation 35 to represent the area contained between the circular segment and line segment P_2-P_3. This area is determined by equation 43.

$$A_{seg} = \tfrac{1}{2}R^2(\alpha - \frac{c}{2R}) \qquad (43)$$

where; α = angle of arc,
R = radius of arc,
C = chord length of arc.

The area contained in all the arc segments can be written;

$$A = \tfrac{1}{2}\sum_j S_j R_j^2(\alpha_j - \frac{C_j}{2R_j}) \qquad j = 1,\ldots,\text{all arcs} \quad (44)$$

The variable, S, has a value of +1 or -1 depending on whether the area is positive or negative.

S = +1 if the center of the arc is inside the part contour,

S = -1 if the center is outside the part contour.

Thus, the complete equation which constrains the area of a part shape can be written;

$$A_T = \frac{1}{2}\sum_n^N \left[\left(\sum_i^p x_i y_{i+1} - y_i x_{i+1} \right) + \left(\sum_j^a S_j R_j^2 \left(\alpha_j - \frac{C_j}{2R_j} \right) \right) \right] \qquad (45)$$

where; $j = 1,\ldots,a$ — a = number of arcs,
$i = 1,\ldots,p$ — p = number of points,
$n = 1,\ldots,N$ — N = number of contours.

As before, a single row in the Jacobian is computed by differentiating equation 45 with respect to each of the coordinates. The residual is simply the difference between the present area and the desired area.

6.0 NUMERICAL METHODS

The method used to solve the system of non-linear equations is the Newton-Raphson iteration. This procedure is outlined in Appendix A. The purpose of this section is to discuss some of the modifications made to the basic Newton-Raphson method. These modifications were necessary in order to accomplish three major objectives:

The first was to permit the numerical procedures to partially solve a system of dependent equations, that is, a system for which the Jacobian is singular.

The second objective was to implement techniques for modifying the rate of convergence of Newton's method. This entails adjusting the amount by which the geometry is altered on each iteration.

The third goal was to reduce the number of constraint equations to which the numerical procedures must be applied. When one or more dimensions are altered, there exists a minimum number of equations which must be solved in order to compute the new geometry. In order to accomplish this, a segmentation algorithm was developed which determines a subset of equations and unknowns which must be included in the iteration given any dimensional change.

6.1 Stability of Numerical Procedures

In the application of an iterative numerical method, the question of numerical stability is of concern. It is often difficult to determine if a solution can be found to a set of non-linear algebraic equations. The rate and success of convergence depends not only on the set of equations but also on the "initial guess" and the final solution.

A major cause of instability is an "ill-conditioned" dimensioning scheme. A dimensioning scheme is considered ill-conditioned when a small variation in the numeric value of a dimension causes a large variation in the geometry. An example of this type of dimensioning scheme is shown in Figure 21.

With this dimensioning scheme, the X-coordinate of point 3 will change by a large amount with a small change in angle, A.

The problem of ill-conditioned shapes is compounded by the fact that a shape may be "well behaved" for one set of numeric values for its dimensional constraints but "ill-conditioned" for another set. If the shape in Figure 21 is used with the same dimensioning scheme yet with a different set of numeric values for the angle, A, and dimension D, as in Figure 22, the shape becomes well behaved.

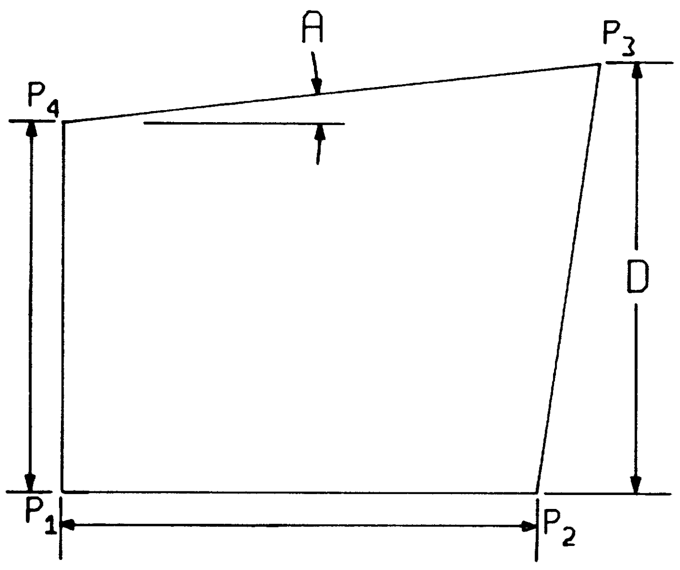


Figure 21, An "ill-conditioned" dimensioning scheme.

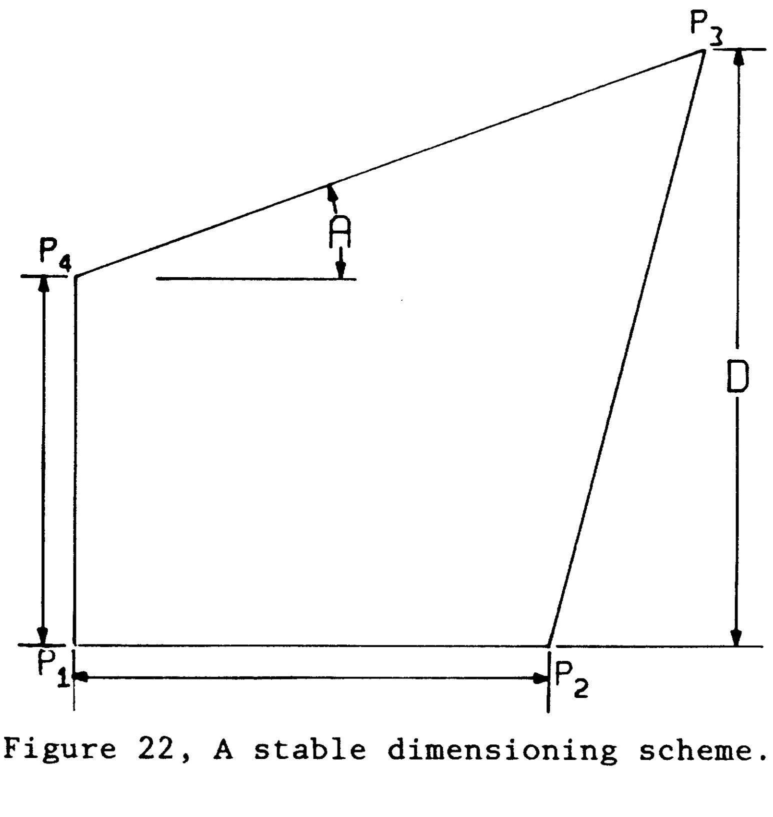


Figure 22, A stable dimensioning scheme.

When the angle, A, approaches zero and the dimension D approaches the value of dimension B, the Y-coordinate of point 3 becomes over-constrained and its X-coordinate becomes under-constrained. In Newton's method, as this condition occurs, the determinant of the Jacobian vanishes.

It is useful to look at this problem in more abstract geometrical terms. The solution to a set of N equations is a single point in multi-dimensional space. The number of dimensions in this space is N+1 where the added dimension can be thought of as the magnitude of the residual vector. An equation represents a hyper-surface in this multi-dimensional space where the magnitude of the residual vector varies over the surface.

The intersection of the N surfaces is essentially an N+1 dimensional curve. The purpose of any numerical method is to find the point along this curve where the residuals vanish. In terms of constraining the geometry of a shape with N points, we have 2N equations in 2N+1 dimensions. The geometry which conforms to a set of explicit dimensions corresponds to the solution point in 2N+1 space;

$$(X_1, Y_1, X_2, Y_2, X_3, Y_3, \ldots X_n, Y_n, 0)$$

The last coordinate is the magnitude of the residual vector which is zero at the solution point.

Newton's method computes the gradient vector along this multi-dimensional curve. Problems arise when the gradient vector vanishes. This would correspond, in the two-dimensional case, to the first derivative vanishing at a maximum or minimum (extremum).

6.2 Direct Matrix Techniques

An extremum may be encountered due to either a redundantly dimensioned shape for which the Jacobian will always be singular, or where, for any reason, the Jacobian becomes singular or near-singular somewhere in the iteration.

From section 5.0, the increment, $\underline{dx}$, in the "guess" was determined by equation 46.

$$\underline{\bar{J}}\ \underline{dx} = R \qquad (46)$$

When the Jacobian, $\underline{\bar{J}}$, becomes singular at an extremum, there exists an infinity of vectors, $\underline{dx}$, which will satisfy equation 46 when the redundant equations are ignored. In this section, a technique will be described by which equation 46, with a singular Jacobian, can be solved for some solution, $\underline{dx}$ by ignoring the redundant equations. The details of this method are described in Appendix B.

The technique is a variation of the Doolittle method which in turn is a variation on the Crout reduction [11]. Doolittle's method states that given a system of equations;

$$\underline{\bar{A}x} = \underline{b} \tag{47}$$

where $\underline{\bar{A}}$ is non-singular, the matrix $\underline{\bar{A}}$ can be written;

$$\underline{A} = \underline{\bar{L}}\ \underline{\bar{U}} \tag{48}$$

where $\underline{\bar{L}}$ is lower triangular with unity on the diagonal and zero above the diagonal. Matrix $\underline{\bar{U}}$ is upper triangular with zeros below the diagonal and non-zero elements on the diagonal. The matrices $\underline{\bar{L}}$ and $\underline{\bar{U}}$ can be chosen in such a way that;

$$\underline{\bar{L}}\ \underline{y} = \underline{b} \tag{49}$$

$$\underline{\bar{U}}\ \underline{x} = \underline{y} \tag{50}$$

Equation 49 can be solved for the vector, $\underline{y}$. Then equation 50 is solved for the solution vector, $\underline{x}$.

The modified Doolittle's method produces an $\underline{\bar{L}}$ and a $\underline{\bar{U}}$ which will satisfy equations 48-50 when the matrix $\underline{\bar{A}}$ is non-singular. When $\underline{\bar{A}}$ is singular, the structure of matrix $\underline{\bar{U}}$ is different.

Instead of non-zero elements on the diagonal, zero rows appear for every linearly dependent equation and zero columns for every unconstrained variable. Equation 48 is no longer true if matrix $\underline{\bar{A}}$ is singular, but equations 49 and 50 remain valid.

When equation 49 is solved, a unique vector $\underline{y}$ will be determined. When equation 50 is solved an n-parameter family of solutions is generated. The n-parameters are the n-unconstrained variables. If these unconstrained variables are set to zero, a solution to equation 47 will be determined. Since the solution vector, $\underline{x}$, is simply the increment to the solution of the non-linear equations, the unconstrained coordinates will not vary but the remaining points will be positioned correctly with respect to the independent (non-redundant) constraints. This effectively causes the redundant equation to be ignored and the unconstrained coordinate is simply constrained to remain at its original value.

This procedure has many important properties. First, Doolittle's method is one of the most accurate numerical methods for solving a system of linear equations. With finite precision arithmetic, Doolittle's method tends to minimize the round-off error.

An important feature of Doolittle's method is that not only can the numerical methods determine which variable is unconstrained but also which equations are redundant. This cannot be accomplished with the standard Gaussian elimination. This allows the DIMENSION system to indicate to the user which dimension is redundant and which points are under-constrained. This can be of value when the user is creating a dimensioning scheme.

The greatest advantage of the modified Doolittle's method is that a solution can be found which satisfies equation 46 even when the Jacobian is singular. Thus, if a shape was created with N points, normally 2N constraints would be needed to solve for the geometry. With this method, less than 2N constraints could be specified. Only the portions of the shape that are sufficiently constrained would be affected.

6.3 Iteration Modifications

The previous section dealt with methods which focused on the matrix solution to equation 46. In this section, a more global modification of the iterative technique is used. The basis of the Newton-Raphson method is the calculation of an increment, $\underline{dx}$, to the current shape estimate which will bring the geometry closer to the actual solution. The discussion here is quite separate from that of the previous section. It will not assume that the solution to Equation 46 can be found when the Jacobian is singular. It is assumed that a valid dimensioning scheme is used.

If the dimensioning scheme is truely valid (not redundantly dimensioned) there are methods by which the part geometry can be made to converge from its present dimensional values to the desired dimensional values. These methods fall into the classes of stepping, relaxation, and curve-crawling.

6.3.1 Stepping techniques

Stepping techniques simply cause the numeric values of the explicit dimensions to be incremented in steps from the present values to the desired values. With each increment, a new initial guess is determined which is very close to the solution of the next increment. This is different from the successive guesses made in each iteration of the Newton-Raphson method.

For example, a dimensional value is changed from 10.0 to 15.0 in five steps. This equivalent to altering the dimension from 10.0 to 11.0, 11.0 to 12.0, 12.0 to 13.0 and so on. At each step, Newton's method is executed and the iteration is allowed to converge. In order to reduce the amount of computation, the solutions in the intermediate steps do not need to be as accurate as the final step.

6.3.2 Relaxation techniques

Relaxation techniques are used in Newton's method to stabilize the convergence process. In the conventional Newton-Raphson iteration the increment vector, $\underline{dx}$, is derived from the Jacobian and the residual as shown in Equation 51.

$$\underline{dx} = \underline{J}^{-1}\overline{\underline{R}} \qquad (51)$$

Relaxation simply involves incrementing the coordinate values by scalar multiple, F, of the increment vector $\underline{dx}$ as in equation 52.

$$\underline{X}_{n+1} = \underline{X}_n + F \cdot \underline{dx} \qquad (52)$$

The value of F, the relaxation factor, is usually;

$$0 < F < 1.0$$

to make the convergence slower, and hopefully more stable. If $F > 1.0$ then the convergence will be faster yet less stable.

6.3.3 Curve-crawling techniques

The method of curve-crawling is similar to that of the relaxation factor. In this method the multiplicative factor is not a scalar but a function of the magnitude of the increment, $\underline{dx}$. The choice of the function determines the convergence properties. A useful function of $\underline{dx}$ would be one where the value F approaches 1.0 as the magnitude, $|\underline{dx}|$, approaches zero. Equation 53 was used in the DIMENSION system and resulted in acceptable performance.

$$F = 1 - \frac{3}{4}e^{-0.1/|\underline{dx}|} \qquad (53)$$

Curve-crawling slows the convergence considerably as shown graphically in Figure 23. In this figure, equation 53 is used to modify the convergence for the solution to a single function. In Figure 24, the same function is used but with no relaxation techniques.

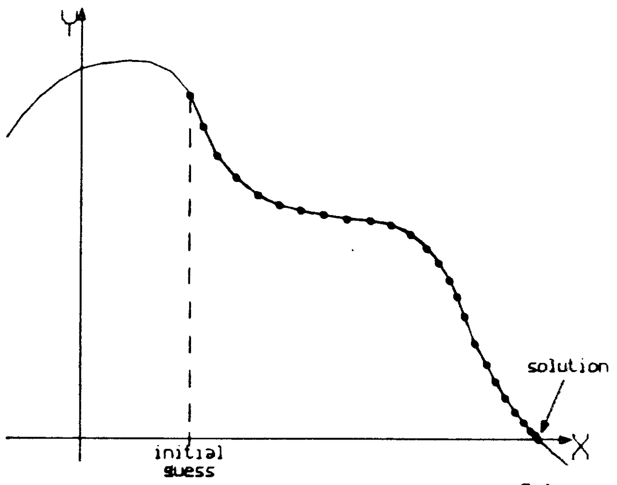


Figure 23, Convergence using curve-crawling techniques.

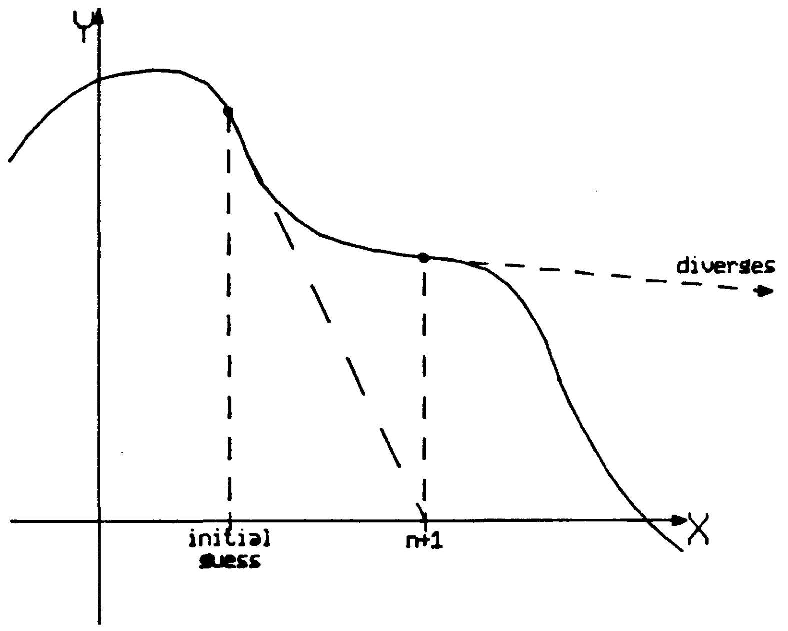


Figure 24, Convergence using standard iteration.

One problem with the use of curve-crawling is that if an extremum exists along the curve, it is likely that the curve-crawling iteration will find it. If the iteration encounters an extremum in which the displacement vector, dx, cannot be calculated it is worthwhile to simply guess a new initial guess in hopes that it will put the iteration past the extremum point. A procedure which was used in an earlier version of the DIMENSION system was to use the previously computed displacements to compute a new initial guess if an extremum is found (ie.-when the Jacobian is singular). This usually pushes the iteration past the extremum to where a new increment vector, dx, can be computed in the normal fashion.

The current method is to use the solution generated by the modified Doolittle's method discussed in the previous section. When this solution is used it is possible to push the iteration past the extremum.

6.4 Segmenting the Jacobian

One of the potential problems associated with symbolic dimensioning is that the size of the Jacobian increases by the square of the number of points involved. For example, if a two-dimensional shape contained N points, 2N equations would be needed to constrain the geometry and the Jacobian would contain $4N^2$ elements. In three dimensions, 3N equations would be required and the Jacobian would have $9N^2$ elements. In order to solve the system of constraint equations for the geometry, all the equations needed to be included in the iteration. The purpose of this discussion is to describe, somewhat pragmatically, an algorithm by which a minimal set of equations and unknowns may be selected to be included in the iteration.

To facilitate the discussion, an example will be used. Given the part shape in Figure 25, a set of constraint equations can be defined (equations 54-65).

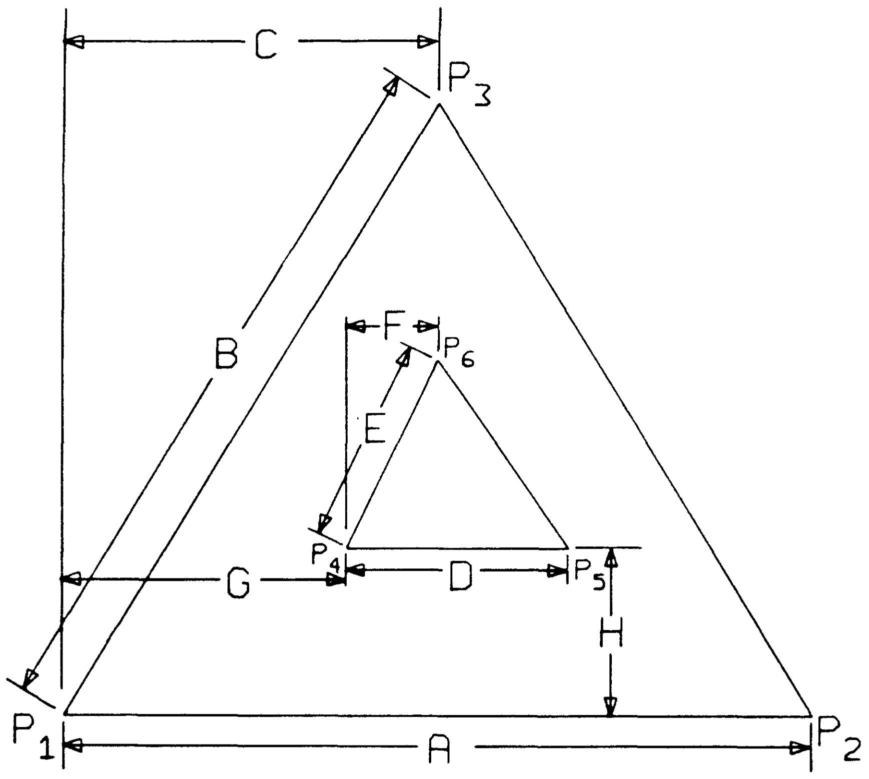


Figure 25, Example shape.

Constraint Number	Constraint Type	Constraint Equation	
1	Horz. Dist.	$X_2 - X_1 - A = 0$	(54)
2	Lin. Dist.	$(X_3-X_1)^2 + (Y_3-Y_1)^2 - B^2 = 0$	(55)
3	Horz. Dist.	$X_3 - X_1 - C = 0$	(56)
4	Horz. Dist.	$X_5 - X_4 - D = 0$	(57)
5	Lin. Dist.	$(X_6-X_4)^2 + (Y_6-Y_4)^2 - E^2 = 0$	(58)
6	Horz. Dist.	$X_6 - X_4 - F = 0$	(59)
7	Horz. Dist.	$X_4 - X_1 - G = 0$	(60)
8	Vert. Dist.	$Y_4 - Y_1 - H = 0$	(61)
9	Horz. Line	$Y_2 - Y_1 = 0$	(62)
10	Horz. Line	$Y_5 - Y_4 = 0$	(63)
11	Origin	$X_1 = 0$	(64)
12	Origin	$X_2 = 0$	(65)

If dimension G is altered, the only coordinates that will be affected will be X_4, X_5 and X_6. In order to solve for these coordinates, constraints D, F and G must be satisfied. A general approach to the selection of this set of equations and unknowns is desired.

The first step in this process is to determine which coordinates are sensitive to a given constraint. This can be accomplished by simply solving equation 66 for the vector, $\underline{S}_i$.

$$\underline{\bar{J}}\ \underline{S}_i = \underline{R} \tag{66}$$

where; $R_j = 0$ for $j \neq i$

$R_j = 1$ for $j = i$

Any non-zero element of $\underline{S}_i$ is sensitive to a variation in the i^{th} equation. If this procedure is used for all the equations, the "coordinate sensitivity" matrix, $\underline{\overline{Sc}}$, can be generated.

$$\underline{\overline{Sc}} = (\underline{S}_1, \underline{S}_2, \ldots\ \underline{S}_n) \tag{67}$$

where the $\underline{S}_i$ are the columns of $\underline{\overline{Sc}}$.

The coordinate sensitivity matrix indicates which variables are sensitive to any given constraint equation. If the element $Sc_{ij} = 1$ then the ith coordinate is sensitive to the jth equation. Since the only information needed is whether or not the coordinate is sensitive to a certain equation, the elements in the coordinate sensitivity matrix need only be zero or non-zero.

If all non-zero elements are set to unity, the coordinate sensitivity matrix can be computed for the example in Figure 25.

$$S_c = \begin{bmatrix} 0&0&0&0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0&0&0&0&1 \\ 1&0&0&0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0&1&0&0&1 \\ 0&0&1&0&0&0&0&0&0&0&1&0 \\ 0&1&1&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&1&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1&0&0&0&1 \\ 0&0&0&1&0&0&1&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1&0&0&0&1 \\ 0&0&0&0&0&1&1&0&0&0&1&0 \\ 0&0&0&0&1&1&0&1&0&0&0&1 \end{bmatrix} \qquad (68)$$

A second matrix which needs to be generated is the "dimensional sensitivity" matrix, $\overline{\underline{S}}_D$. This matrix indicates which dimensions are sensitive to which coordinates. If the element $S_{D_{ij}} = 1$, then the j th dimension is sensitive to the i th coordinate. The dimensional sensitivity matrix is analogous to the coordinate sensitivity matrix. Two conditions must be satisfied in order for the j th generalized dimension to be sensitive to the i th coordinate. First, the i th coordinate must be sensitive to the j th equation as indicated by the coordinate sensitivity matrix (ie.-$Sc_{ij} = 1$). Secondly, the i th coordinate must appear explicitly in the j th constraint equation.

The dimensional sensitivity matrix, $\overline{\underline{S}}_D$, can be generated directly from the coordinate sensitivity matrix, $\overline{\underline{Sc}}$, by setting all elements $Sc_{ij} = 0$ when the i th coordinate does not appear explicitly in the j th equation. The dimensional sensitivity matrix for the example shape is;

$$S_D = \begin{bmatrix} 0&0&0&0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0&0&0&0&1 \\ 1&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0&0&0 \end{bmatrix} \qquad (69)$$

The algorithm by which the set of equations and unknowns can be determined is as follows;

1) Given that the j^{th} dimensional constraint is altered, choose as unknowns, any variables, i, where $Sc_{ij} = 1$.

2) Choose as equations any dimension, j, where $S_{D_{ij}} = 1$, for all unknowns, i, which were determined in step 1. If number of equations equals number of unknowns, then stop.

3) Scan down the column, j, of all equations chosen in step 2. If $S_{D_{ij}} = 1$ for any row, i, which is not included in the set of unknowns, then include the unknown and repeat step 2.

This procedure will determine a subset of the total number of equations and unknowns, which need to be solved to compute the new geometry.

Returning to the example, dimension G, the 7th constraint, will be altered. Column 7 of the coordinate sensitivity matrix, equation 68, must be scanned for non-zero elements. Rows 7, 9, and 11 are chosen. This corresponds to coordinates X_4, X_5 and X_6. In order to determine the set of equations, the dimensional sensitivity matrix, equation 69, is analyzed as in step 2. Rows 7, 9, and 11 are scanned for non-zero elements. In this case the columns 4,6, and 7 have non-zero elements in rows 7, 9, or 11. Since the number of unknowns equals the number of equations, the procedure is complete.

The only discrepancy which can occur is that the number of unknowns is less than the number of equations. If this occurs, additional unknowns must be found by executing step 3 and then reprocessing step 2.

The use of this procedure greatly reduces the number of calculations required to solve the matrix equation 46 due to the fact that there are fewer equations. This will result in a reduction in the round-off errors associated with those calculations. The coordinate sensitivity matrix requires approximately 4/3 N^3 operations for an NXN matrix. This is four times the number of calculations associated with a single iteration of the Newton-Raphson method. The reduction lies in the fact that usually, the sensitivity matricies $\underline{\overline{S}_D}$ and $\underline{\overline{Sc}}$ need to be computed once for each set of constraint equations since the zero/non-zero nature of $\underline{\overline{S}_D}$ and $\underline{\overline{Sc}}$ tends to be invariant with a change in geometry.

It is possible that this procedure would not choose the correct set of dimensions and unknowns. The shape in Figure 26 illustrates this problem.

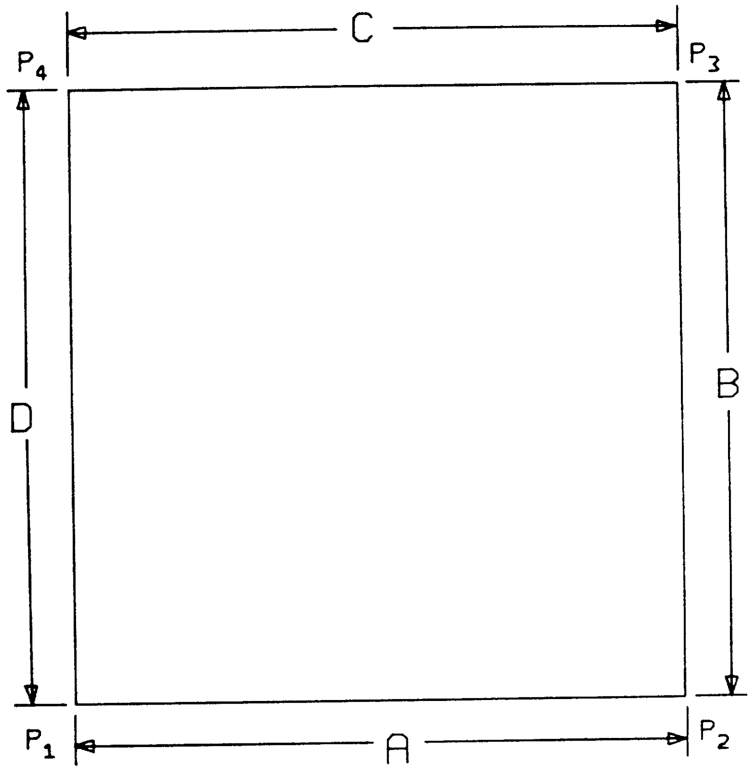


Figure 26, A cube.

Dimensions A and B are horizontal and vertical dimensions. Dimensions C and D are linear dimensions. Line $\overline{P_1P_2}$ is constrained to be horizontal and line $\overline{P_2P_3}$ is constrained to be vertical. As before, $X_1=0$ and $Y_1=0$. The sensitivity matricies would show that Y_4 is not sensitive to dimension C, and that X_3 is not sensitive to dimension D. From Figure 26, it can be seen that for a small variation in dimension C, the vertical position of point 3 will not change.

Therefore, if dimension C was altered, steps 1-3 above would select only equation C along with coordiate X_3 to be solved in the iteration. After solving equation C for the new value for X_3, it would be determined that the residual in equation D is no longer zero. This would indicate that the correct equations and unknowns were not chosen. At this point, the sensitivity matricies must be recomputed and the total process repeats.

Because of this potential hazard, two more steps must be added to the process;

4) Perform the Newton-Raphson iteration using the equations and unknowns found in steps 1-3.

5) If the residuals for all the equations are not small after the iteration completes, recompute $\bar{S}_C$ and $\bar{S}_D$ and go to Step 1. Note- the dimensional constraint(s) being altered in step 1 is now the equation(s) whose residual is non-zero.

The author must admit that no underlying mathematical theory has been found which proves that this algorithm works in all cases, nor that it produces the minimal set of equations and unknowns. All that can be stated is that the procedure has been incorporated in the DIMENSION system and the results look very promising.

7.0 IMPLEMENTATION

The previous section outlines the basic extentions of Newton's method which improve the convergence properties and capabilities of the numerical methods. In addition, a procedure was described which allowed for the definition of higher order graphic entities and general geometric constraints. In section 1.0, a qualitative description was made of the "design process" in order to lay a foundation for the conceptual development of the DIMENSION system. In this section, the realization of a CAD system utilizing these mathematical and conceptual techniques will be discussed. A user's manual for the DIMENSION system is incorporated as Appendix D and is intended to be a self-contained document. Although some information appears both in Appendix D and in this section, a more detailed treatment of the important topics is presented here.

7.1 Graphic Input Procedures

The purpose for the graphic input routine is to provide a description of the basic topology of the shape. This includes the approximate position and orientation of the lines, arcs and circles which comprise the shape contours. The position of each point is used as the initial guess to the Newton-Raphson iteration. Since this initial guess need not be accurate, the graphic input may take the form of a free-hand sketch.

The data which is derived from the graphic input routine is of two types;

1) X,Y coordinates of all points that will be used in the Newton-Raphson method.
2) Indicies of points which are defining the specific graphic entities;
    - endpoints of lines
    - center and endpoints of arcs
    - center of circles

The sketch-like graphic input is facilitated by the use of a data tablet, digitizing pen and a tracking cross on a dynamic refresh display. The input of any shape topology is divided into "Loops". A loop is any string of successively connected graphic entities (lines and arcs) which at some point closes on itself and connects to the initial point entered. In Figure 27, the first loop is about to be completed. A loop is started by pressing and releasing the pen. This enters a point into the data base. A "rubber-band" is then connected to the point just entered and the tracking cross. When the pen is pressed and released a second time, another point is entered and the existance of a line between the last point entered and the present point is indicated in the data base.

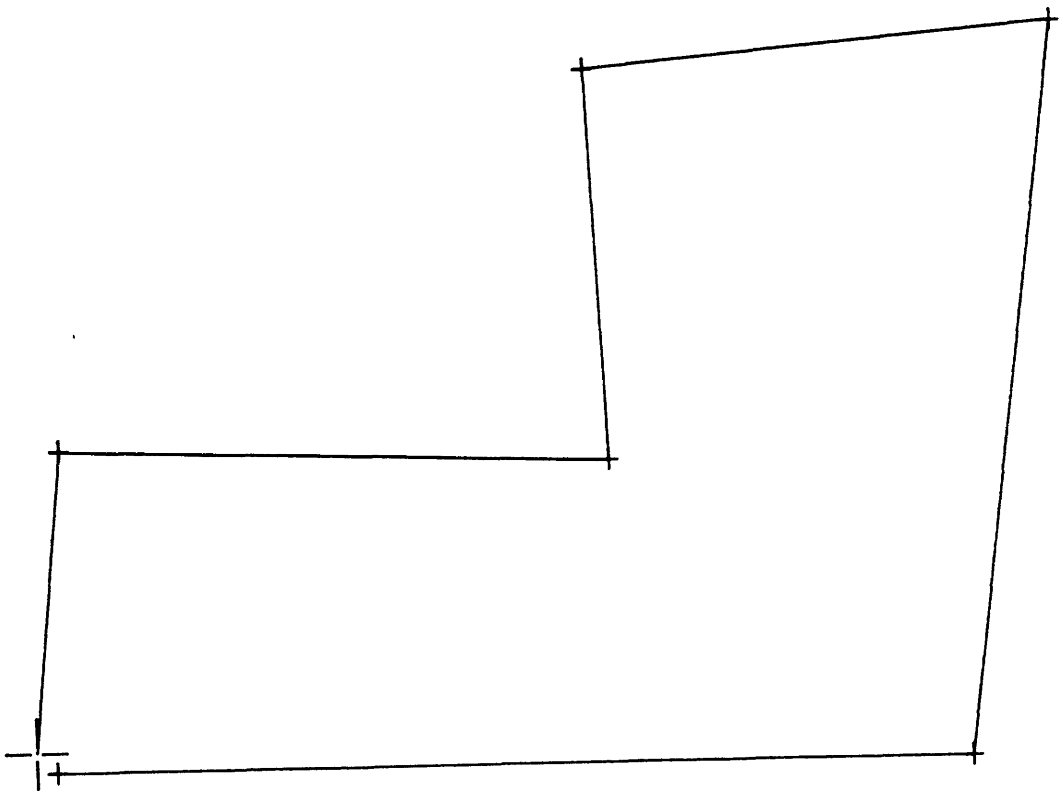


Figure 27, Completion of the first loop.

If an arc or circle is to be entered, the pen is pressed as the arc is sketched. In Figure 28, an inner loop is being specified, the arc is being drawn with the pen in the depressed position. The added "X" on the tracking cross indicates that the pen is pressed.

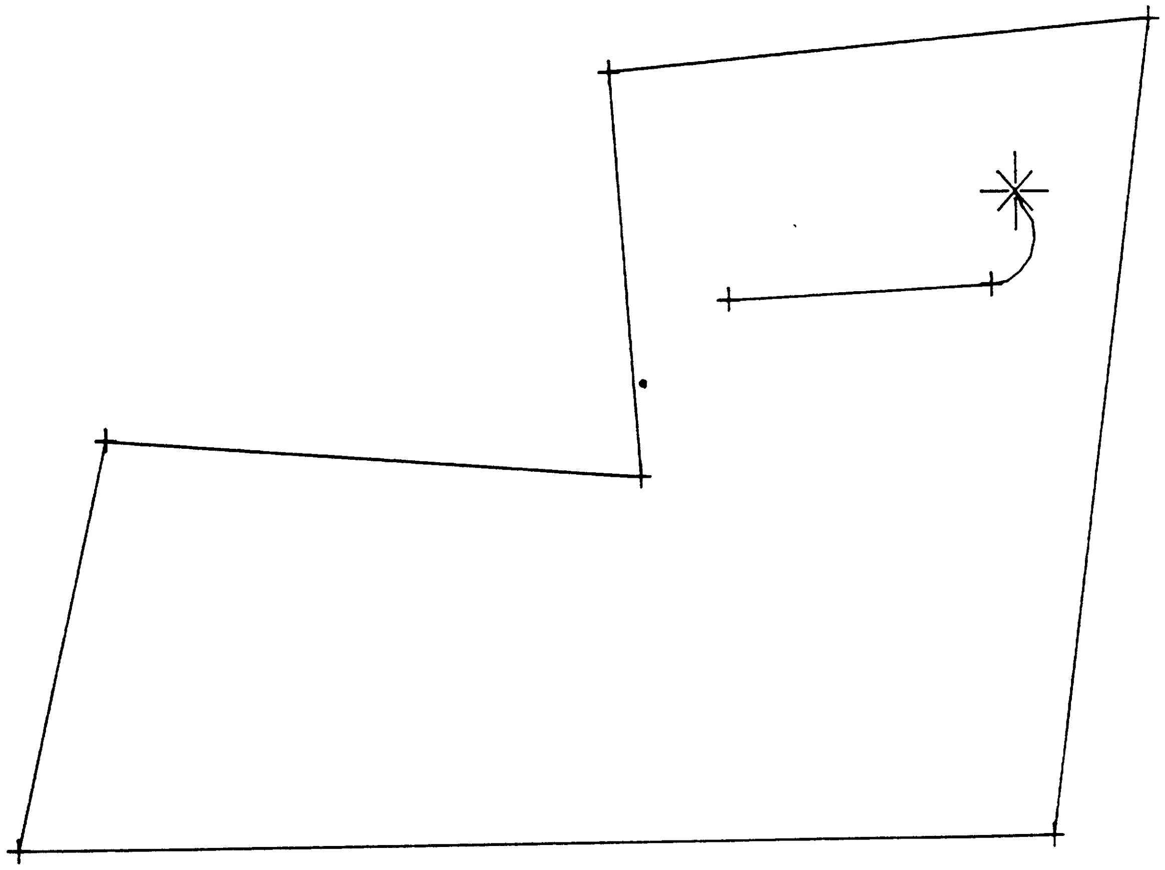


Figure 28, Sketching of an arc.

In Figure 29, two arcs have been entered and the second loop is is about to be completed. After any loop is completed, the operator can do one of three operations;

1) Start another loop.
2) Insert a single point.
3) Insert a circle.

A second loop is entered in the same manner as described above. A "single point" is entered by pressing and releasing the pen twice at the same location. A circle is entered by simultaneously pressing the pen and drawing the contour. Since a circle is defined by its center, the circle must always be drawn around a previously entered "single point".

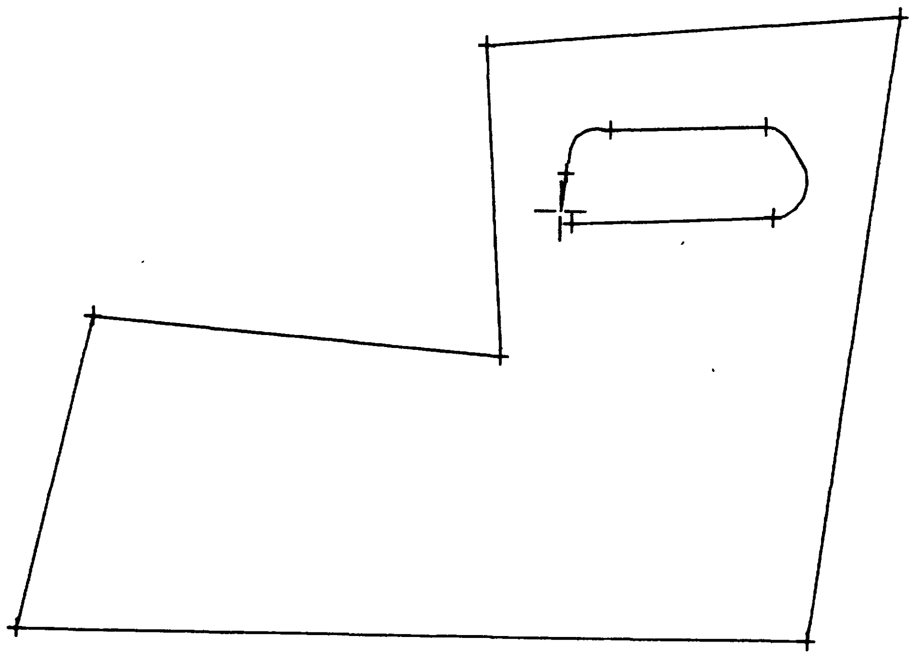


Figure 29, Completion of an inner loop.

In order to maintain a clearly defined topology in the data base, there are many validity checks made as the contours are being input. If the topology is not acceptable, the offending entity is automatically erased at time of input. This gives the operator immediate notification of a data input error. For example, the operator in Figure 30, tried to enter a circle which did not encircle a single point.

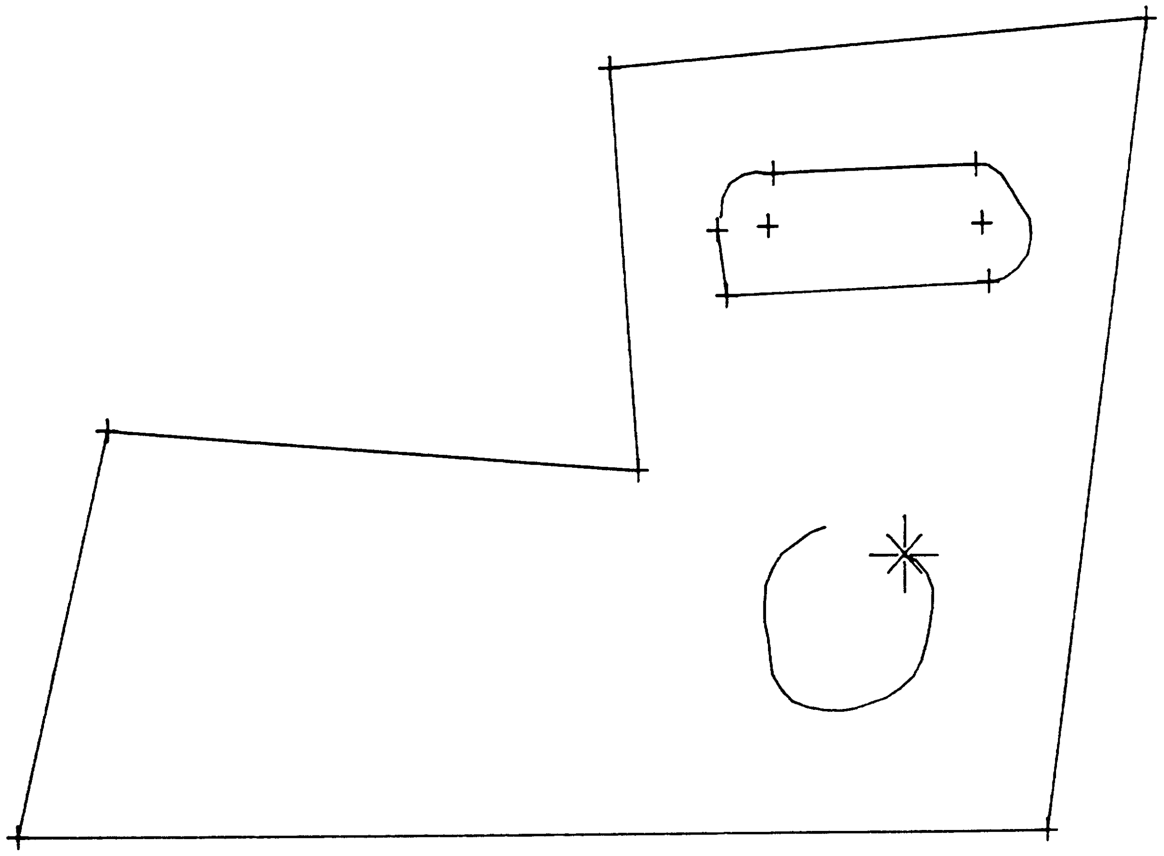


Figure 30, Incorrectly specified circle.

When the pen is lifted, signalling the end of the circle input, the circle is erased automatically, resulting in Figure 31.

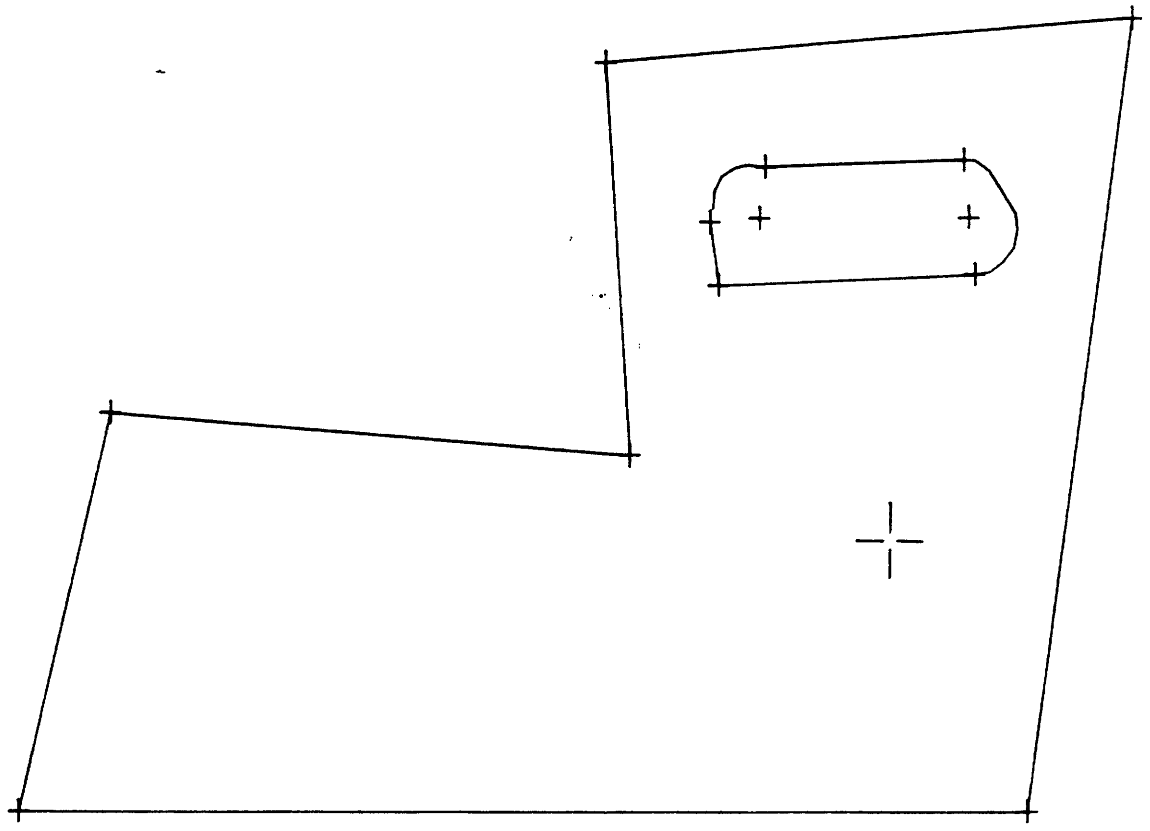


Figure 31, Notification of input error by disappearance of the graphic entity.

7.2 Definition of the Dimensioning Scheme

After the topology has been entered, control is passed to a program which allows the user to define the generalized geometric constraints. These constraints fall into three classes:

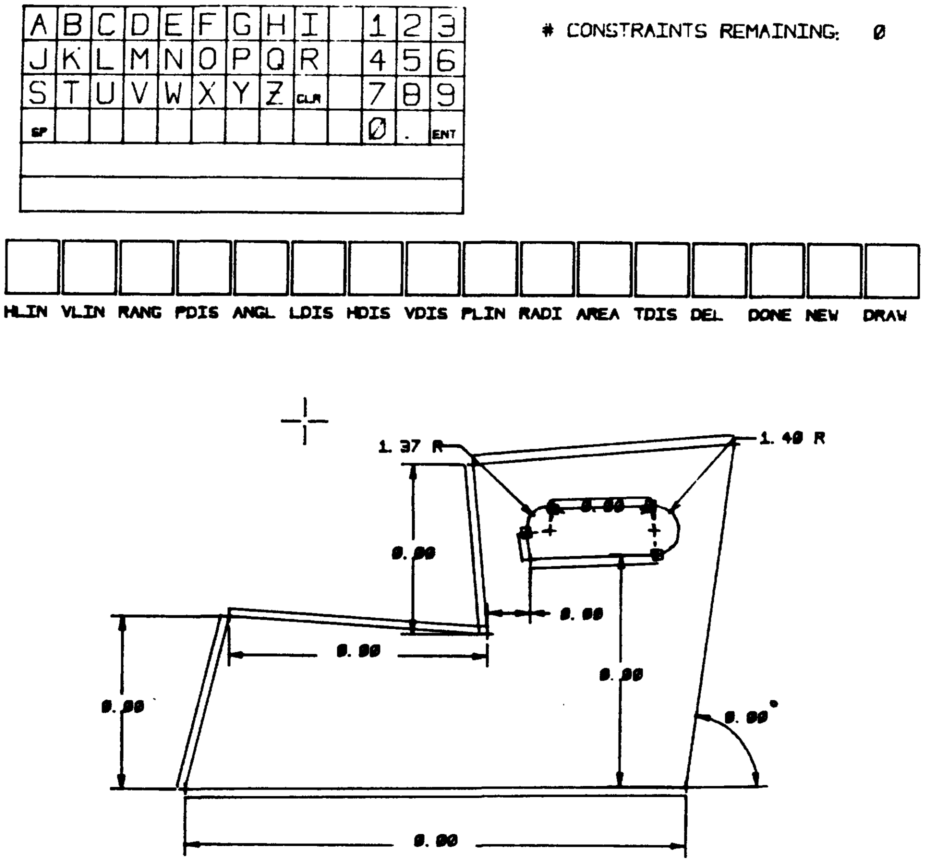
1) Explicit dimensions
2) Implicit constraints
3) Geometric constraints

As described in section 3.1, the first catagory, explicit dimensions, are the standard drafting dimensions such as angular measure, linear, horizontal or vertical displacement, or distance from a point to a line. Implicit constraints are those constraints like horizontal and vertical lines, perpendicular lines, or tangency.

The third catagory encompasses the general geometric constraints which cannot be defined as "dimensions". These include constraints on the surface area, moment of inertia, or center of mass, etc.

In all cases, the operator is allowed to specify the constraints with an interactive, semi-automatic dimensioning protocol. This protocol is described in detail in Appendix D. In Figure 32, the part shape has been fully constrained.

Note, the double lines indicate horizontal and vertical implicit constraints and the boxes indicate points of tangency. The user is always notified as to the number of constraints which remain to be defined. In Figure 32, the message in the upper right corner indicates that no more constraints are needed. This informs the user that the part has the correct number of dimensions but it does not determine if the part is redundantly dimensioned. This will occur when the user executes the Newton-Raphson method in the next routine.



A B C D E F G H I 1 2 3
J K L M N O P Q R 4 5 6
S T U V W X Y Z CLR 7 8 9
SP Ø . ENT
# CONSTRAINTS REMAINING: Ø
HLIN VLIN RANG PDIS ANGL LDIS HDIS VDIS PLIN RADI AREA TDIS DEL DONE NEW DRAW
1.37 R
1.40 R
0.00
0.00
0.00
0.00
0.00
0.00
0.00°
0.00
0.00

Figure 32, A dimensioning scheme.

7.3 Changing the Values of the Dimensions

After the shape has been sketched and the geometric constraints defined, control is passed to another procedure where the user may execute the Newton-Raphson iteration (by pressing "GO"). The sketched input geometry is used as an initial guess and the dimensioning scheme is used to compute the Jacobian and the residuals. The part shape is essentially "straightened up" by this operation as in Figure 33. The values of the dimensions result from simply measuring the sketched geometry.

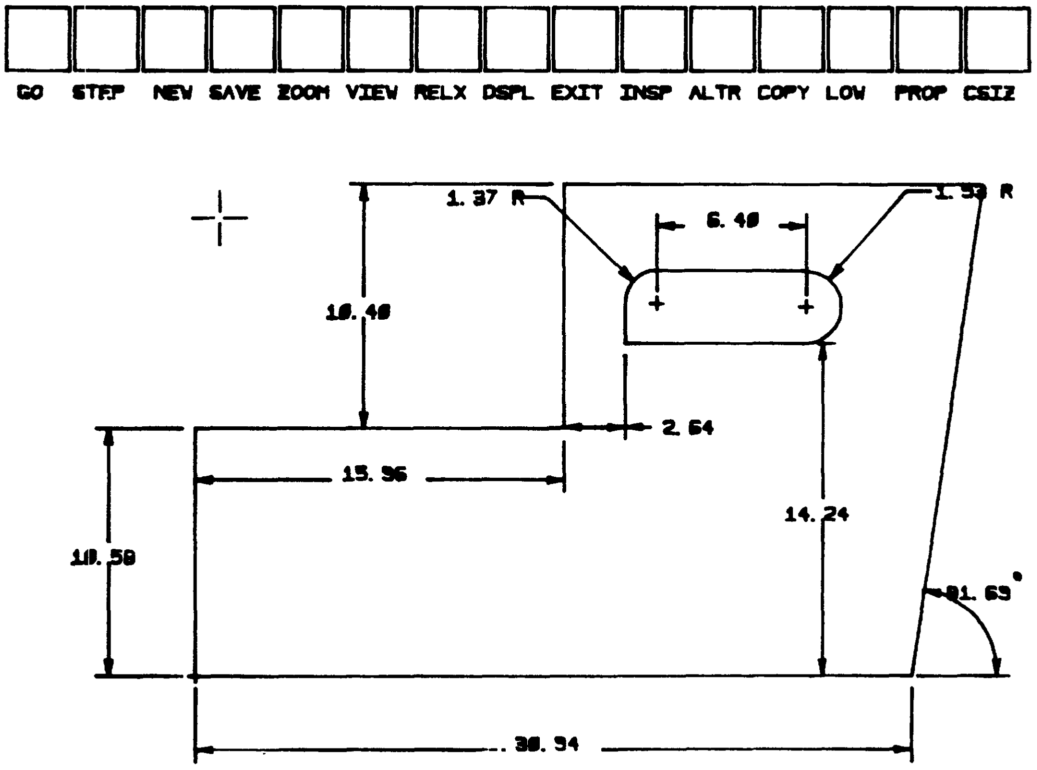


Figure 33, Geometry which has been "straightened".

The user may then change the values of the dimensions by simply indicating, with the tracking cross, the dimension to be changed as shown in Figure 34. The desired numeric value, 15.0, is then entered. The inputting of alphanumerics is accomplished with the use of the interactive keypad in the upper left corner. Upon executing the Newton-Raphson method, the part geometry is updated to conform to the change in the explicit dimension. The resultant shape is shown in Figure 35.

As a result of many such dimensional changes, the part geometry can be modified to the designer's needs. The part shape in Figure 36 is the same part with a different set of dimensional values.

The problem of under- or redundant dimensioning is handled in this program in two modes. The first mode is simply to notify the user which constraints are redundant and which coordinates are under-constrained. This occurs when the user presses "GO" and the Newton-Raphson iteration begins. In Figure 37, a shape which is redundantly dimensioned illustrates the ability of the system to notify the user of errors in the dimensioning scheme. The under-constrained points are circled and the coordinates which are under-constrained are indicated by the letter X or Y by the point. The redundant dimensions are indicated by blinking.

The second mode is through the use of the "LOW" button. This LOW gear solution technique implements the curve crawling algorithm in conjunction with the modified Doolittle's solution for the singular Jacobian in equation 46. This permits the user to create a geometry and then specify a subset of the total dimensioning scheme. The user may then control the position, via the dimensional values, of only those points which are correctly constrained.

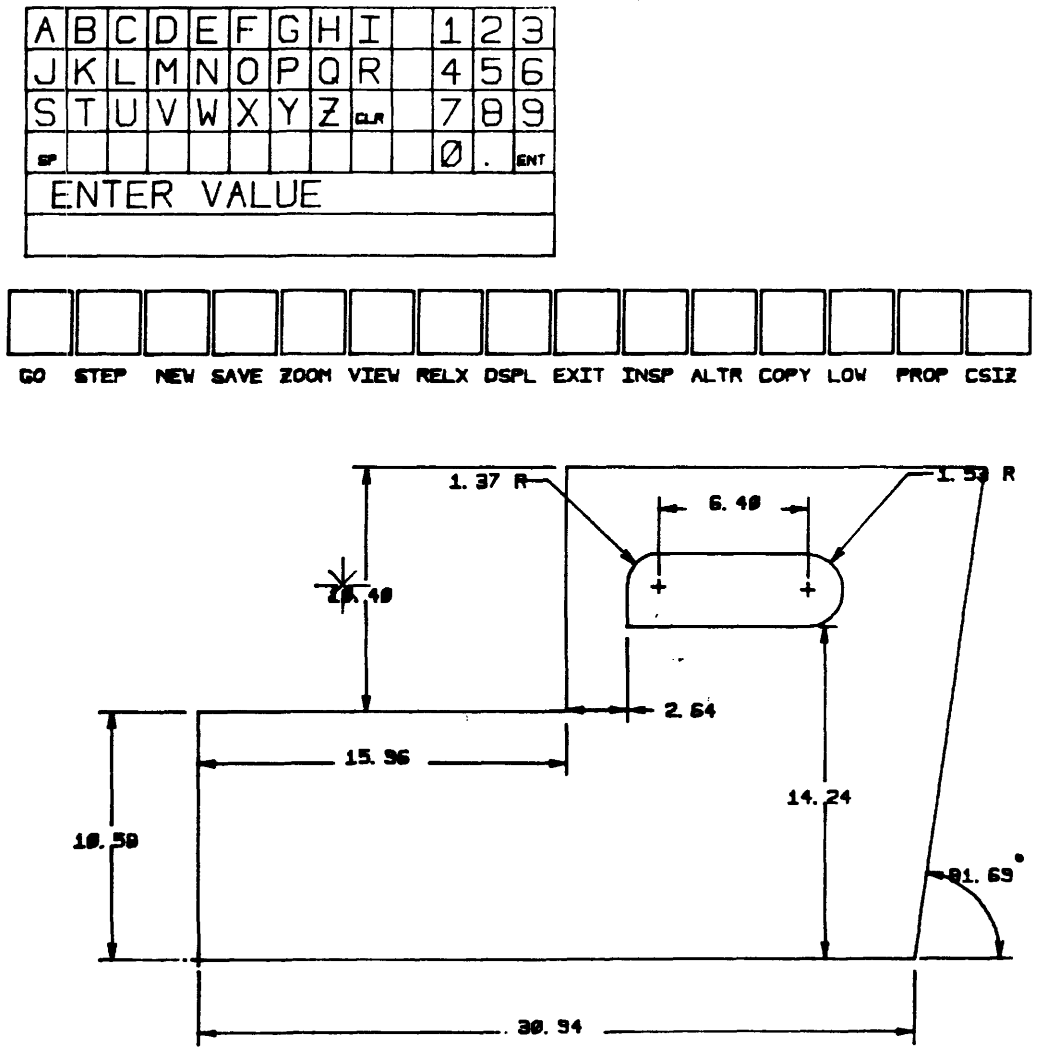


Figure 34, Dimensional value about to be changed.

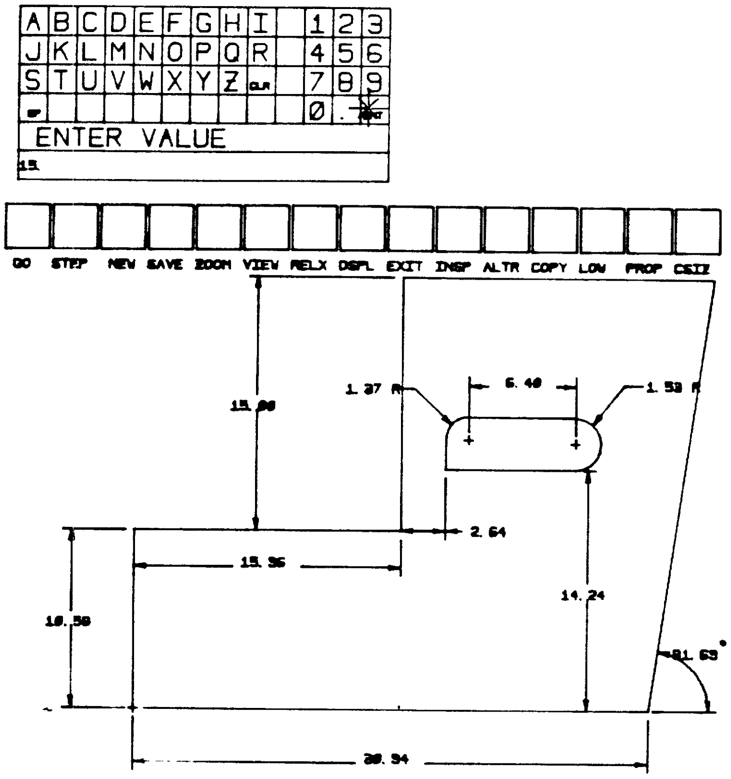


Figure 35, Result of a dimensional change.

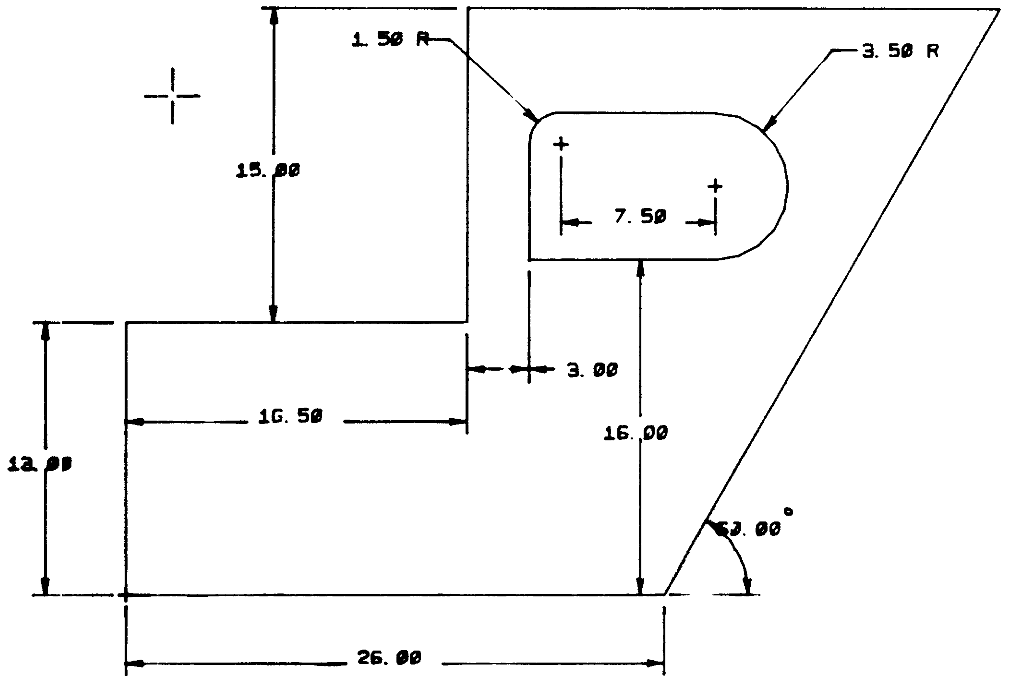


Figure 36, Result of many dimensional changes.

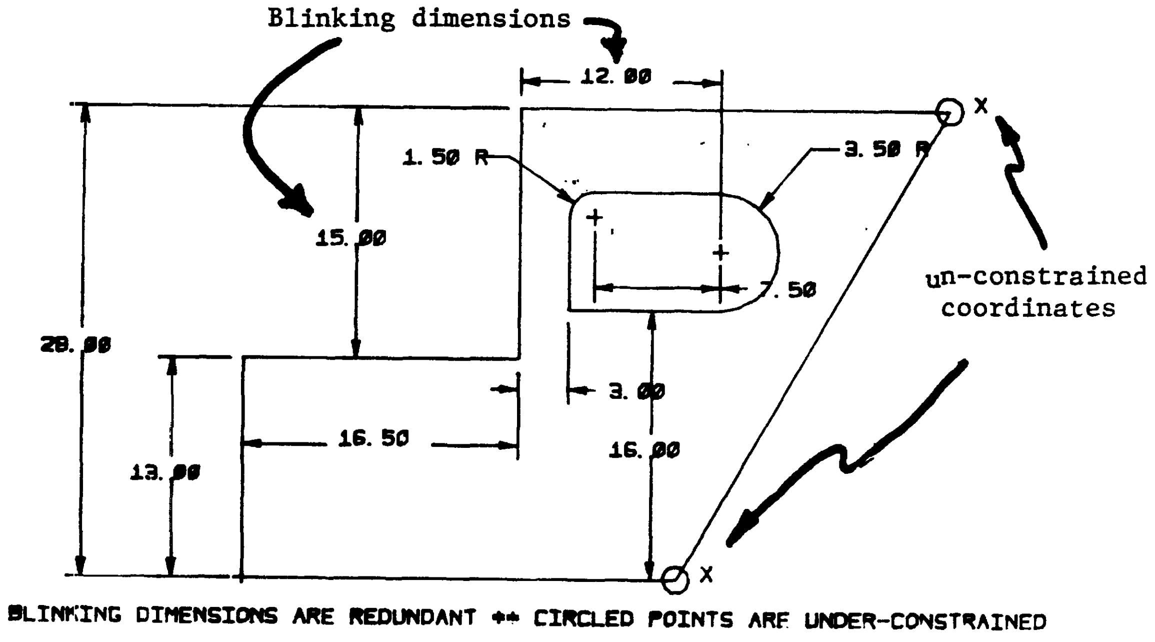

Blinking dimensions
12.00
1.50 R
3.50 R
15.00
7.50
28.00
3.00
16.50
16.00
13.00
X
X
un-constrained coordinates
BLINKING DIMENSIONS ARE REDUNDANT ** CIRCLED POINTS ARE UNDER-CONSTRAINED

Figure 37, Notification of a redundant dimensioning scheme.

7.4 Software/Hardware Configuration

The DIMENSION system was developed at the Joint Computer Facility of MIT on a VAX 11/780. The software is written entirely in Fortran and is structured as in Figure 38.

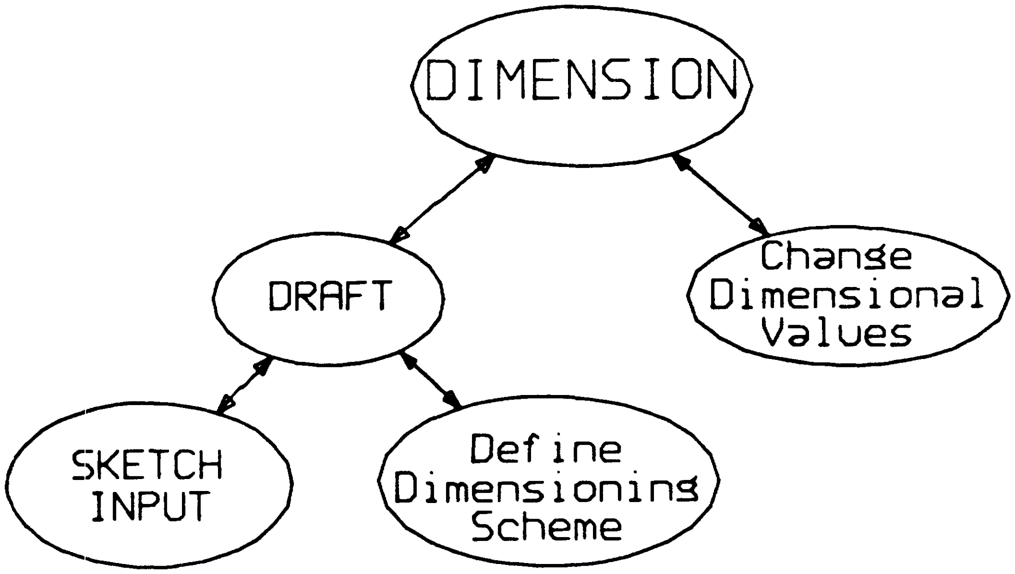


Figure 38, Basic software structure.

Although the program can be run on a single computer with a storage tube CRT, in order to use the real-time dynamic plotting capabilities of a MEGATEK 5000 driven by a PDP11/34 (see Figure 39), the software was divided into two sections. The main set of routines which perform the basic numerical calculations, were implemented on the VAX. A smaller set of programs were implemented on the PDP11.

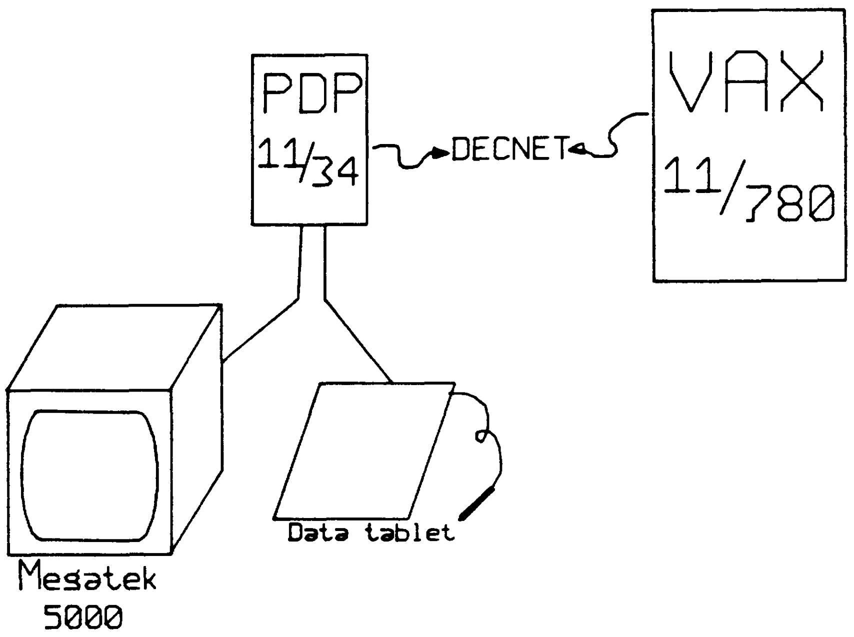


Figure 39, Hardware configuration.

With this configuration, the programs which reside on the PDP11 communicate with the programs running on the VAX via a data network (DECNET). The program which is on the PDP11 acts like an "intellegent terminal" to the VAX program. For example, if any of the DIMENSION routines, running on the VAX, require the user to indicate a point on the screen, it sends a code to the "terminal" program running on the PDP11. The terminal program then conducts the real-time task of tracking the pen with the data tablet. When the user depresses the pen, the "terminal" sends back the coordinates of the penpress. The DIMENSION programs must then decide what to do depending on the coordinates it receives from the "terminal".

In the same manner, all the interactive I/O is transmitted through DECNET. For example, if the VAX program needs alphanumeric input, the terminal program displays a graphic keypad on the MEGATEK with a prompt from the VAX (see Figure 40). The terminal program then controls the inputting of the alphanumerics with the use of the pen and data tablet.

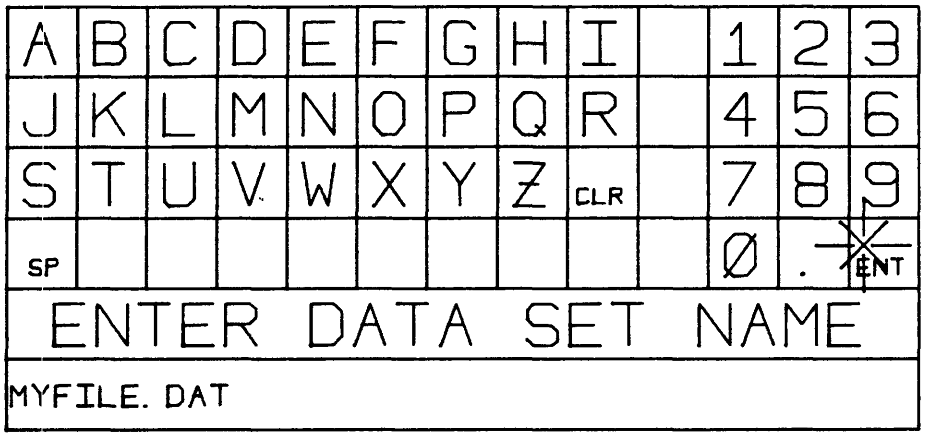


Figure 40, Interactive graphic keypad for data entry.

With this configuration, the different program modules were structured as in Figure 41. The mainline of the DIMENSION routines is program SYMDIM. Its purpose is to call the subroutines which allow the user to define and interact with the part shape. Its primary task is to execute the Newton-Raphson method. A flow chart of the Newton-Raphson method, as implemented in SYMDIM is presented in Figure 42.

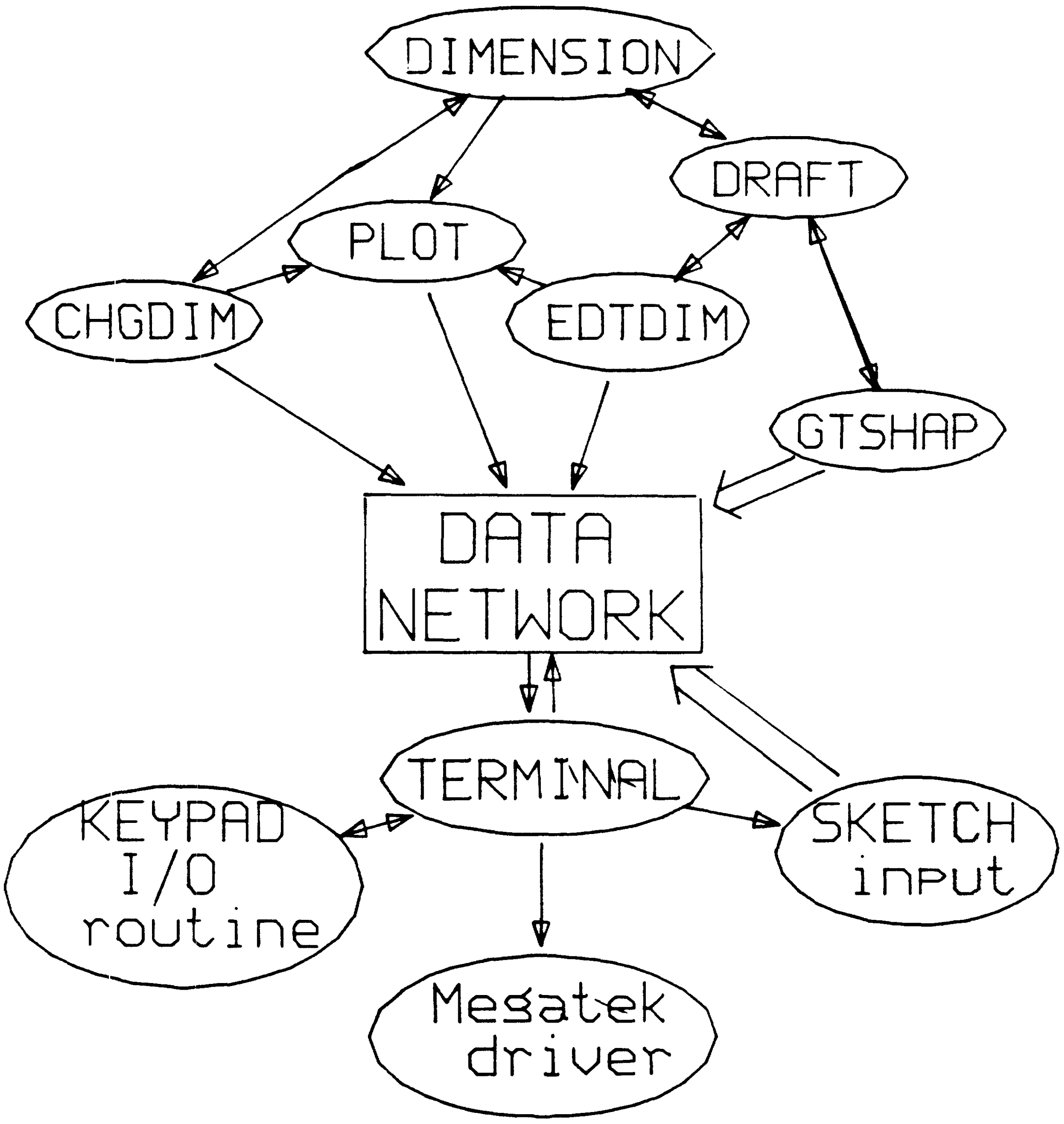


Figure 41, Configuration of program modules.

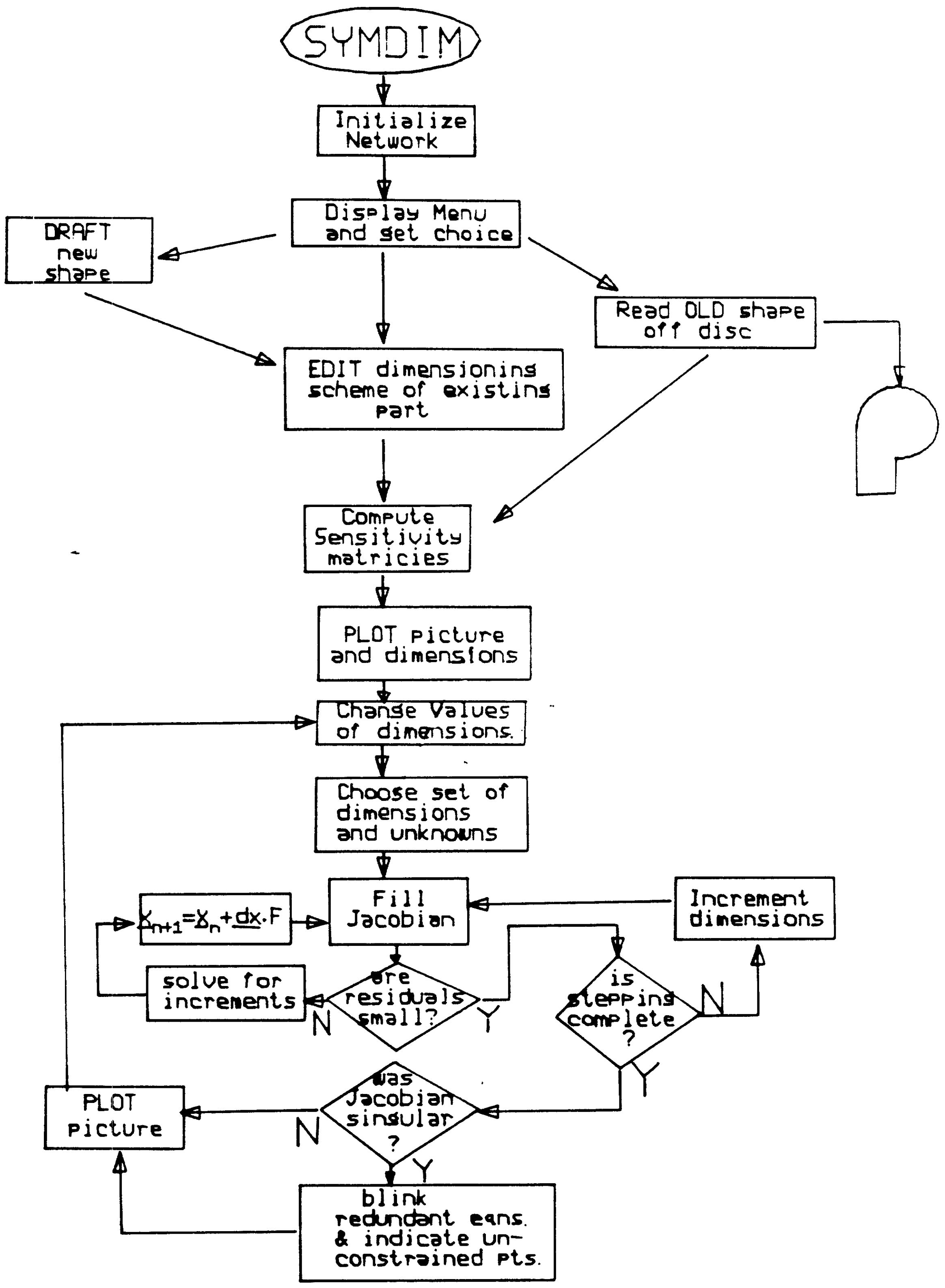

SYMDIM
Initialize Network
Display Menu and get choice
DRAFT new shape
Read OLD shape off disc
EDIT dimensioning scheme of existing part
Compute Sensitivity matricies
PLOT picture and dimensions
Change Values of dimensions
Choose set of dimensions and unknowns
$X_{n+1}=X_n+dx \cdot F$
Fill Jacobian
Increment dimensions
solve for increments
are residuals small?
N
Y
is stepping complete ?
N
Y
was Jacobian singular ?
N
Y
PLOT picture
blink redundant eqns. & indicate unconstrained pts.

Figure 42, Flow chart of mainline - SYMDIM

The module, CHGDIM, is used to permit the user to change the numeric values of the explicit dimensions. The user is allowed to simply point, with the tracking cross, to the number that is to be changed and then enter the desired value using the graphic graphic keypad. In addition, the user may input the values of the relaxation factor, number of steps in the iteration, and alter the viewing scale.

When a user wishes to enter a new shape or edit the dimensioning scheme of an existing shape, control moves to the DRAFT subroutines. If the user wishes to start from scratch, the GTSHAP (get shape) routine is called and a message is sent to the terminal program on the PDP11. Here the sketch input routine, EDTLIN, is called. EDTLIN analyzes the user input and sends back the data needed to define a "shape" as described in section 7.1. Control is then passed to the EDTDIM routine (on the VAX) which allows the user to interactively specify the dimensioning scheme.

Both EDTDIM and CHGDIM use a set of programs which draw the different types of dimensions. The important concept in these plotting routines is that the numeric text associated with each dimension must occupy a unique region on the screen, independent of character size and shape geometry. This is to insure that the operator can always point to a single dimension on the CRT. The "ALTR" command in CHGDIM allows the user to alter the different parameters which are used in the search techniques for positioning the dimensions.

8.0 RESULTS

The utility of a general symbolic dimensioning system has been demonstrated through the use of the DIMENSION system. An example session using the DIMENSION system is included as Appendix C. This system allows the user to sketch the contour of the shape with no concern for the actual dimensions of the part. With the present system, the shape contour may consist of two-dimensional points, lines, arcs and circles. The user then specifies the dimensional constraints and the values of the dimensions in order to modify the geometry as desired.

A method for including higher order graphic entities, such as arcs and circles, in the shape model has been implemented. It allows the user to specify common dimensions in order to constrain the arc or circle. General geometric constraints such as area were incorporated into the DIMENSION system.

The problem of numerical stability in large, complex geometries has been reduced through the use of the techniques developed in this work. The use of the modified Doolittle's method allows a singular Jacobian to be partially solved by ignoring redundant equations and unconstrained variables. The actual number of calculations can be reduced through the use of the segmentation algorithm. Given a change in dimensions, the segmentation algorithm selects a minimum number of equations and unknowns to be solved by the Newton iteration. As a result of the reduction in calculations, numerical error and computation time are also reduced.

These techniques, in conjunction with the use of the relaxation, stepping and curve-crawling techniques described in section 6.2, result in a very stable numerical iteration. For the varied set of shapes and dimensioning schemes implemented on the DIMENSION system, the mathematical techniques discussed in this paper were quite adequate.

Any interactive design system must address the problem of the human interface - the ease of use. Because the DIMENSION system emulates very natural shape construction and specification techniques, users have found the system easy to use and to understand. The fact that the system informs the user of any problem in the dimensioning scheme is a major advantage of this approach. It eliminates the problem of producing designs which are over-, under- or redundantly dimensioned. More importantly, the use of symbolic dimensioning gives the designer a better "feel" for the effect of a dimension on the part geometry by producing visual feedback for any change in dimensions.

9.0 RECOMENDATIONS AND CONCLUSIONS

The DIMENSION system has demonstrated some of the features of using symbolic dimensioning in a CAD system. This system emulates the design process by allowing the user to sketch the initial shape. More importantly, it facilitates the process of design modification by variation of the dimensional constraints. In addition, design synthesis can be accomplished through the use of the geometric constraints. This permits the designer to omit dimensional constraints and specify constraints such as mass, inertia or center of mass in order to obtain the desired geometry.

The application of the numerical techniques developed in this research effort can be applied to the case of a 3-D shape model. The dimensional equations must be defined between points in three-space and implied constraints must be identified. The problem of graphic input in three-space must also be studied in the context of eventually specifying 3-D constraints and geometry in an interactive manner.

The ability to alter the geometry of a part by varying the numeric values of the dimensional constraints, permits direct analysis of a tolerancing scheme. The effect of a single tolerance, or the complete set of tolerances, on the total shape can be directly evaluated. This would be valuable in the analysis of assemblies. By using the same mathematical techniques, one could specify the nominal dimension and the clearances between mating parts, the system could then compute the maximum variation in dimensions (tolerances) that would maintain the specified clearances. This would give the designer a quantitative basis for specifying tolerances.

An important application of the techniques developed in this paper would be to the process of converting a design which is in the form of a drawing to a computer data base. These techniques would produce a very efficient procedure for converting a very large "paper" data base to a computer data base. In practice, much of the sketching and dimension specification could be done automatically through the use of feature recognition.

The application of the DIMENSION system to the problem of associativity of dimensions in assemblies should also be studied. In this application, the value of one or more dimensions in one part of an assembly is related in some way to the dimensions in other assemblies. When a dimension is altered in any part, the associativity is carried through the entire assembly. The Newton-Raphson method could be used to find the set of dimensional values which is a solution to a set of constraining equations representing the associativity between the dimensions. For example, assume that the value of a dimension in part A is a function of a dimension in part B and also of the mass (area) of part C. The dimension in B and the area in C could in turn be related to some other property or dimension of other parts in the assembly. If the designer altered any dimension that changed the area in C or the dimension in B, the set of constraints for the values of the dimensions would be solved to produce a new set of dimensional values and in turn, a new part geometry. This would represent a form of design optimization based on a set of equations which may be kinematical, thermodynamical or mechanical in nature.

One drawback to the present system is that memory requirements increase by the square of the number of points in the shape. The reason is that the Jacobian is a 2N x 2N matrix (N = number of 2-D points). In this application, the Jacobian has two important features. First, it is a sparse matrix. In the present system, even if the matrix is 2N x 2N, in most cases, the matrix is 90% zeros. Secondly, the position of all the non-zero elements are known. The core requirements for the system could be reduced by not requiring the storage of all the zero locations in the Jacobian. In addition, the computation time would be reduced by not requiring the multiplications by zero.

In summary, the use of the procedures developed in this research effort has resulted in a new class of CAD system. The DIMENSION system eliminates the need for precise geometric input and provides a means for checking the validity of a dimensioning scheme. In addition, it enables the computer to assume more of the role in geometric analysis in the areas of dimensioning, tolerancing and shape representation. It offers a natural, efficient mechanism for shape definition and modification with potential development in the areas of 3-D representation, design synthesis, tolerance analysis and associativity in assemblies.

It is the author's firm belief that the mathematical and procedural techniques demonstrated in the DIMENSION system will advance the state of the art in the areas of geometric input, dimensioning, tolerancing and design modification and result in a major advancement in the field of computer-aided design.

REFERENCES

1. Computervision Corp., CADDS 3 - Parametric Element Processor Manual, 1979.

2. Gopin, A. M., "Development of a Dimension Dependent Data Structure for Two-Dimensional Computer Graphics", Unpublished Master's Thesis, M.I.T., 1978.

3. Gossard, D. C., "Symbolic Dimensioning", Unpublished Proposal, M.I.T., 1979.

4. Hildebrand, F. B. , Methods of Applied Mathematics, Prentice-Hall, N.J., 1965.

5. Hillyard, R. C., "Dimensions and Tolerances in Shape Design", Technical Report No. 8, University of Cambridge, U.K., 1978.

6. Hillyard, R. C., Braid, I. C., "Analysis of Dimensions Tolerances in Computer-Aided mechanical Design", Computer-Aided Design, Vol. 10, No. 3, P161-166, May 1978.

7. Hillyard, R. C., Braid, I. C., "Characterizing Non-Ideal Shapes in Terms of Dimensions and Tolerances", Computer Graphics, pp 234-238, Feb. 1979.

8. Katzman, H., Fortran 77, Reinhold, N. Y., 1978.

9. Light, R. A., "A Dimension Based, 2-D Shape Editing Feature for Computer-Aided Drafting Systems", Bachelor of Science Thesis, M.I.T., May, 1979.

10. Lin, V. C., "Review of the Symbolic Dimensioning System", Internal Document, MIT, December, 1979.

11. Ralston, A., Wilf, H. S., Mathematical Methods for Digital Computers, Vol. 2, Wiley, New York, 1967.

12. Requicha, A. A. G., PADL - Dimensioning and Tolerancing, University of Rochester, May 1977.

13. Thomas, G. B., Calculus and Analytic Geometry, Addison-Wesley, Reading MA, 1968.

APPENDIX A - The Newton-Raphson Method

A.1 Newton's Method

Consider a function f(x) which is continuous in the interval of interest. The objective is to determine the zero of the function, X_o, where $f(X_o)=0$.

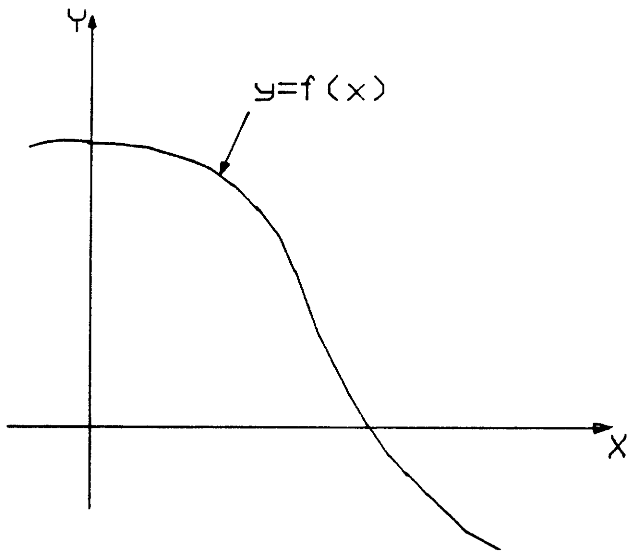


Figure A.1, Graph of f(x).

The steps of Newton's method are described below;

(1) Guess a value X_1 which is "near" the zero, X_o. (how "near" depends on how eradic the function is in the region of interest).

(2) Determine the slope of the function, $f'(X_1)$, at X_1.

(3) Determine X_2 from the equation below which corresponds to the graphical argument in Figure A.2.

$$X_{n+1} = X_n - f(X_n)/f'(X_n)$$

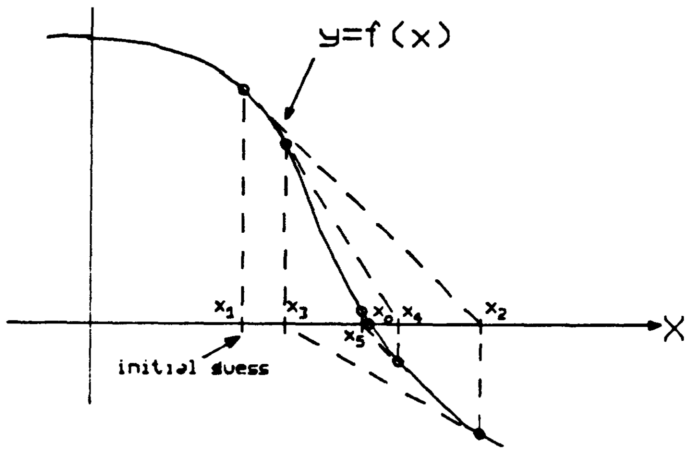


Figure A.2, Graphical description of Newton's method.

(4) X_2 is now the initial guess, go to step (1) and iterate until $f(X_n)=0$.

(5) If $f(X_n)=0$ then exit.

A.2 The Newton-Raphson Method

The Newton-Raphson method is an extention to the basic Newton's method presented above. It is used to determine the zeros of a set of simultaneous non-linear equations. First, we will investigate the case of two equations in two unknowns.

$$\text{Given; } F(x,y) = 0$$
$$G(x,y) = 0$$

Geometrically, this corresponds to finding the intersections of two curves; F=0, and G=0. Let X_o, Y_o be an initial guess for the common zero. The Taylor series expansion about the point (X_o,Y_o) of the functions, $F(x,y)$ and $G(x,y)$ are;

$$F(x,y)=0=F(X_o,Y_o)+(x-X_o)f_x(X_o,Y_o)+(y-Y_o)f_y(X_o,Y_o)+ \ldots$$
$$G(x,y)=0=G(X_o,Y_o)+(x-X_o)g_x(X_o,Y_o)+(y-Y_o)g_y(X_o,Y_o)+ \ldots$$

By neglecting the higher order terms, the above equations may be written;

$$(x-X_o)f_x + (y-Y_o)f_y = -F(X_o,Y_o)$$
$$(x-X_o)g_x + (y-Y_o)g_y = -G(X_o,Y_o)$$

Where f_x, f_y, g_x, and g_y are the partials taken at the guess point, (X_o,Y_o). Solving this set of linear equations for $x-X_o$ and $y-Y_o$ results in values for the increments in the initial guess. This provides a new "guess" which is closer to the real solution. Therefore, the Newton - Raphson method, generalized to two equations, takes the form;

$$(X_{n+1}-X_n)f_x + (Y_{n+1}-Y_n)f_y = -F(X_n,Y_n)$$

$$(X_{n+1}-X_n)g_x + (Y_{n+1}-Y_n)g_y = -G(X_n,Y_n)$$

where f_x, f_y, g_x, and g_y are the partials of F and G taken at (X_n,Y_n).

In matrix form, the above equations take the form;

$$\begin{bmatrix} f_x & f_y \\ g_x & g_y \end{bmatrix}_{x_n, y_n} \bullet \begin{bmatrix} X_{n+1} - X_n \\ Y_{n+1} - Y_n \end{bmatrix} = \begin{bmatrix} -F \\ -G \end{bmatrix}_{x_n, y_n}$$

In the general case, there are m-equations (f_1, f_2, f_3, f_4 ... f_m) in m-unknowns (X_1, X_2, X_3 ... X_m). To simplify the notation, let;

$$dx_1 = X_{1_{n+1}} - X_{1_n}$$

$$f_{12} = \partial f_1 / \partial X_2$$

The generalized Newton-Raphson method takes the form;

$$\begin{bmatrix} f_{11} & f_{12} & f_{13} & \cdot\cdot & f_{1m} \\ f_{21} & f_{22} & & & f_{2m} \\ f_{31} & & \ddots & & \vdots \\ \cdot & & & \ddots & \vdots \\ f_{m1} & f_{m2} & & \cdots & f_{mm} \end{bmatrix} \bullet \begin{bmatrix} dx_1 \\ dx_2 \\ \vdots \\ dx_m \end{bmatrix} = \begin{bmatrix} -f_1 \\ -f_2 \\ \vdots \\ -f_m \end{bmatrix}$$

Or;

$$\underline{J}\ \underline{dx} = \underline{R}$$

where; $\underline{J}$ is the Jacobian matrix
$\underline{dx}$ is the vector of displacements
$\underline{R}$ is the vector of residuals.

APPENDIX B - Modified Doolittle's Method

Doolittle's method for solving a system of linear equations $\underline{\bar{A}}\underline{x}=\underline{b}$ is a variation of the Crout reduction [11]. Basically, if the matrix, $\underline{\bar{A}}$, is non-singular, it can be written in the form $\underline{\bar{A}}=\underline{\bar{L}\bar{U}}$ where the matrix $\underline{\bar{L}}$ is lower triangular with unity on the diagonal and matrix $\underline{\bar{U}}$ is upper triangular. The solution vector, $\underline{x}$, may be found by solving the equation $\underline{\bar{L}y}=\underline{b}$ for the vector $\underline{y}$, and then the equation $\underline{\bar{U}x}=\underline{y}$ for the vector $\underline{x}$. The initial discussion will assume that the matrix, $\underline{\bar{A}}$, is non-singular.

The equation $\underline{\bar{A}}=\underline{\bar{L}\bar{U}}$ may be written term by term;

$$a_{ik} = \sum_{j}^{\min(i,k)} L_{ij}U_{jk} = (LU)_{ik} \qquad \text{(B.1)}$$

The upper limit on the summation, min(i,k) is a result of the triangularity of the matricies $\underline{\bar{L}}$ and $\underline{\bar{U}}$. Examining the summation in its two regions, $i \leq k$ and $i > k$;

$$i > k \quad a_{ik} = \sum_{j=1}^{k} L_{ij}U_{jk} = \sum_{j=1}^{k-1} L_{ij}U_{jk} + L_{ik}U_{kk} \qquad \text{(B.2)}$$

$$i \leqslant k \quad a_{ik} = \sum_{j=1}^{i} L_{ij}U_{jk} = \sum_{j=1}^{i-1} L_{ij}U_{jk} + L_{ii}U_{ik} \qquad \text{(B.3)}$$

From equation B.2 and using the fact that $\bar{\underline{L}}$ has unity on the diagonal (ie.-L_{ii}=1), the element of the $\bar{\underline{U}}$ matrix may be written;

$$U_{ik} = a_{ik} - \sum_{j=1}^{i-1} L_{ij} U_{jk} \qquad i<k \tag{B.4}$$

From equation B.3, the element of the $\bar{\underline{L}}$ matrix may be written;

$$L_{ik} = \left(a_{ik} - \sum_{j=1}^{k-1} L_{ij} U_{jk}\right) \Big/ U_{kk} \qquad i>k \tag{B.5}$$

In solving for L_{ik}, it is desired to divide by the largest available number, U_{kk}. If the matrix $\bar{\underline{A}}$ is singular, at some point, the largest U_{kk} available will be zero. When this occurs, the column must be pivoted with a new unprocessed column and reprocessed. The process of row pivoting so that the largest U_{kk} is on the diagonal in conjunction with the column pivoting will result in all the "bad" columns (associated with unconstrained variables) and all the dependent rows being placed in the last columns and rows of the $\bar{\underline{L}}$ and $\bar{\underline{U}}$ matricies.

When the matrix, $\bar{A}$, is singular, it is no longer true that $\bar{A}=\bar{L}\bar{U}$. The equations $\bar{L}y=b$ and $\bar{U}x=y$ will result in a parametric solution to the equation $\bar{A}x=b$ if the dependent rows are ignored. The free parameters are simply the unconstrained variables. If these parameters are set to zero, a partial solution may be found to the equation $\bar{A}x=b$.

At this point, a detailed description of the algorithm will be presented. This is based directly on equations B.2 and B.3. For simplicity, the matricies $\bar{L}$ and $\bar{U}$ will exist in a single matrix $\overline{LU}$. The unity diagonal of $\bar{L}$ will be assumed and not stored in $\overline{LU}$. The $\bar{U}$ matrix will include the diagonal and all upper elements of $\overline{LU}$. The $\bar{L}$ matrix will include all elements below the diagonal. Assume that the orignal matrix, $\bar{A}$, is NxN. The counter NGOOD is the number of linearly independent equations and hence the number of fully constrained unknowns.

The following procedure will create the $\overline{LU}$ matrix.

1) Initialize NGOOD=N and K=1. K indicates the column which is being processed.

2) If K>NGOOD then exit.

3) Process every element in column K according to equation B.4.

$$LU_{ik} = a_{ik} - \sum_{j=1}^{\min(ik)} L_{ij} U_{jk} \qquad \text{(B.4)}$$

4) Find the position of the element in column K with the max. absolute value - this will be the IMAX row.

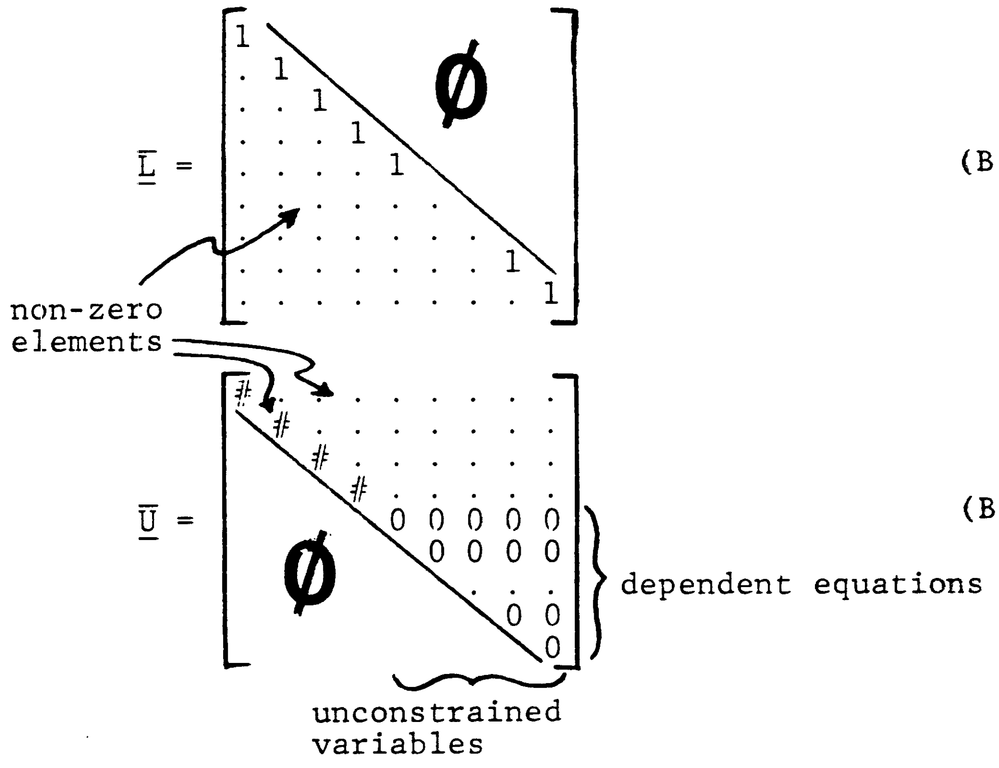
5) If maximum is non-zero then go to step 6 otherwise, pivot column K with column NGOOD, decrement NGOOD then go to step 2 and reprocess the column.

6) Pivot row K with row IMAX. The maximum element is now on the diagonal.

7) Divide by LU_{KK} down column K from row K+1 to row NGOOD.

8) Increment K and go to step 2

At this point, the $\underline{\bar{L}}$ and the $\underline{\bar{U}}$ matricies are complete. If the matrix, $\underline{\bar{A}}$, is non-singular, the equations $\underline{\bar{L}}\underline{y}=\underline{b}$ and then $\underline{\bar{U}}\underline{x}=\underline{y}$ may be solved by simple row reduction to obtain the solution vector $\underline{x}$. If the matrix, $\underline{\bar{A}}$, is singular, the $\underline{\bar{L}}$ and the $\underline{\bar{U}}$ matricies have the form;



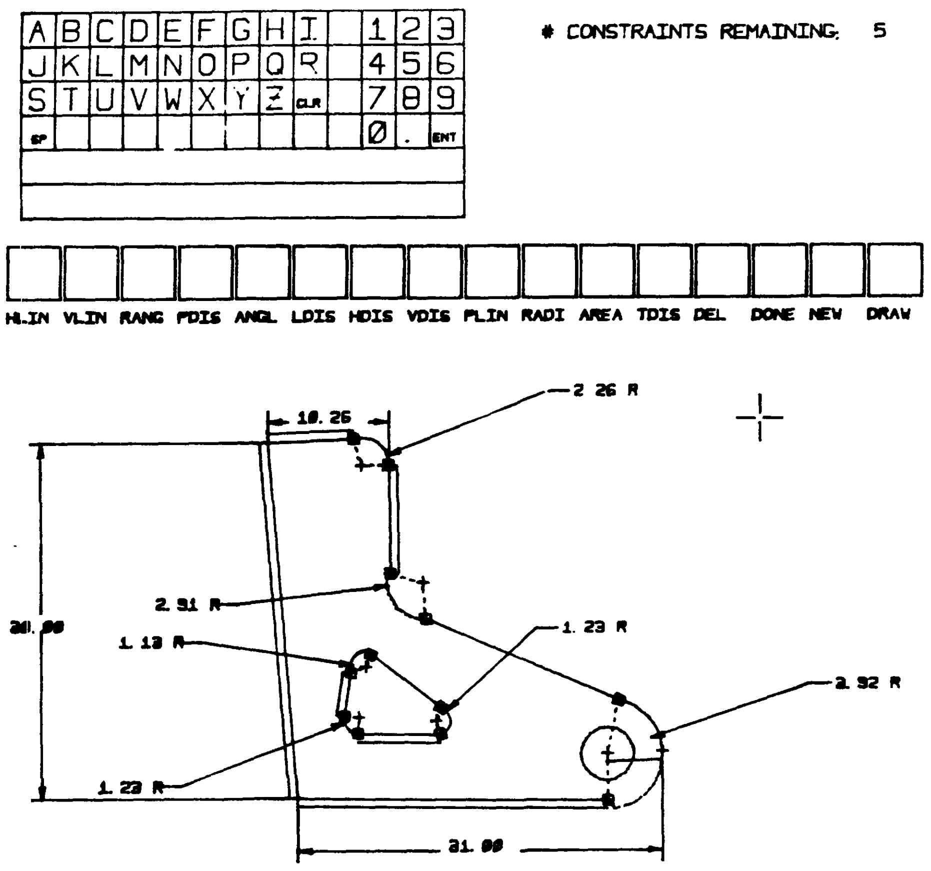
(B.5)

(B.6)

The equation $\underline{\bar{L}}\underline{y}=\underline{b}$ may be solved for the vector, $\underline{y}$, by row reduction. In order to solve the equation $\underline{\bar{U}}\underline{x}=\underline{y}$ we must set $x_i=0$ for i=NGOOD+1, ... N and then solve for the elements x_i, i=1,...NGOOD using row reduction. This will produce a solution, $\underline{x}$, for the equation $\underline{\bar{A}}\underline{x}=\underline{b}$ when $\underline{\bar{A}}$ is singular by ignoring the dependent equations. The dependent rows are rows NGOOD+1 ...N, and the unconstrained variables are x_i, i=NGOOD+1,...,N. If the original position of the rows and columns are noted as the pivoting is carried out, the indicies of the dependent equations and unconstrained unknowns would be determined. In practice, only pointers to the rows and columns are actually pivoted thus maintaining the identity of the dependent equations and unconstrained unknowns.

APPENDIX C - Example Session

In order to illustrate some of the features of the DIMENSION system, a sample session using the system will be presented. The initial topology, as shown in Figure C.1, is entered using the interactive sketch input procedures.



A B C D E F G H I 1 2 3
J K L M N O P Q R 4 5 6
S T U V W X Y Z CLR 7 8 9
SP 0 . ENT
# CONSTRAINTS REMAINING: 5
HLIN VLIN RANG PDIS ANGL LDIS HDIS VDIS PLIN RADI AREA TDIS DEL DONE NEW DRAW
2.26 R
10.26
2.31 R
1.23 R
30.00
1.13 R
2.92 R
1.23 R
31.00

Figure C.1, Initial sketch.

The next step is to apply dimensional constraints to the geometry. The double lines in Figure C.1 represent implicit constraints and the boxes represent points of tangency. At this point, only a few constraints will be specified.

The part is under-dimensioned at this point. Note that there are five constraints remaining. Even so, the part may be straightened, as in Figure C.2, by the Newton-Raphson method.

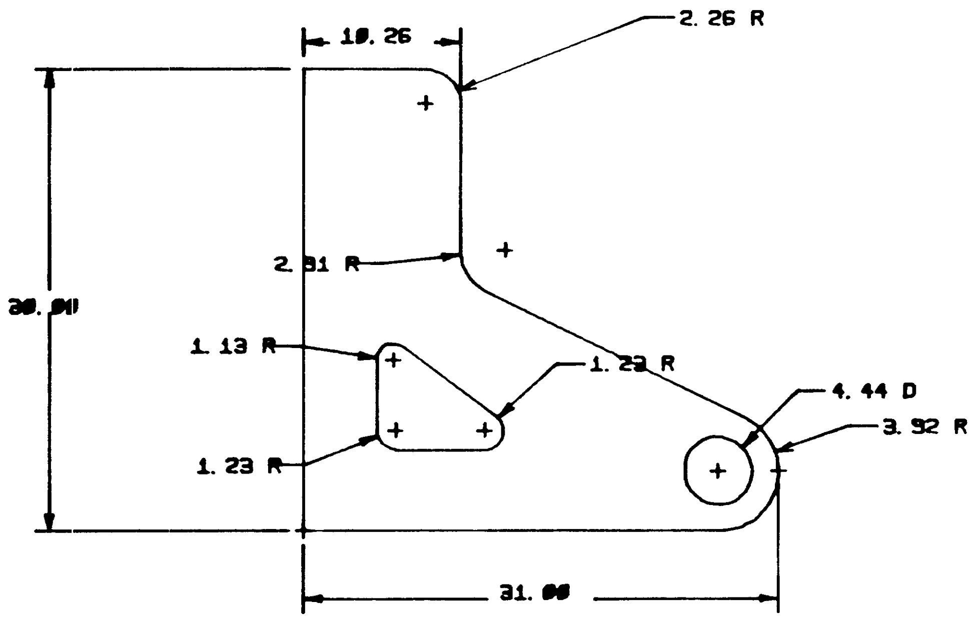


Figure C.2, Straightened part.

Note that only those constraints which were defined are used to modify the part geometry. The user is notified as to which coordinates remain unconstrained and any redundant dimensions (if there were any). This is useful in determining the dimensional constraints which remain to be specified.

The remaining constraints may then be defined. A mistake will be made and the part will be redundantly dimensioned. Note that in Figure C.3, the part has the correct number of constraints but not the correct placement.

When the Newton-Raphson method is executed (by pressing "GO"), the display as shown in Figure C.4 results. The blinking dimensions are redundant and the unconstrained points are circled and the coordinates (X and/or Y) are indicated.

The user returns to the dimension specification routine and deletes the redundant dimensions and specifies other dimensions so as to constrain the coordinates which were previously unconstrained. The resulting dimensioning scheme is shown in Figure C.5.

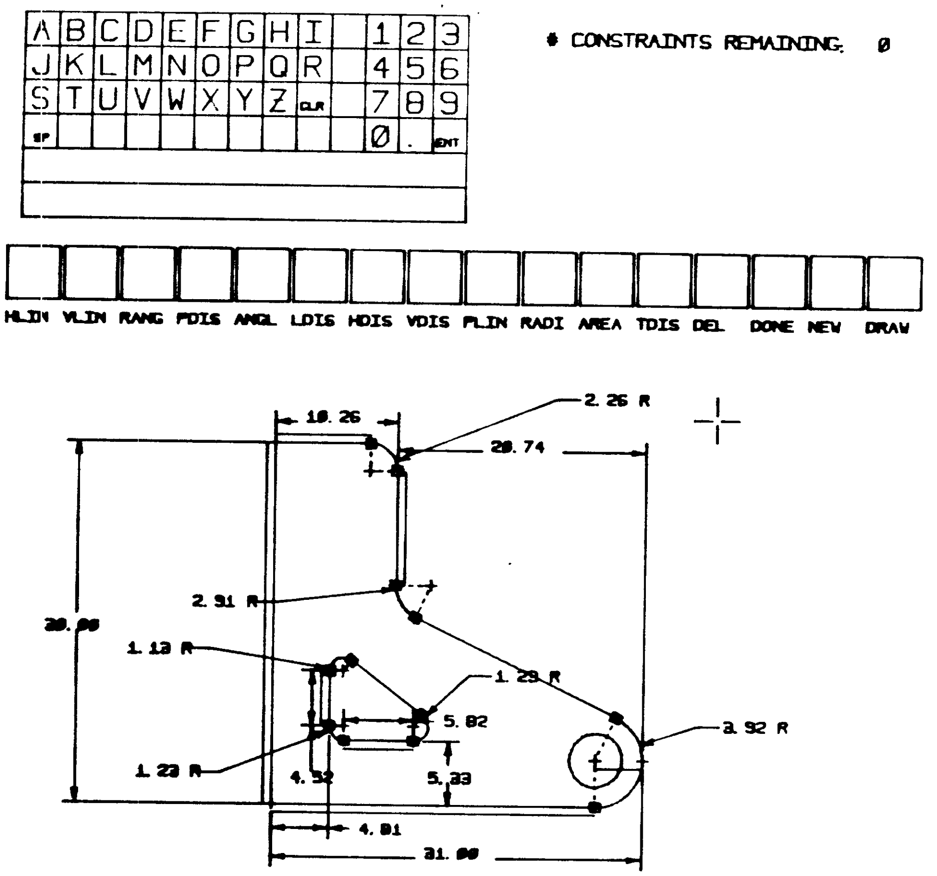
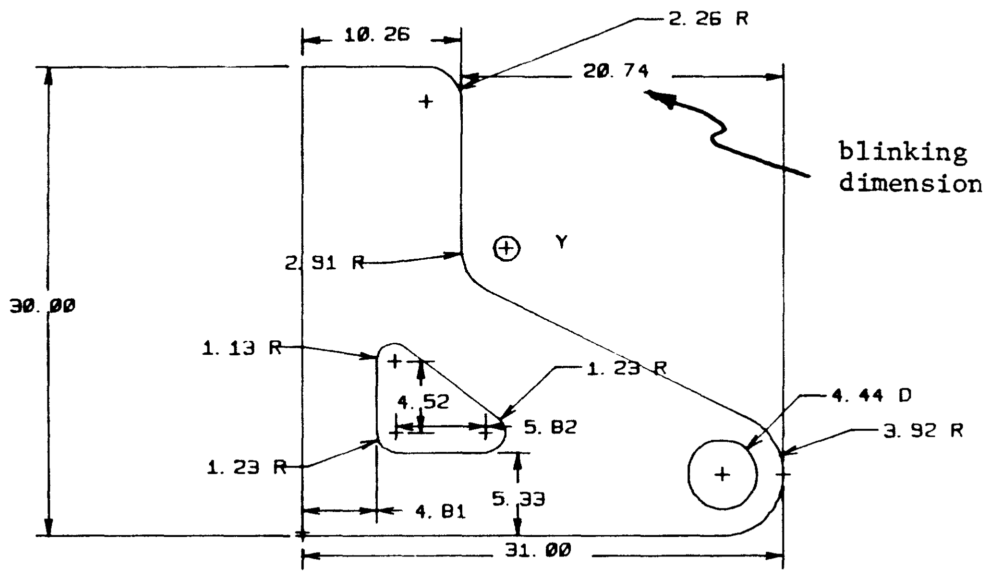

A B C D E F G H I 1 2 3
J K L M N O P Q R 4 5 6
S T U V W X Y Z CLR 7 8 9
SP Ø . ENT
# CONSTRAINTS REMAINING: Ø
HLIN VLIN RANG PDIS ANGL LDIS HDIS VDIS PLIN RADI AREA TDIS DEL DONE NEW DRAW
2.26 R
10.26
20.74
2.31 R
30.00
1.13 R
1.29 R
5.82
3.92 R
1.23 R
4.52
5.33
4.81
31.00

Figure C.3, Full dimensioning scheme.



BLINKING DIMENSIONS ARE REDUNDANT ** CIRCLED POINTS ARE UNDER-CONSTRAINED

Figure C.4, Notification of redundant dimensions.

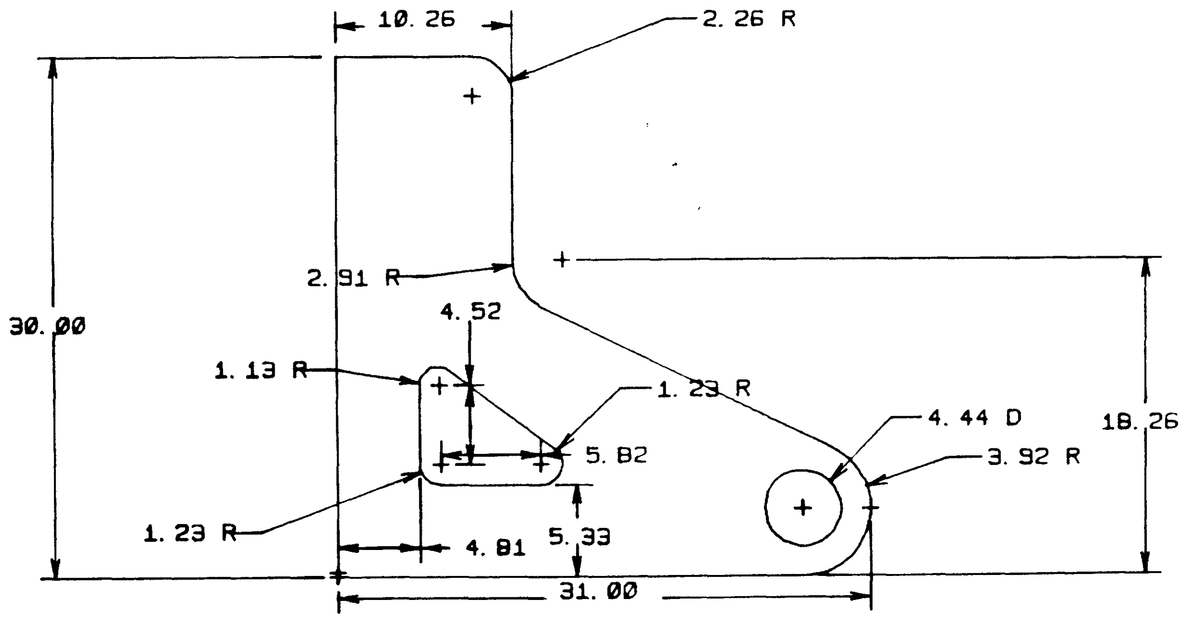


Figure C.5, Corrected dimensioning scheme.

The values of the dimensions are then modified. This results in the desired geometry as shown in Figure C.6.

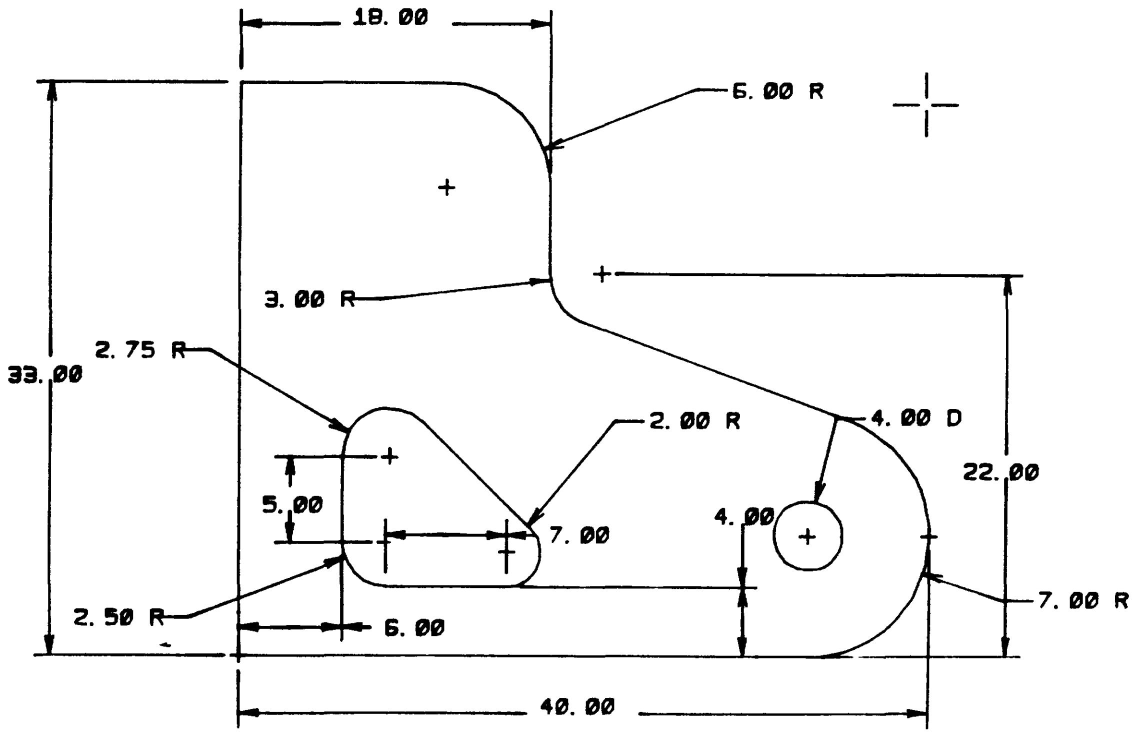
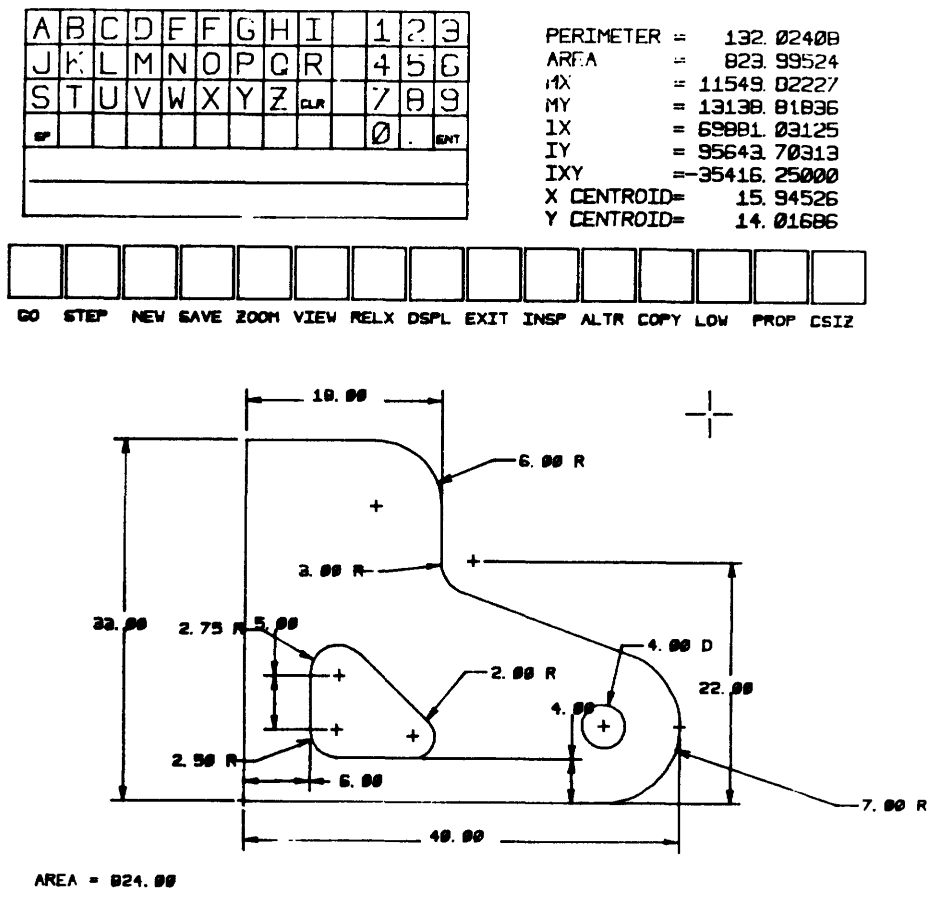


Figure C.6, Desired geometry.

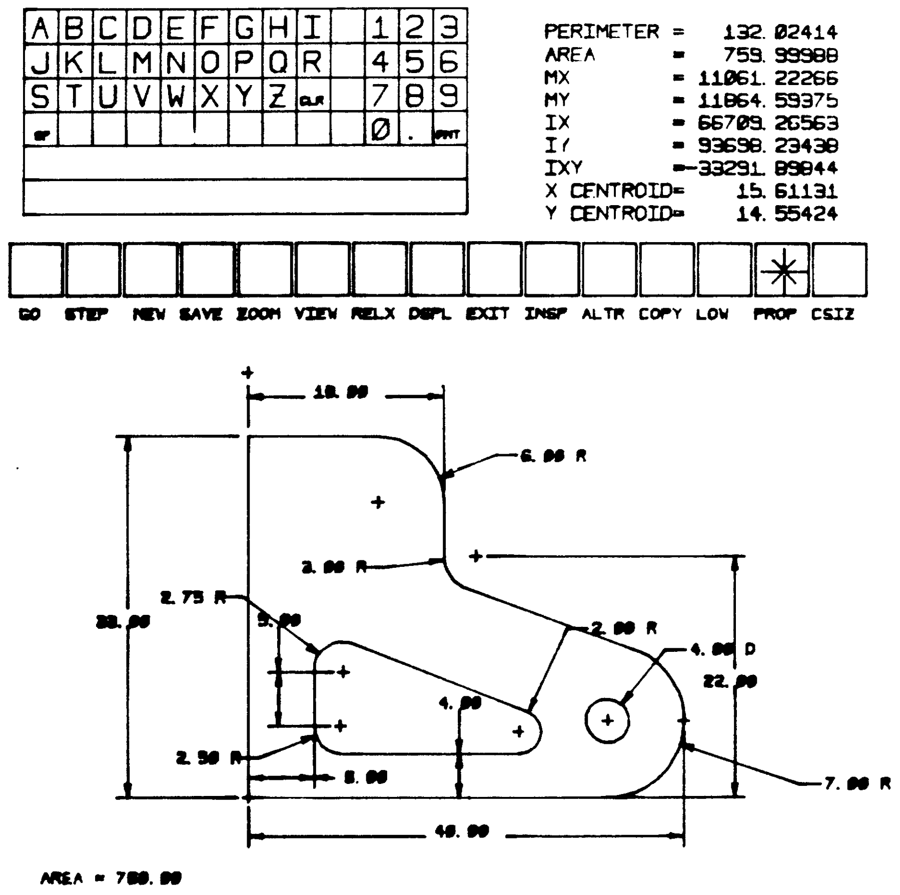
If, for example, the user was not concerned about the horizontal width of the triangular hole but instead wished to constrain the area of the part, the dimensioning scheme of Figure C.7 results. The "PROP" button was pushed to display the current geometrical properties. The text "AREA = 824.00" at the lower left indicates that the area has been constrained.



```
PERIMETER =    132.02408
AREA      =    823.99524
MX        =  11549.02227
MY        =  13138.81836
IX        =  69881.03125
IY        =  95643.70313
IXY       =-35416.25000
X CENTROID=     15.94526
Y CENTROID=     14.01686
```

Figure C.7, Constraining the area.

The area of the part may now be specified. The text "AREA = 824.00" is digitized and the desired area, 760.00, is entered via the graphic keypad. The geometry may then be modified to reflect this change in the area. This is shown in Figure C.8.



```
PERIMETER =    132.02414
AREA      =    759.99988
MX        =  11061.22266
MY        =  11864.59375
IX        =  66709.26563
I/        =  93698.23438
IXY       = -33291.89844
X CENTROID=     15.61131
Y CENTROID=     14.55424
```

Figure C.8, Resultant geometry.

This conclues the example session. All of the features developed in this paper have been demonstrated to some extent by the previous session.

An in-depth desciption of the pen strokes needed to perform the various functions may be found in Appendix D.

APPENDIX D - Operator's Manual

This appendix is intended as an operator's manual for the DIMENSION system. The information contained within is, in some places, repetitive of that which is in the thesis. It is included here as a self-contained document for the purpose of clarifying the use of the system.

DIMENSION SYSTEM

OPERATOR'S MANUAL

This document describes the DIMENSION system from an operator's viewpoint. It is divided into three main sections:

1) General information
2) Graphic input
3) Specification of constraints
4) Modification of geometry

The first section describes the use of the data tablet and the initial menu choices in order to shift control to the different procedures. The second section describes the protocol used for sketching the initial topology. The third section describes the different methods of specifying dimensional constraints. The last section describes the method for changing the numeric values of the dimensions, control of the numerical iteration, and file handling.

SECTION I. General information

The DIMENSION system uses a data tablet and digitizing pen for all data entry. The keyboard is not used but the VT100 terminal on the PDP 11/34 must be turned on as the program on the PDP11 uses the terminal to produce an audible bell.

The digitizing pen and data tablet interact in three modes as shown in Figure D.1. The first mode is when the pen is not in proximity to the tablet surface. In this mode, no useful data is being sent to the program.

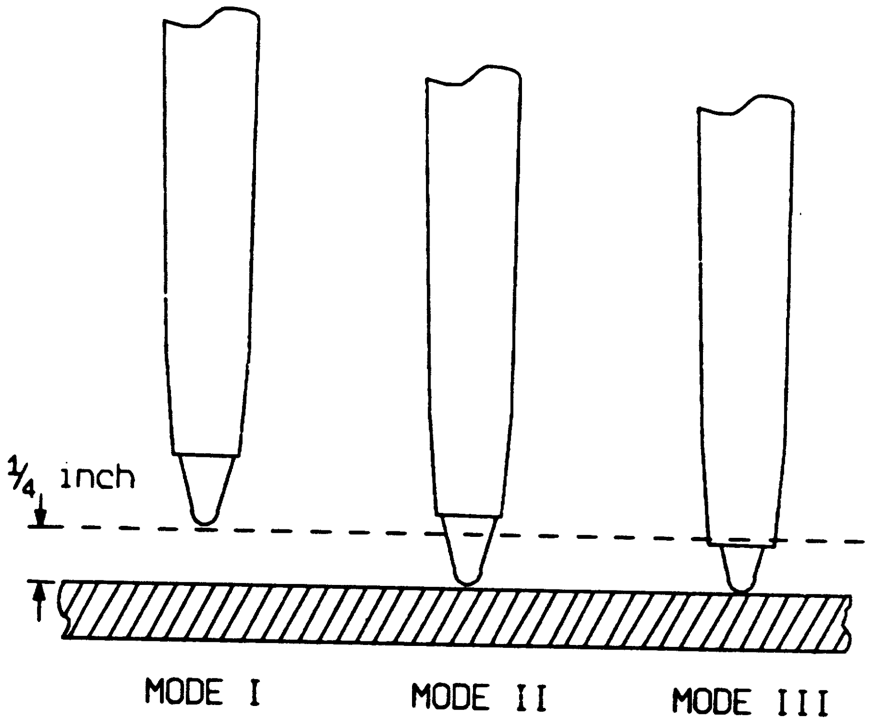


Figure D.1, Different modes of the digitizing pen.

In the second mode, the pen is in proximity to the tablet surface. In mode three, the small button at the tip of the pen is depressed. The data tablet is sending useful data to the program in both modes II and III.

The initial menu is displayed at the start of the program as shown in Figure D.2. The tracking cross which is also shown, is used to track the position of the pen on the tablet surface.

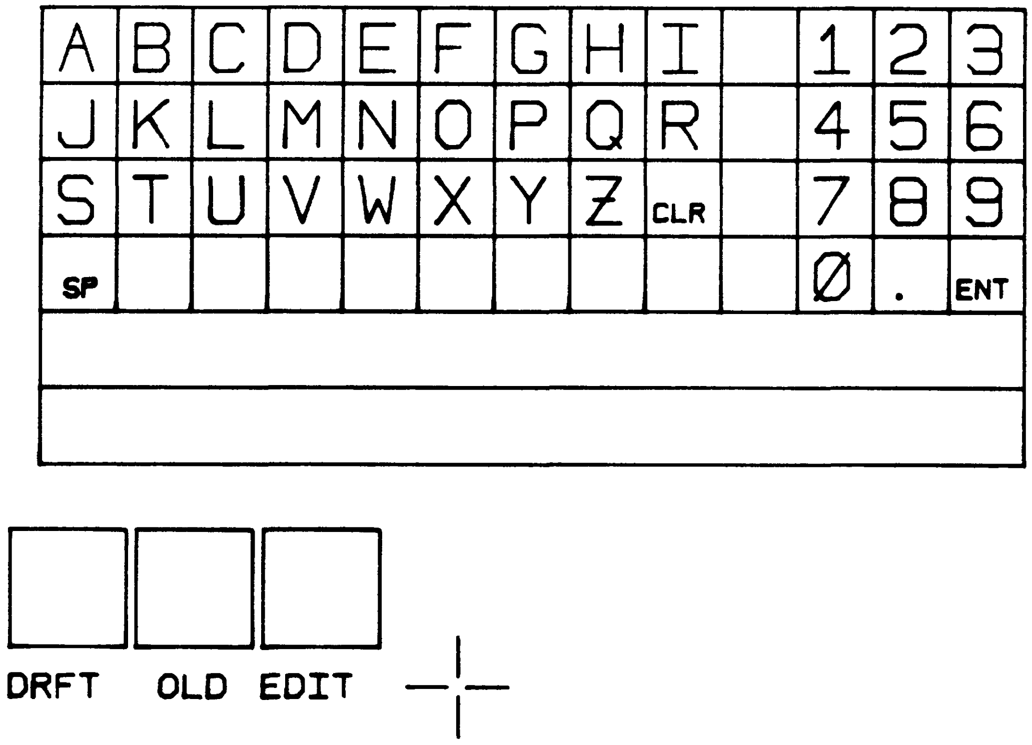


Figure D.2, Menu selection and tracking cross.

When the pen is depressed (mode III) an additional cross appears as in Figure D.3.

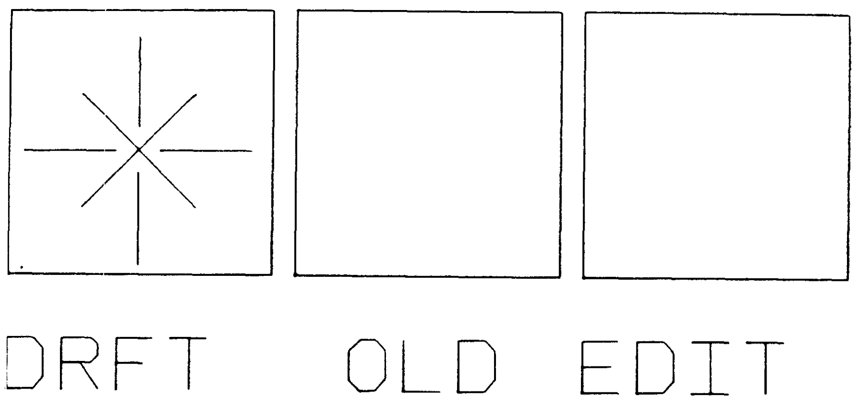
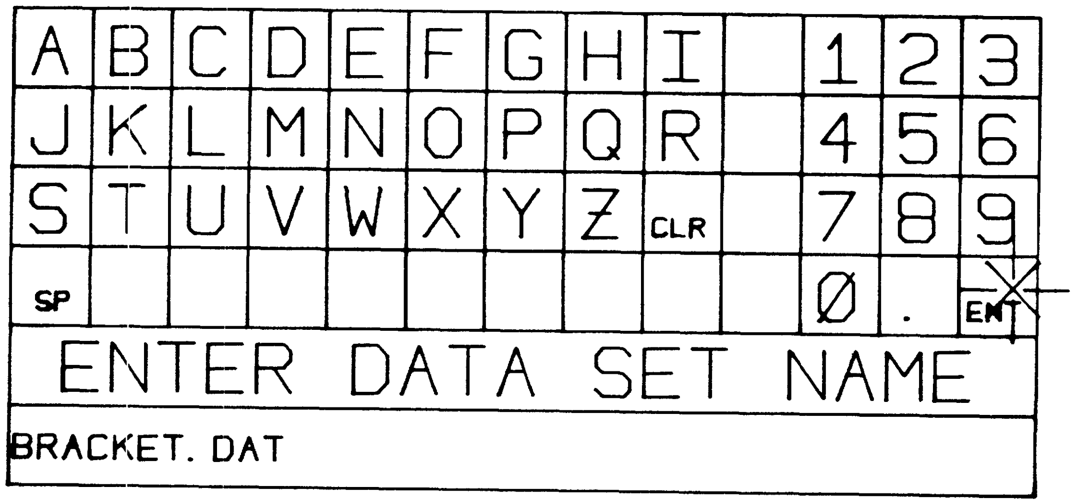


Figure D.3, Picking menu item.

When DRFT is picked (see Figure D.3), the user is permitted to "draft" a new shape. Program execution is then passed to the sketch input routines.

The OLD button is used to read in a previously stored (old) data set. The operator is prompted to enter the data set name by using the graphic keypad which is displayed on the screen (see Figure D.4).



ENTER DATA SET NAME

BRACKET. DAT

Figure D.4, Graphic keypad entry.

The prompt is written in the graphic keypad area and flashes until the response is entered. The pen is used to press the appropriate letters (or numbers). If a mistake is made, the entry may be cleared by pressing the CLR key. The enter key, ENT, is the equivalent of a carriage return and is used to terminate the data entry.

The EDIT key is used when the dimensioning scheme is to be modified (edited). Control is then passed to the dimension specification program which permits the modification of the dimensioning scheme.

SECTION II. Graphic Input

When the DRFT button, in Figure D.3, is pressed, control is passed to the graphic input section. The menu, as in Figure D.5, permits the user to either clear (CLR) the display and enter a new shape, exit from the input section (DONE) and enter the dimension specification section, or make a hardcopy (COPY) of the display.

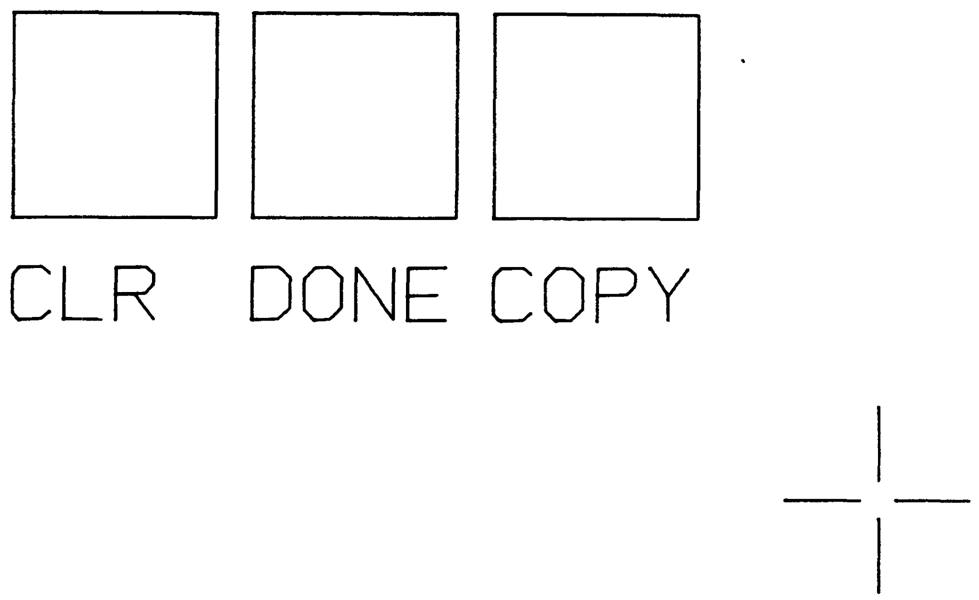


Figure D.5, Menu display in graphic input section.

The input of any shape contour is separated into "loops" each loop is a connected sequence of lines and arcs which finally connect back to the initial point. The first entity of each loop must be a line.

To insert the first loop, the tracking cross is placed at the position of the first point. The pen is depressed (mode III) and released. A plus "+" sign is drawn and a "rubberband" is attached to the tracking cross and the first point as in Figure D.6.

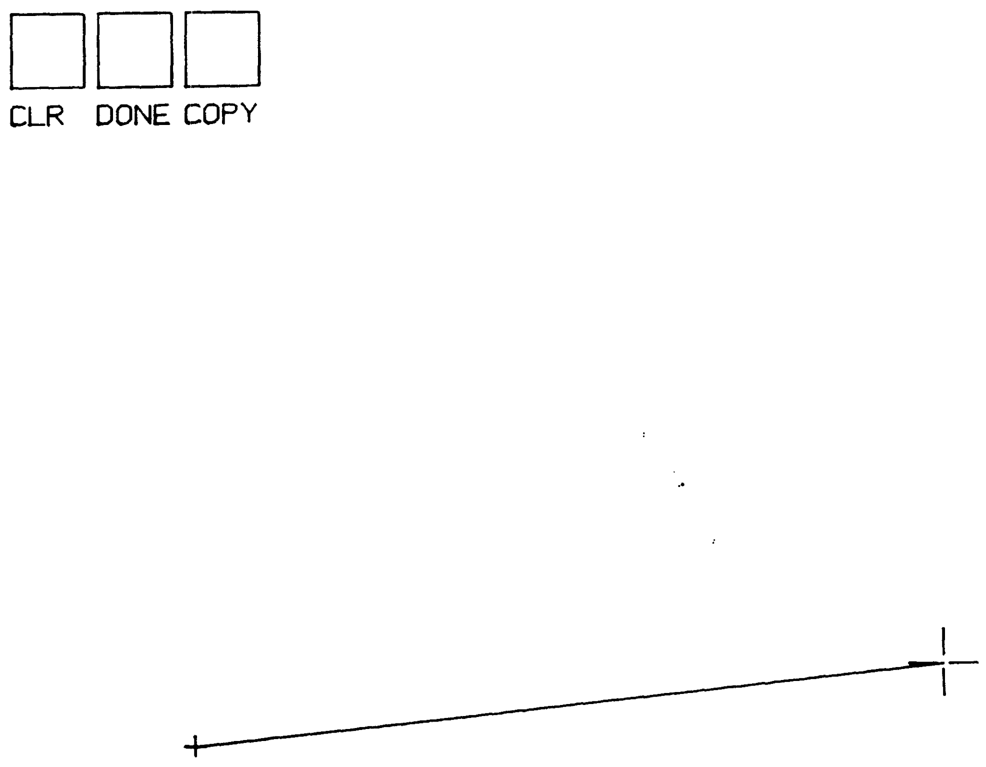


Figure D.6, "Rubberbanding" of the first line.

To enter the second point, the tracking cross is positioned, depressed and released. Now, the rubberband is between the tracking cross and the second point (see Figure D.7).

This process continues until the tracking cross is positioned near the initial point and depressed. Then, the rubberband is connected to the first point and the loop is complete (see Figure D.8).

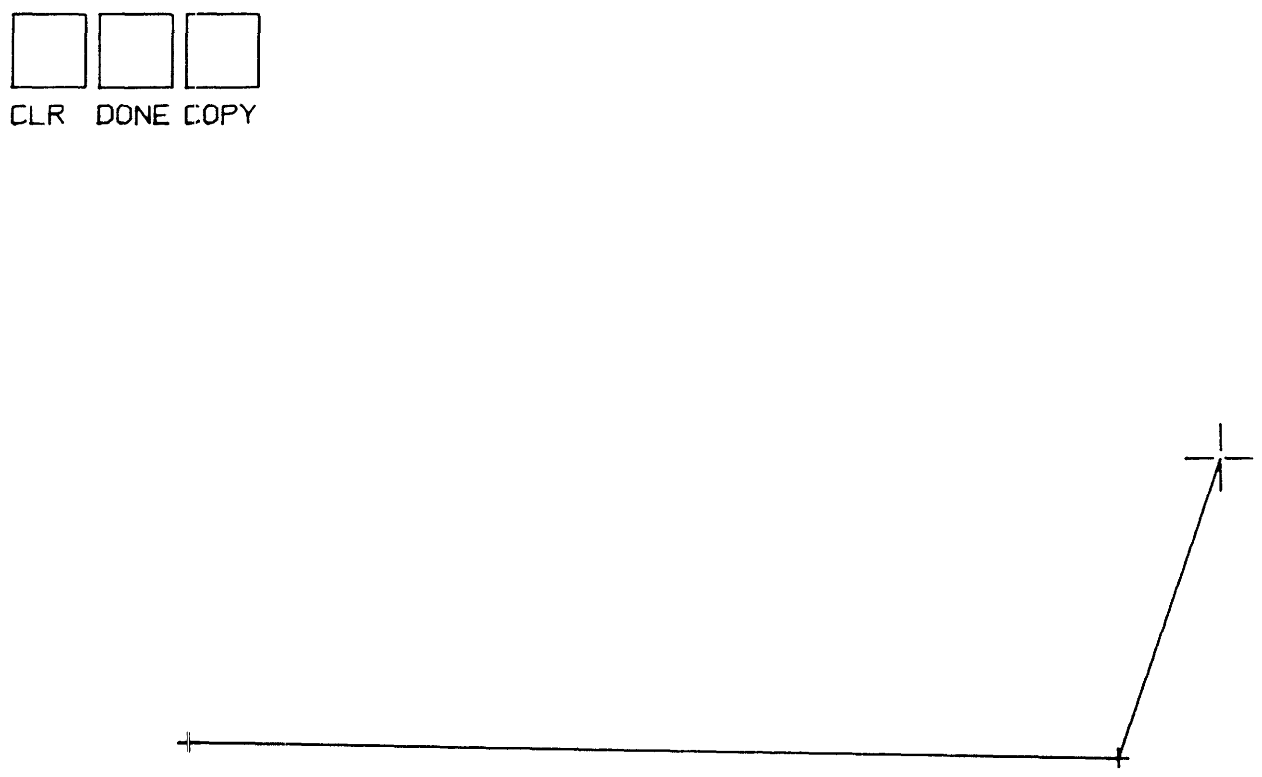


Figure D.7, Insertion of the second point.

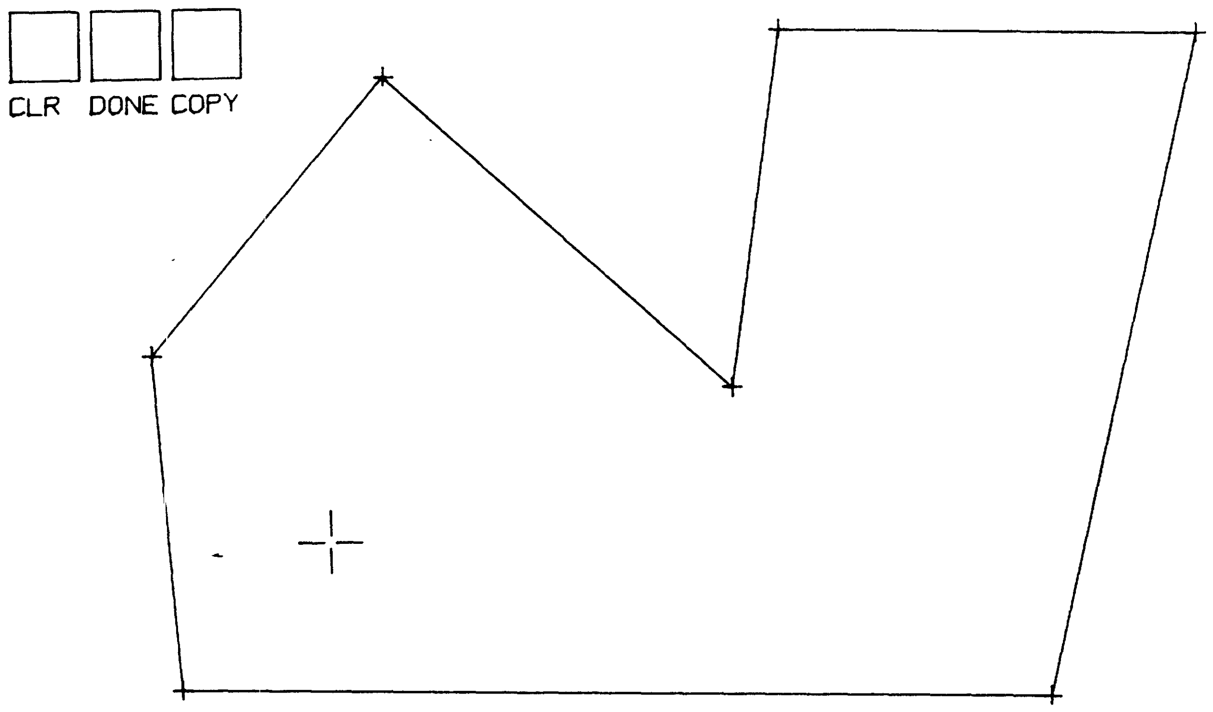


Figure D.8, Completion of the first loop.

At this point, the tracking cross is free of rubber-bands, and the operator may do one of four operations:

1) push a menu button (in Figure D.5)
2) start another loop
3) insert a single point
4) insert a circle

It must be emphasized that these options can only be used when the tracking cross is free of "rubberbands". In other words, any loop must be completed before the options above are valid.

To insert a single point, the tracking cross must be placed at the position of the point and then depressed and released two times. Again, the tracking cross is free of rubberbands.

A circle is entered by positioning the tracking cross on the diameter of the desired circle, then depressing the pen while tracing the contour of the circle.

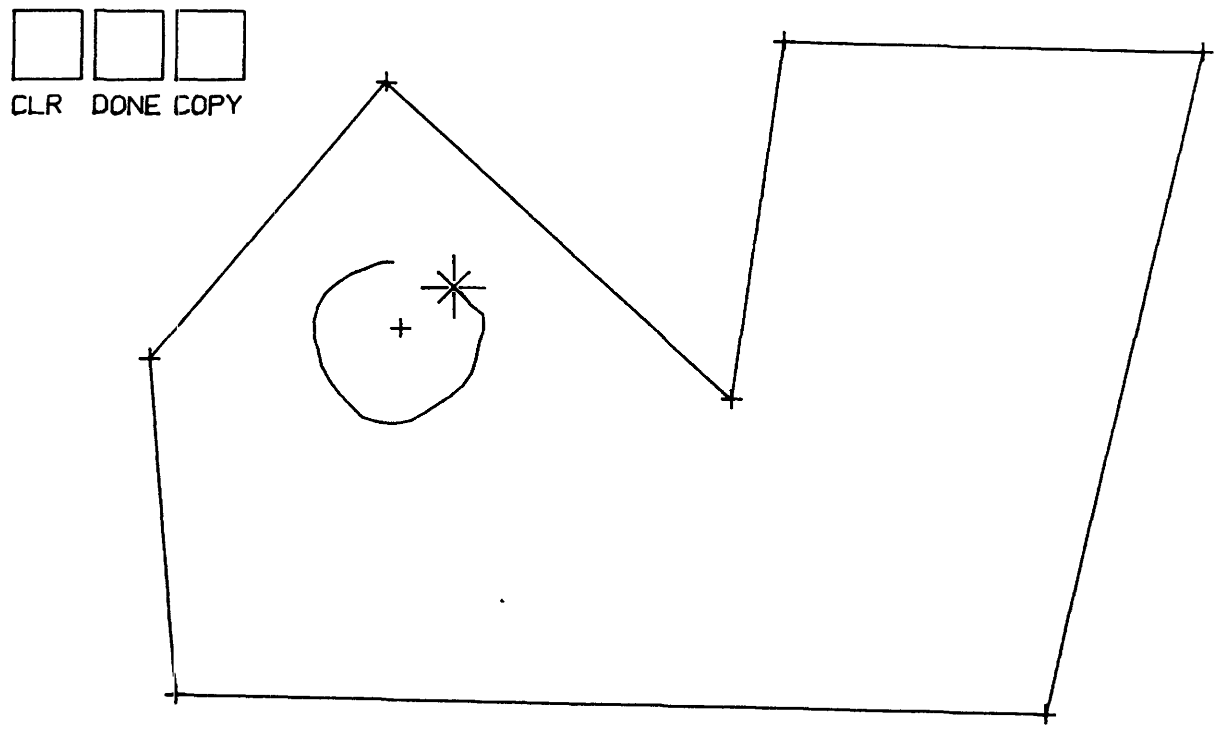


Figure D.9, Insertion of a circle. Notice that the pen is depressed as the circle is traced.

A circle must be entered around a single point and must be traced a full 360 degrees. If either of these criteria is violated, the circle automatically disappears and the user is then free to insert a loop, point or another circle.

Entering Arcs

Arcs may be specified by depressing the pen and simultaneously tracing the contour of the arc as in Figure D.10.

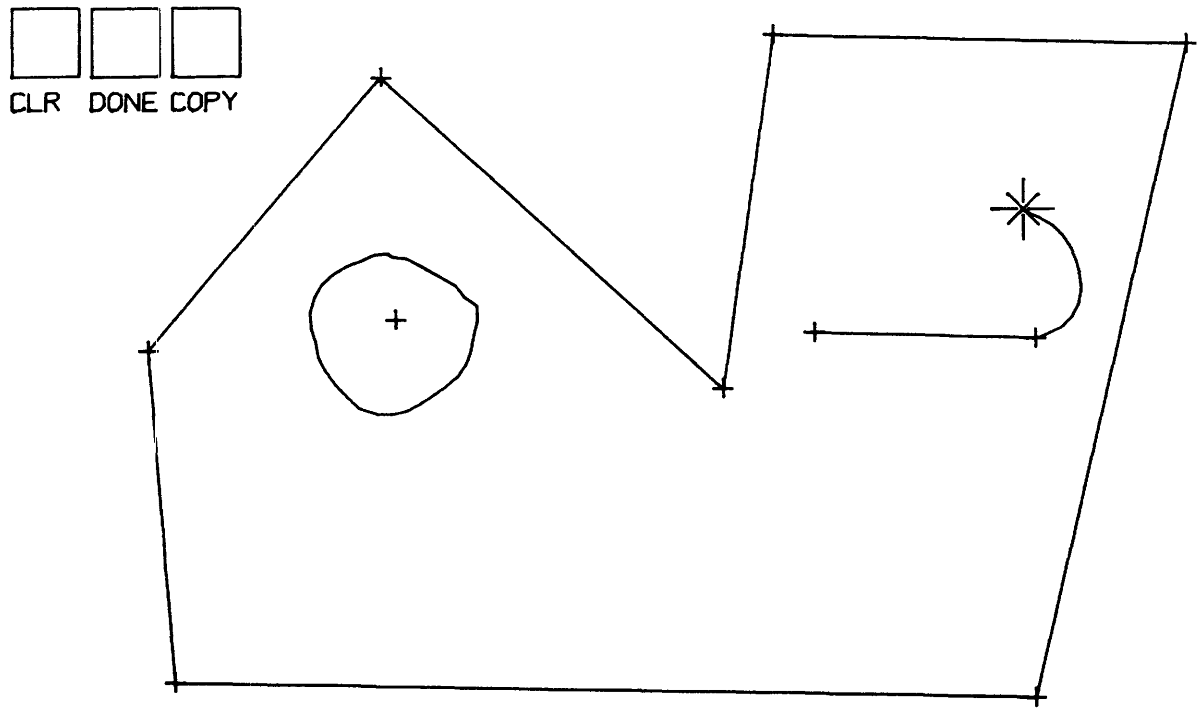


Figure D.10, Sketching an arc.

At the end of the arc, the pen is lifted and a plus sign is drawn to indicate the endpoint of the arc. A rubberband is then attached from this point to the tracking cross, as in Figure D.11.

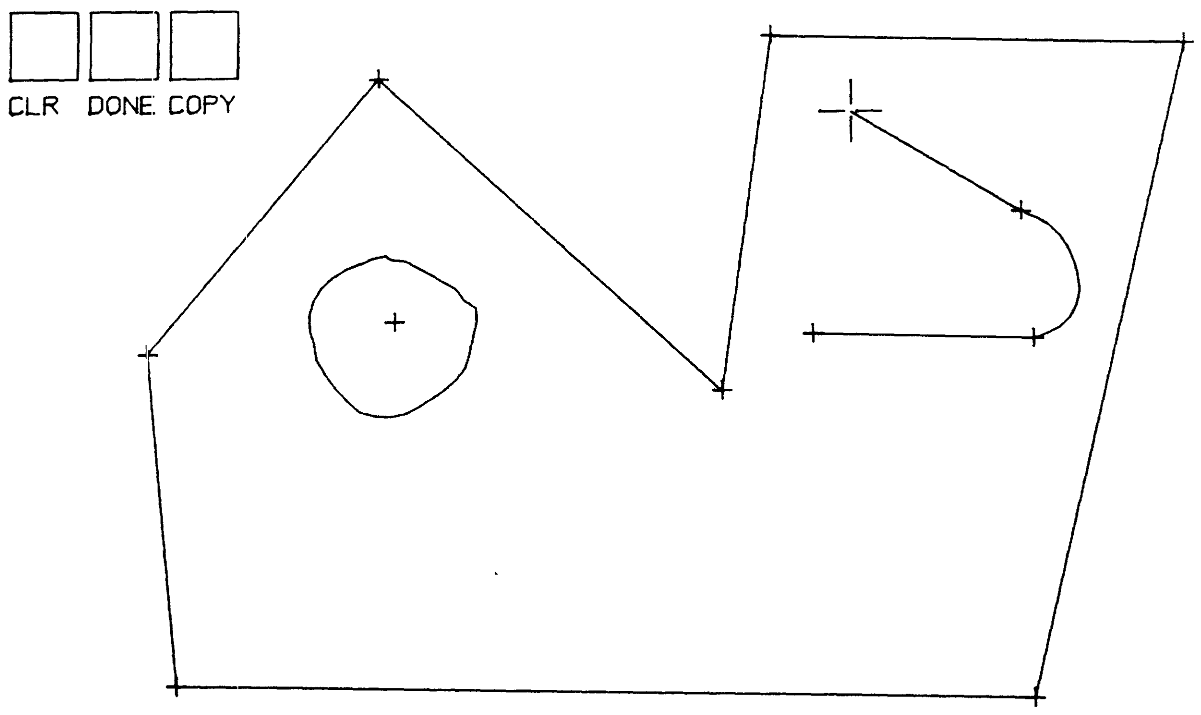


Figure D.11, Completion of an arc.

The operator must make sure that there is a point near the center of the sketched arc. This point may be a point on a loop or a single point. Two arcs may share the same center. If a point is not found near the center of the sketched arc when the DONE command is pressed, the operator is notified by the display in Figure D.12. The crossing of the arc indicates that the center of that arc was not found. The operator may then insert the center point and continue.

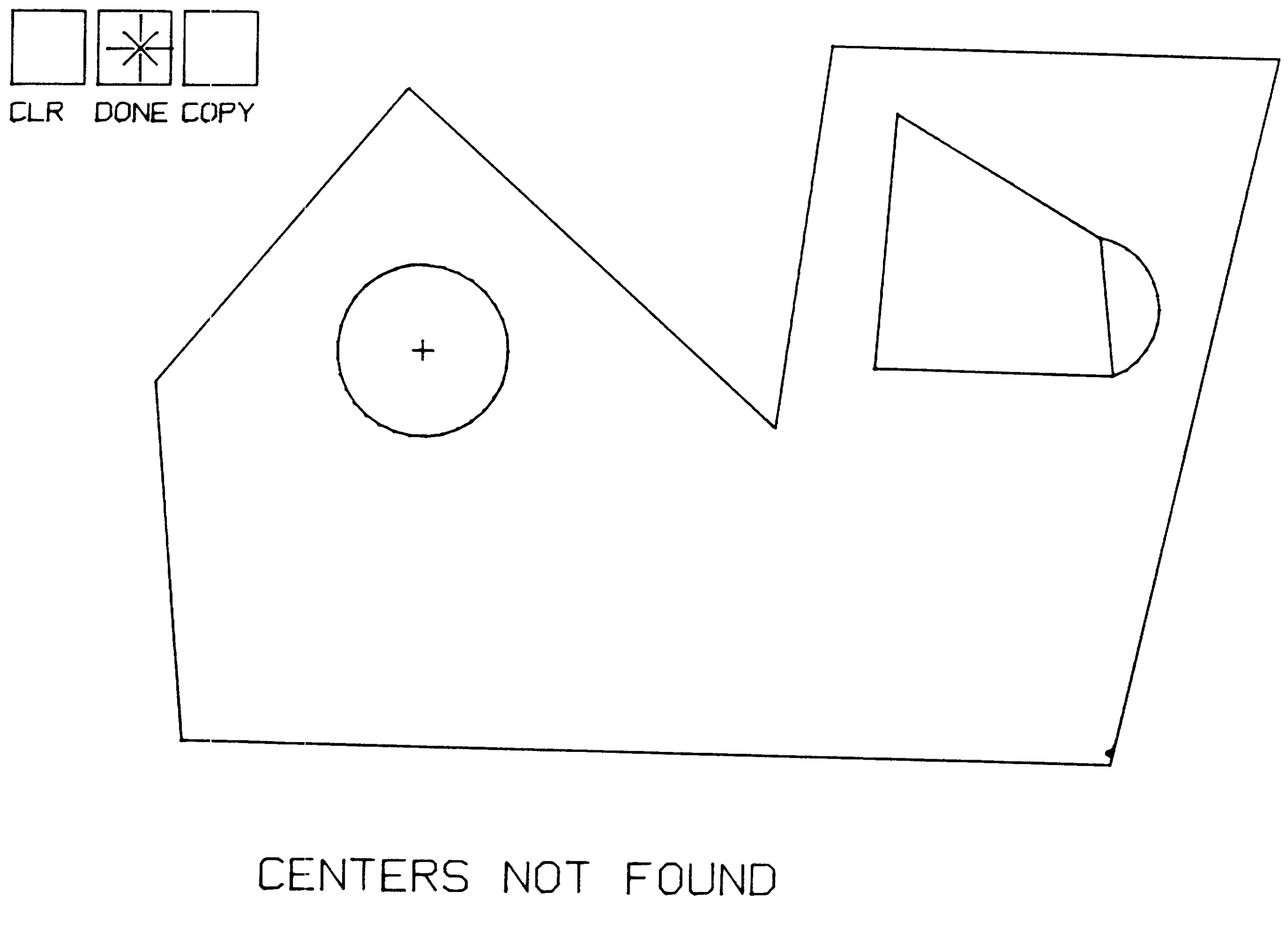
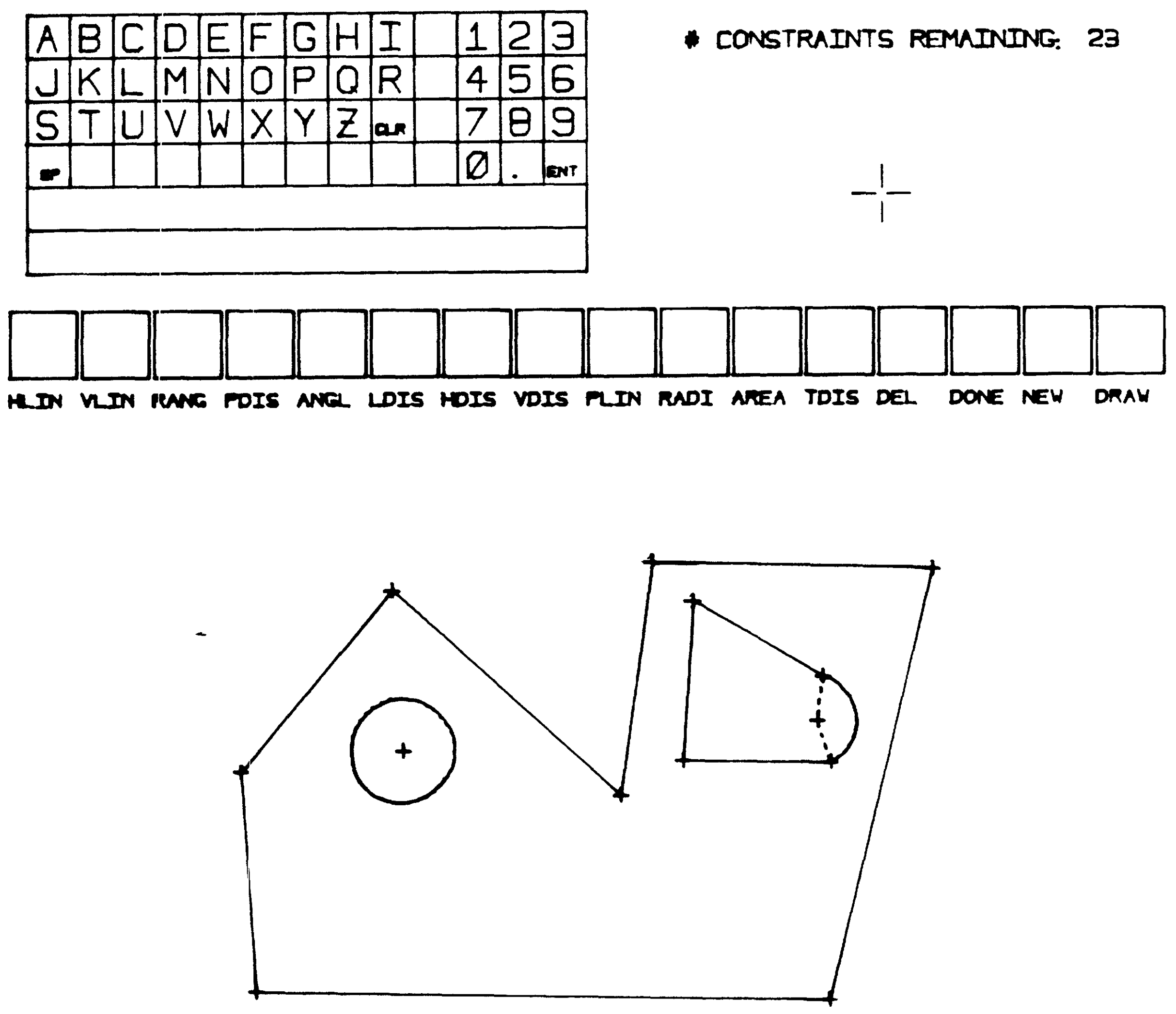


Figure D.12, Detecting absence of center point.

SECTION III. Specification of Constraints

This section describes the use of the interactive "constraint definition" feature of the DIMENSION system. This is where the operator may either define or delete a dimensional constraint. Program control may be shifted into this section via two paths. The first path is directly from the sketch input routine. The second path is by pressing the EDIT key in Figure D.2 when the dimensioning scheme of the current part shape is to be modified (edited). The initial discussion will be geared toward the entry path from the sketch input routine. The modification procedures will be discussed at the end of Section III.

When control is passed to this routine from the sketch input section, the operator is presented with the display in Figure D.13. The previously defined shape is drawn at the bottom. The graphic keypad is only used to display prompts to the operator.



CONSTRAINTS REMAINING: 23

A B C D E F G H I 1 2 3
J K L M N O P Q R 4 5 6
S T U V W X Y Z CLR 7 8 9
SP Ø . ENT
HLIN VLIN RANG PDIS ANGL LDIS HDIS VDIS PLIN RADI AREA TDIS DEL DONE NEW DRAW

Figure D.13, Initial display in constraint definition routine.

The message in the top right corner indicates the number of constraints remaining that must eventually be defined. The number is updated each time a constraint is defined. If the number is positive, the shape is currently under-dimensioned. If the number is negative, the shape is currently over-dimensioned.

The definition of a dimensional constraint is accomplished by pressing one of the first 12 menu buttons and then specifying the entities to be constrained. The use of each menu button will be explained. Note-when any explicit dimension is defined, the number that is associated with the dimension is always displayed as zero. It is only used as a "place holder" for the actual value.

The HLIN and VLIN constraints:

The HLIN command allows the operator to constrain a line to be a "horizontal line", and VLIN to constrain a line to be a "vertical line". The operator first indicates the command by depressing and releasing the pen (called "digitizing"), in the menu box above the command. The next digitize can either be on a point or on a line. If it is on a line, as in Figure D.14, the definition is complete and an additional line is drawn to show that an implied constraint has been added.

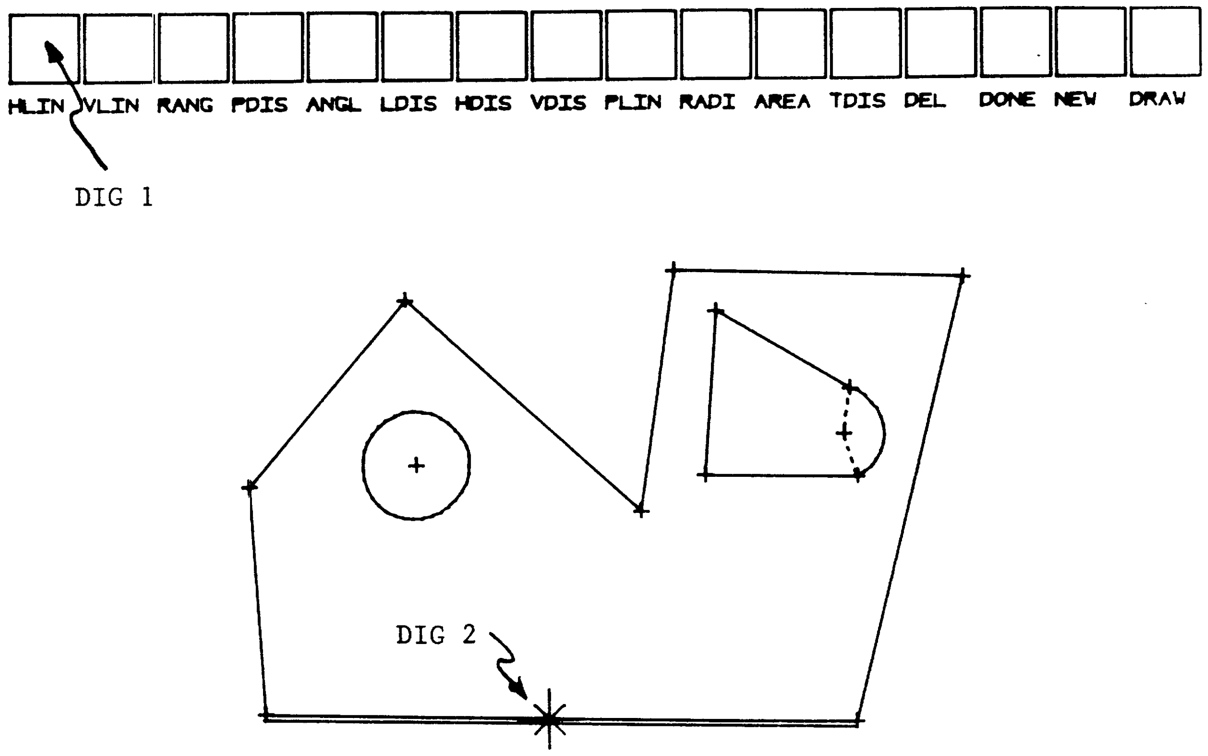


Figure D.14, One method of specifying a horizontal or vertical line.

If the second digitize is on a point, then a third digitize must be made on a second point. In the case of a HLIN command, the Y-coordinates of both these points will be constrained to be equal. Hence, if there was a line between these points (there doesn't have to be), the line would be constrained to be horizontal. In the case of a VLIN command, the X-coordinate of the two points would be constrained to be equal. This last mode of digitizing is shown in Figure D.15.

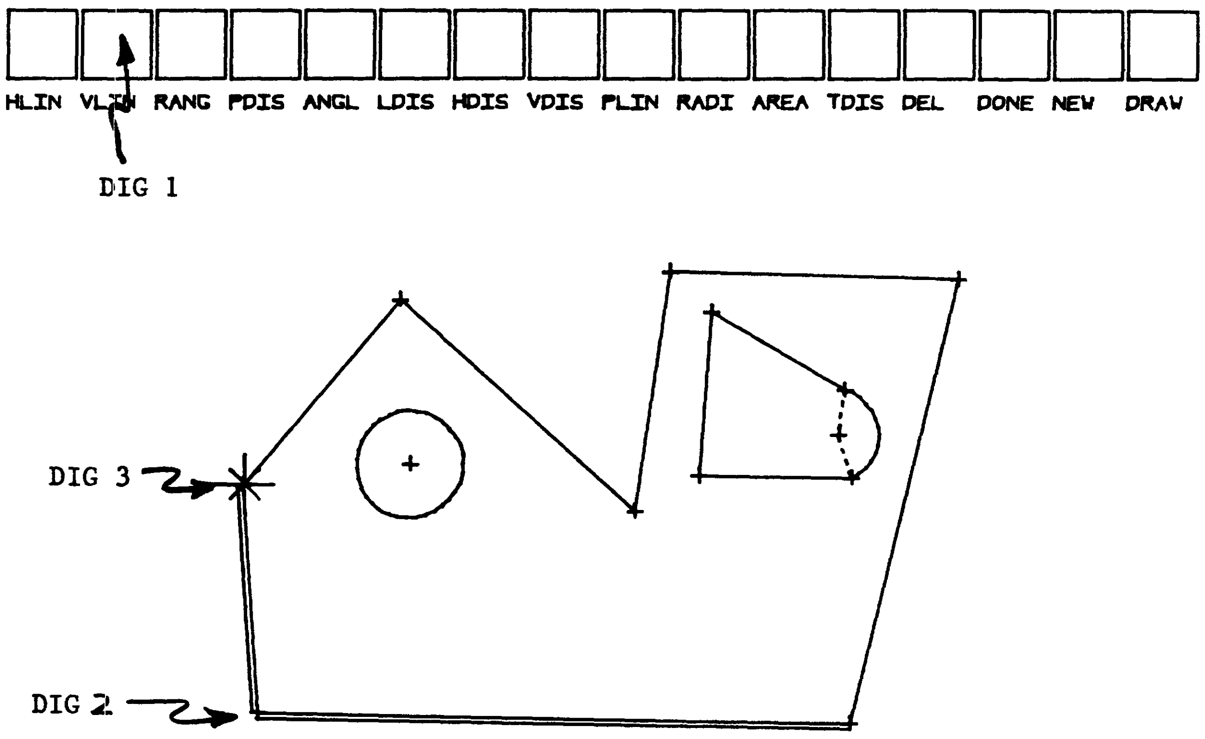


Figure D.15, Second method for constraining a horizontal or vertical line.

The RANG constraint:

The RANG command permits the operator to constrain two adjacent lines to form a "right angle" at their point of intersection. It is also used to constrain a line to be tangent to an arc. The operator must first digitize in the menu box above the RANG command. The next digitize is at a point. The point must either be the point of intersection of the lines which are to be perpendicular, or at the tangency point of a line to an arc. The constraint is displayed by the drawing of a box around the digitized point as shown in Figures D.16 and D.17.

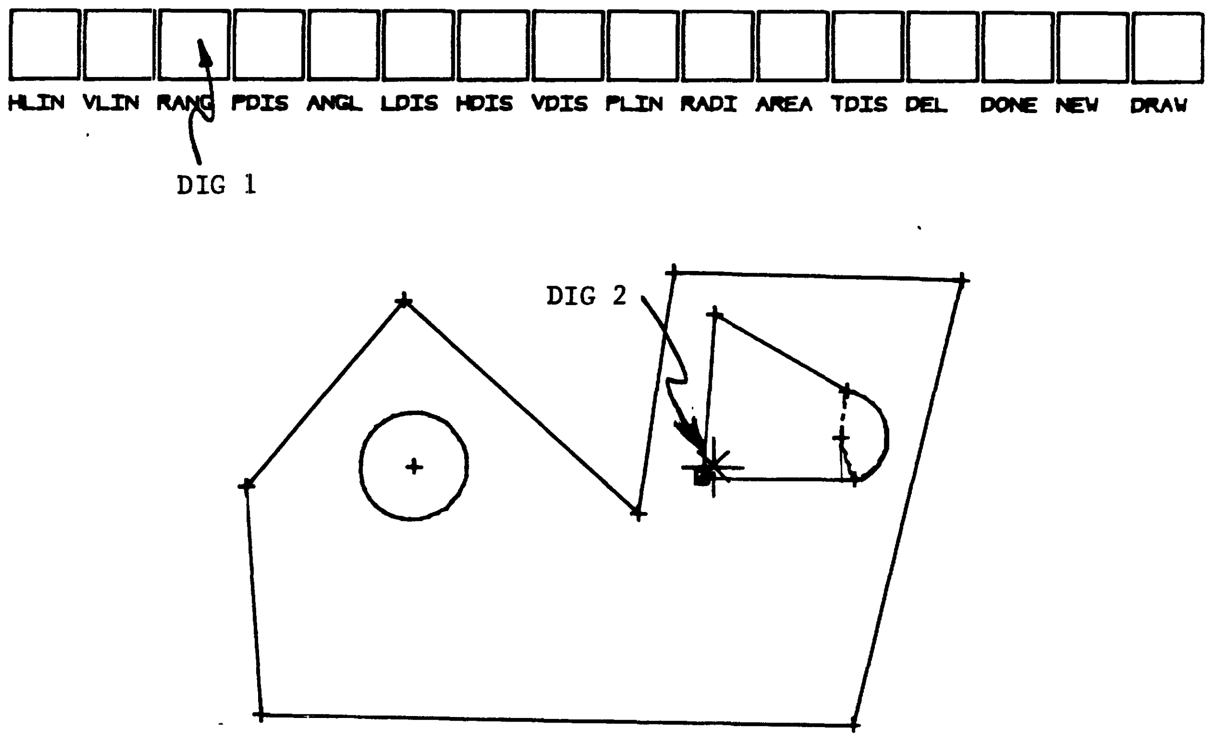


Figure D.16, Constraining a right angle.

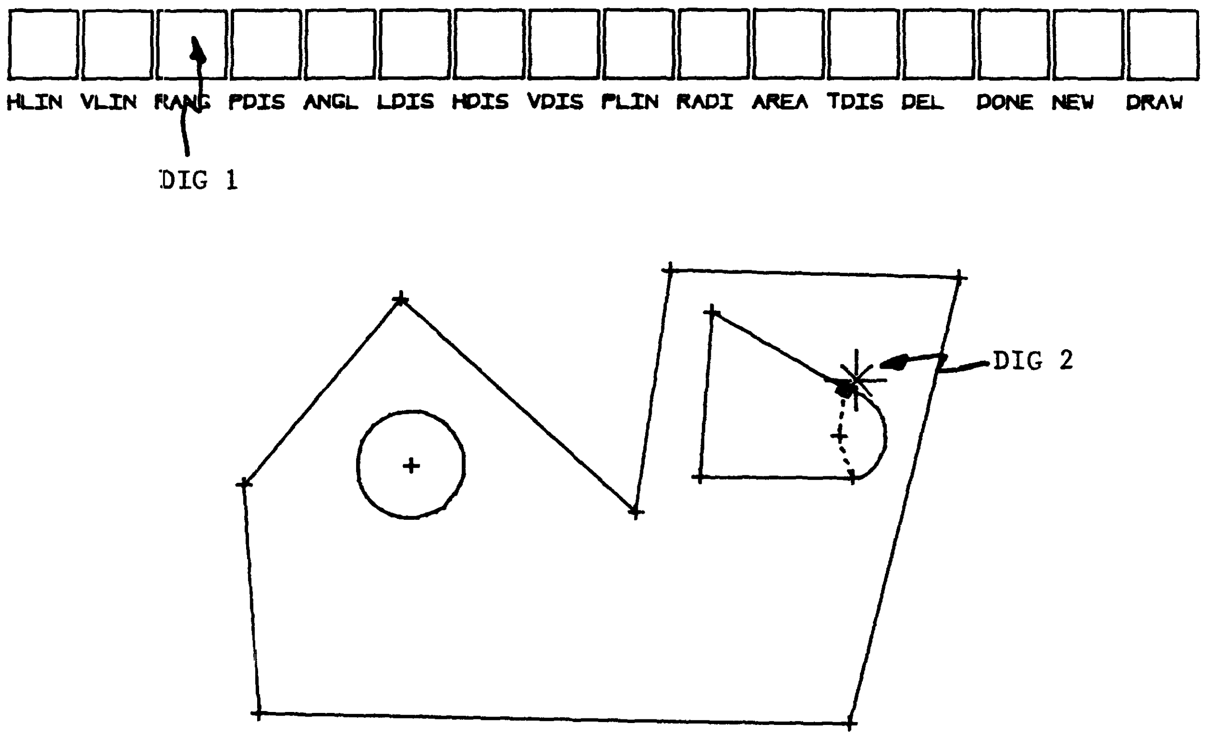


Figure D.17, Constraining a point of tangency.

The PDIS constraint:

The PDIS command allows the operator to constrain the "perpendicular distance" from a point to a line. The operator first digitizes the menu box above the PDIS command. The second digitize is the point and the third digitize is the line, as shown in Figure D.18.

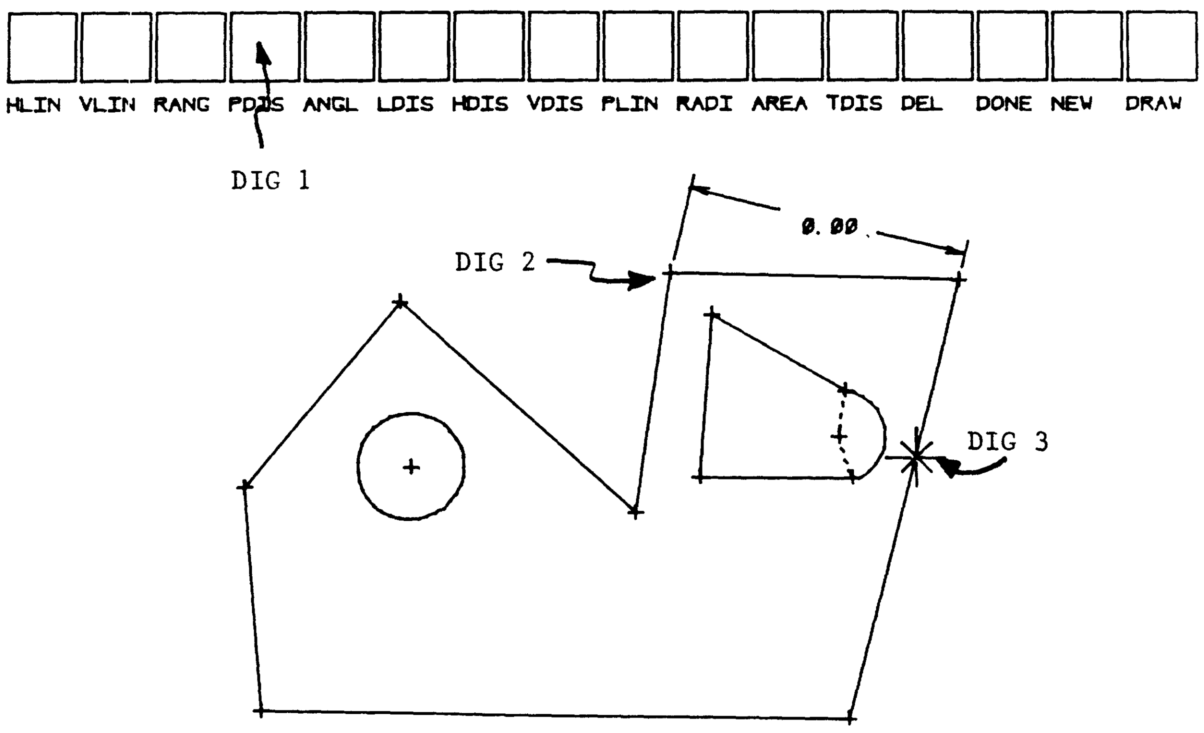


Figure D.18, Constraining the distance from a point to a line.

The ANGL constraint:

The ANGL command is used to constrain the "angle" between two lines. The insertion of an ANGL constraint, like HLIN and VLIN, can be done in two modes. In either case, the operator must first digitize the menu box above the ANGL command. If the angle is to be specified by digitizing lines, the operator digitizes the appropriate lines as shown in Figure D.19.

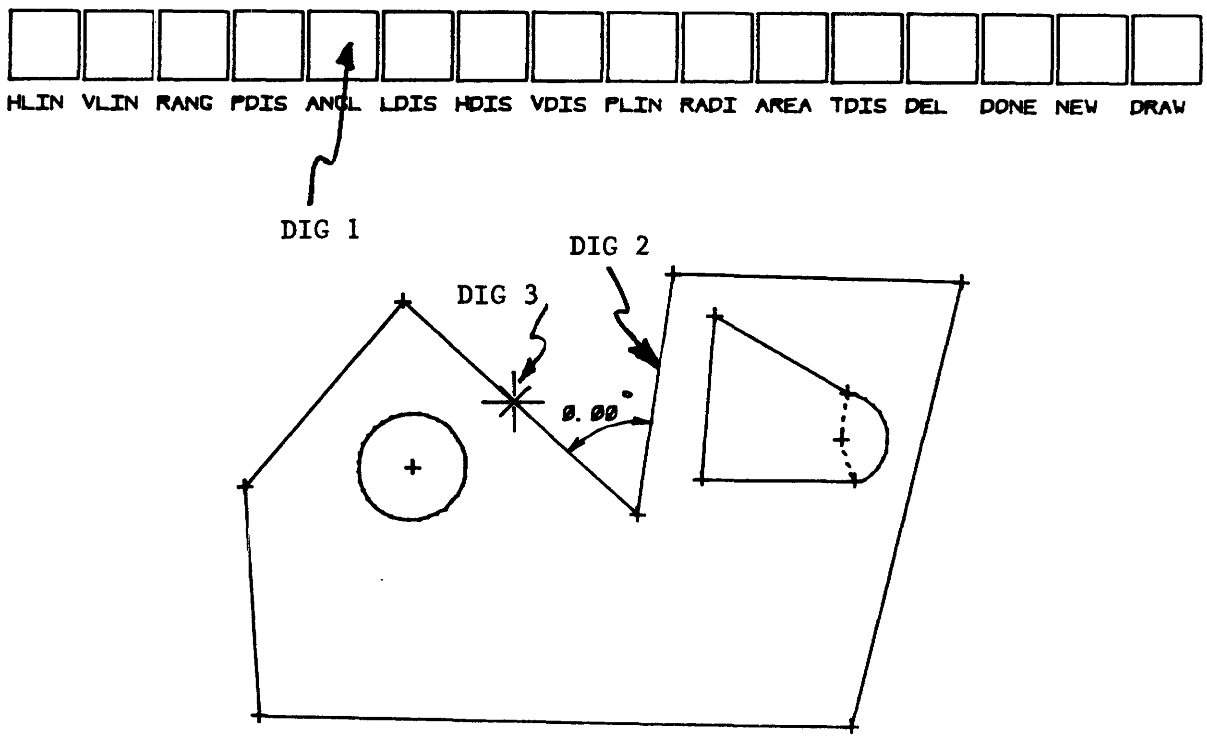


Figure D.19, Constraining an angle by digitizing the lines.

The angle is defined between the first line and the second line using a counter - clockwise convention from the first line to the second line. It must be noted that the definition of an angle extends from 0-180 degrees.

An angle can be defined between any two vectors as in Figure D.20, where the direction of V_{12} is from P_1 to P_2 and the direction of V_{34} is from P_3 to P_4. Note - there does not have to be a line between P_1 and P_2 nor between P_3 and P_4.

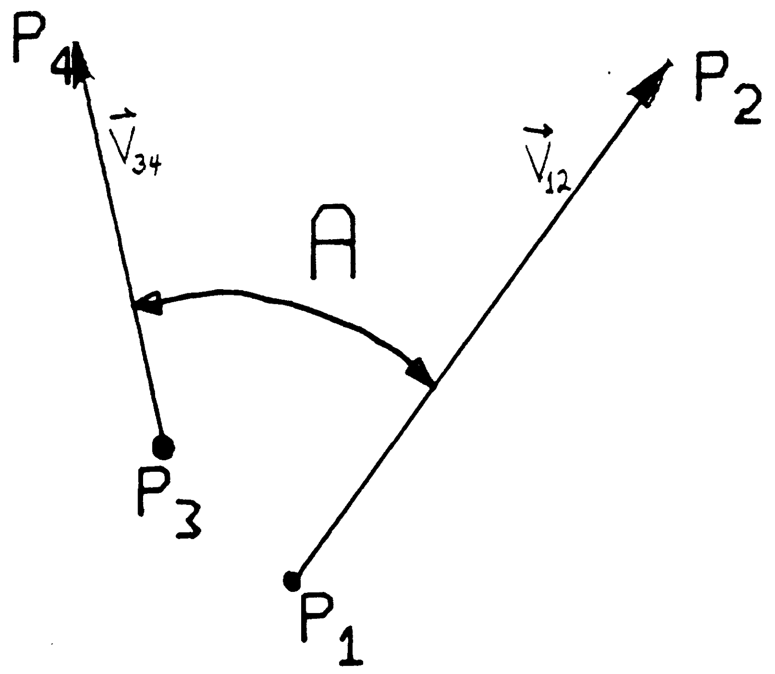


Figure D.20, Angle definition between any four points.

The direction convention for the angle is a counter-clockwise direction form V_{12} to V_{34}.

The operator must first digitize in the ANGL menu box and then digitize the four points P_1, P_2, P_3 and P_4. The angle definition in Figure D.21, was made using the indicated digitizes.

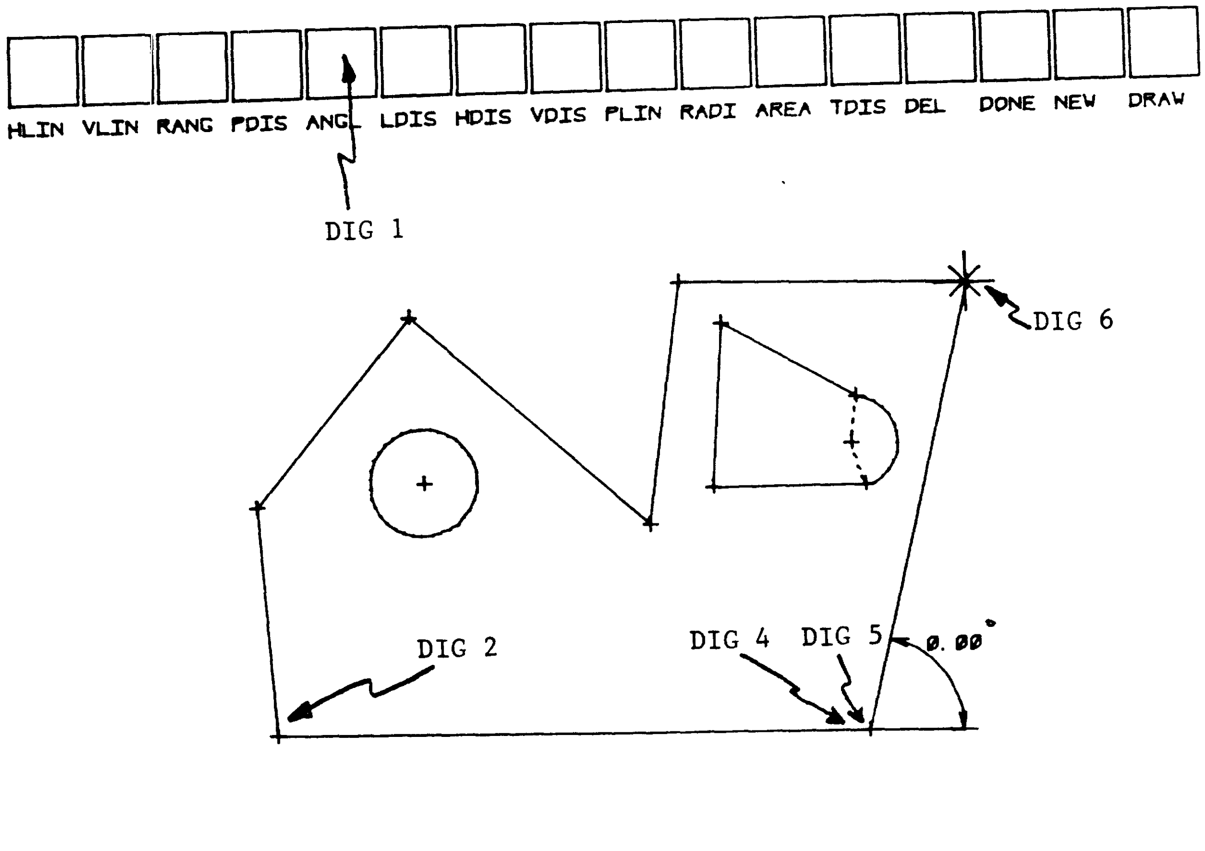


Figure D.21, Constraining an angle by digitizing four points.

The LDIS, HDIS and VDIS constraints:

These commands allow the operator to constrain the linear displacement (LDIS), the horizontal displacement (HDIS) or the vertical displacement (VDIS) between any two points. The operator must first digitize the desired menu button. Then, if a line is digitized as in Figure D.22, the two points constrained are the endpoints of the line.

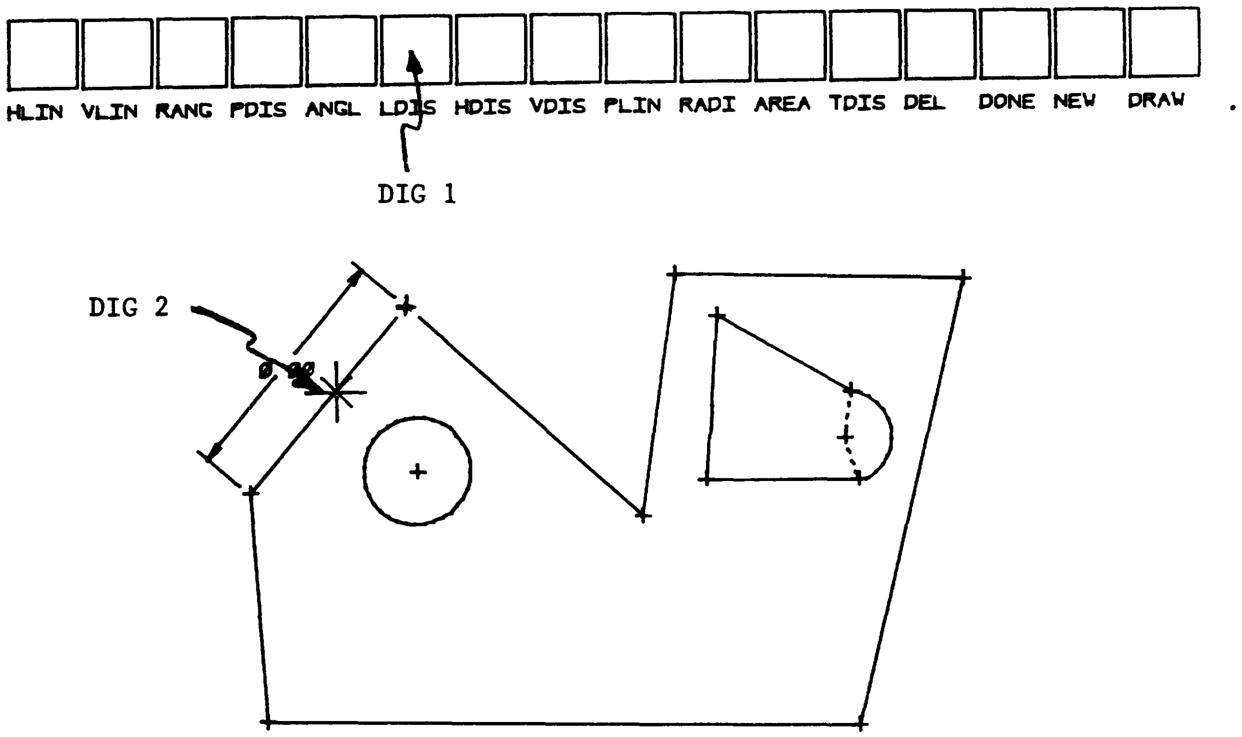


Figure D.22, Inserting an LDIS constraing by digitizing a line.

If two points are digitized as in Figure D.23, the constraint is defined between these two points. Note- the two points which are digitized do not have to be connected by a line.

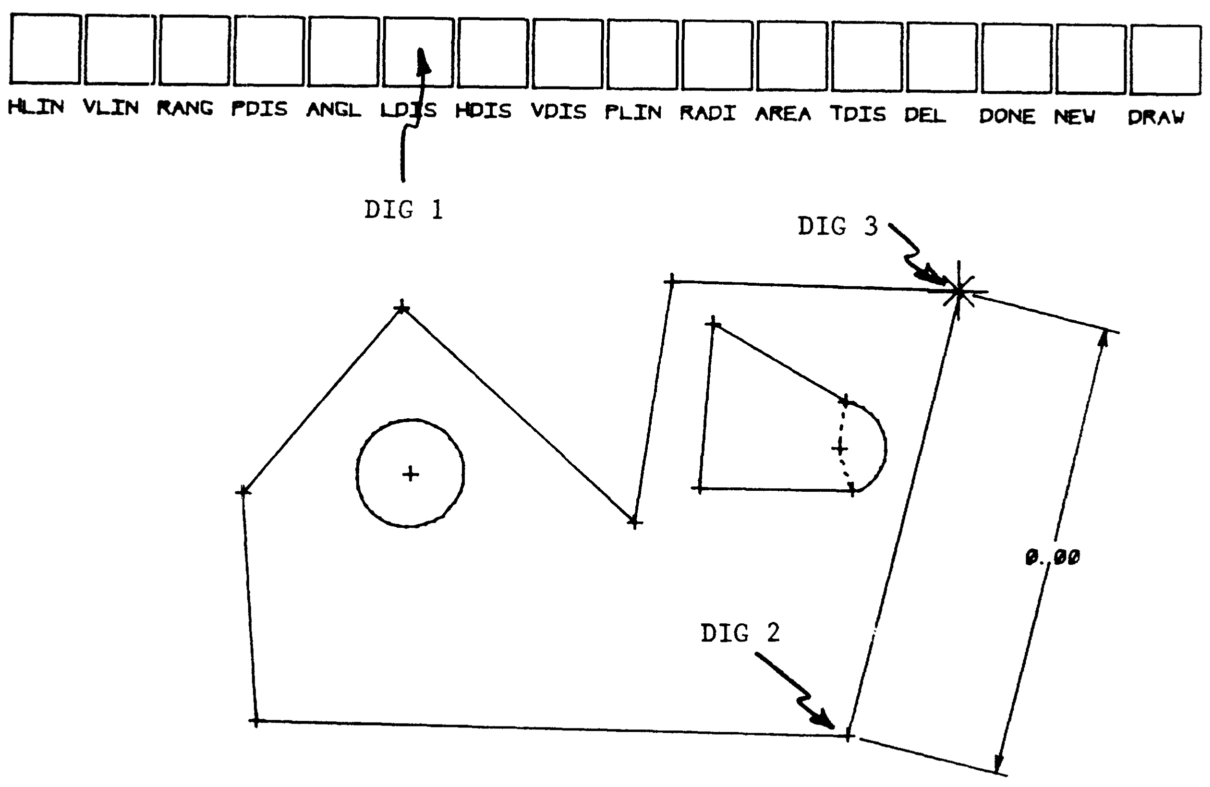


Figure D.23, Inserting an LDIS constraint by digitizing any two points.

The PLIN constraint:

The operator may contrain two lines to be parallel by using the PLIN constraint. The first digitize must be within the PLIN menu button. The second digitize is on one of the line segments and the third digitize is on the other line segments as shown in Figure D.24.

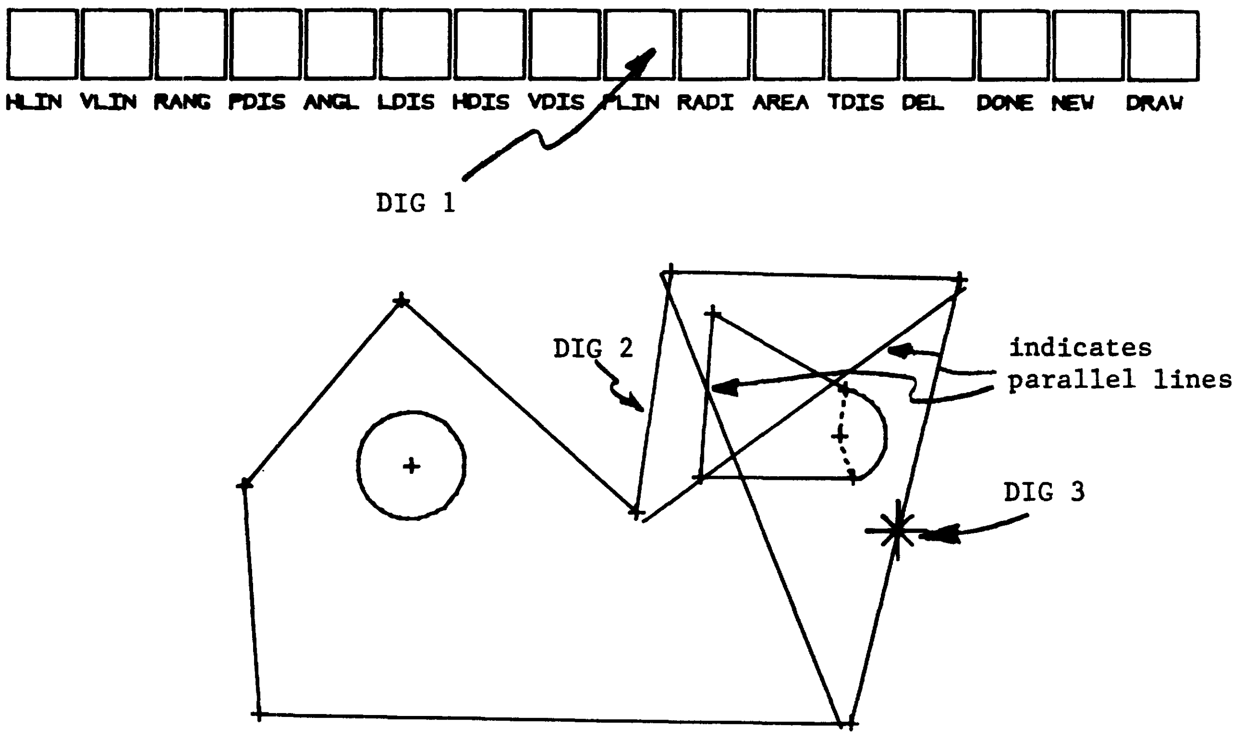


Figure D.24, Constraining parallel lines.

The AREA constraint:

The operator is permitted to constrain the surface area of any shape by using the AREA constraint. The only digitize that is required is in the AREA menu button.

In order to indicate that the area has been constrained "AREA= 0.00" is written in the bottom left corner as shown in Figure D.25.

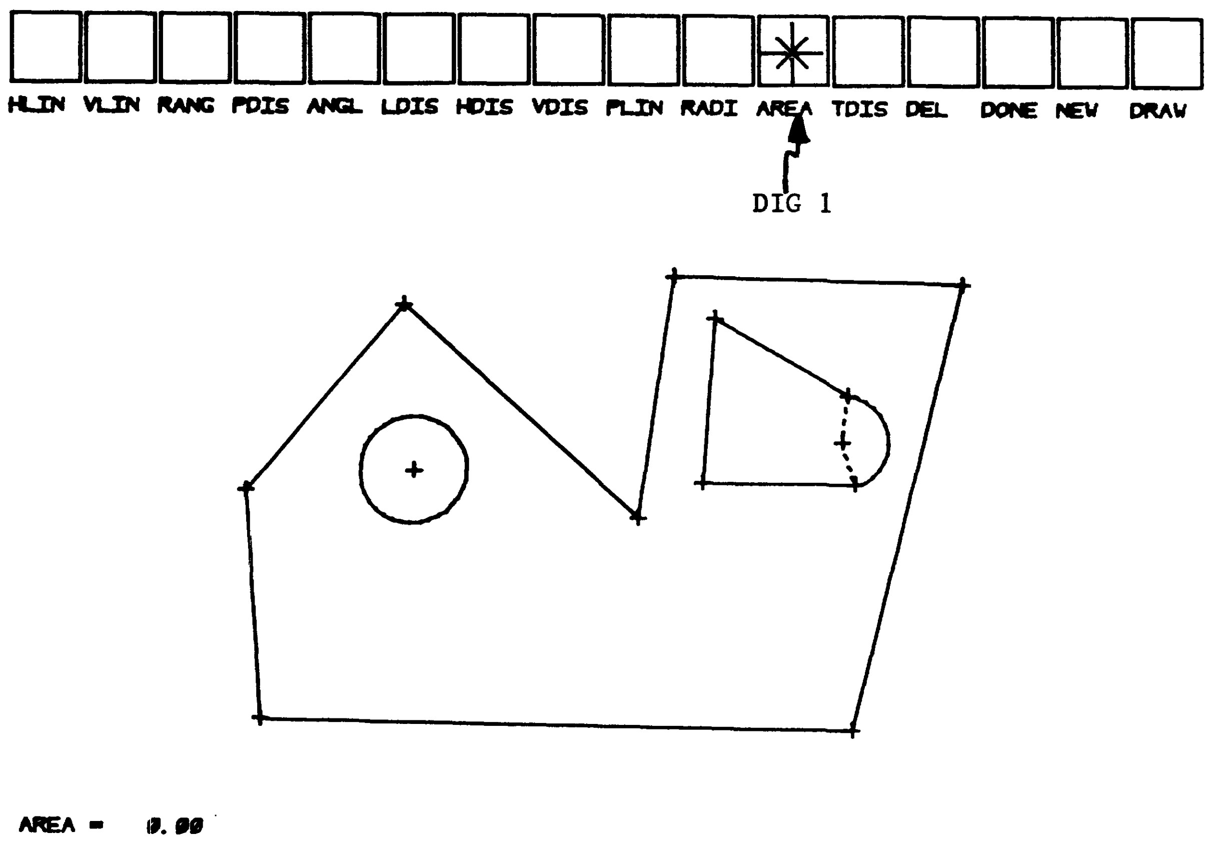


Figure D.25, Constraining the surface area.

The RADI constraint;

This constraint defines a radial measure between the points on an arc and its center. This will permit the operator to specify the radius of an arc. Two digitizes are required as shown in Figure D.26. The first digitize is in the RADI menu box and the second is on the desired arc.

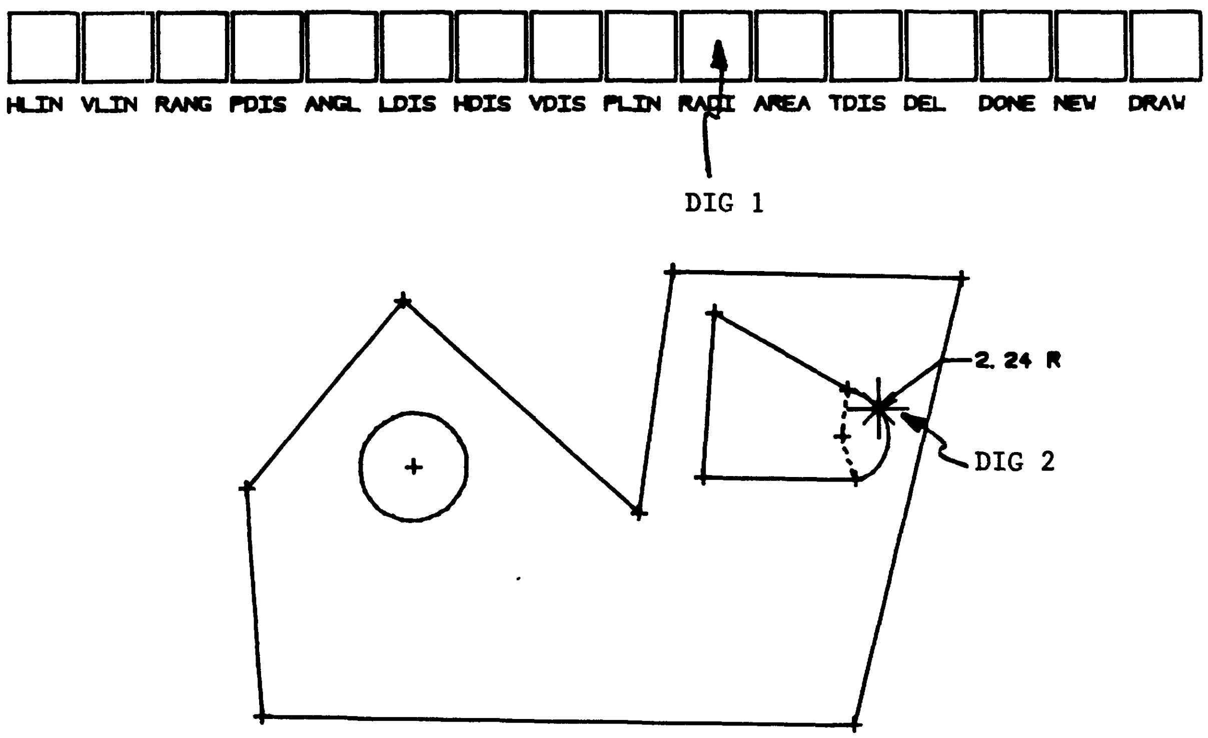


Figure D.26, Specifying a radius.

The TDIS constraint;

For the lack of a better name, the TDIS constraint is used to constrain the distance between a point and the extreme point on an arc or between two arcs. This constraint will permit the dimensioning schemes of Figure D.27 to be used.

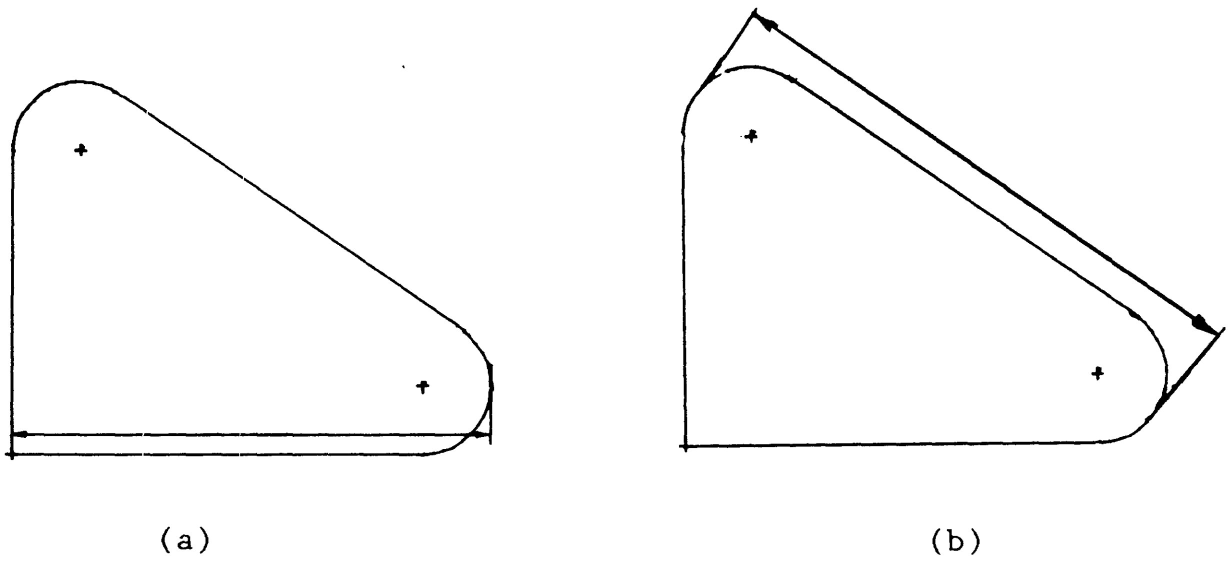


Figure D.27, Dimensioning between: (a) a point and an arc, and (b) between two arcs.

Three types of TDIS constraints may be defined: horizontal, vertical or linear displacement. In order to specify the constraint, the operator digitizes the TDIS menu box and then digitizes either the HDIS, VDIS or LDIS menu boxes. At this point, the operator digitizes the two entities to be constrained. These two entities must either be a point and an arc or two arcs. For example, the dimensioning scheme of Figure D.27a and D.27b may be specified as shown in Figures D.28 and D.29 respectively.

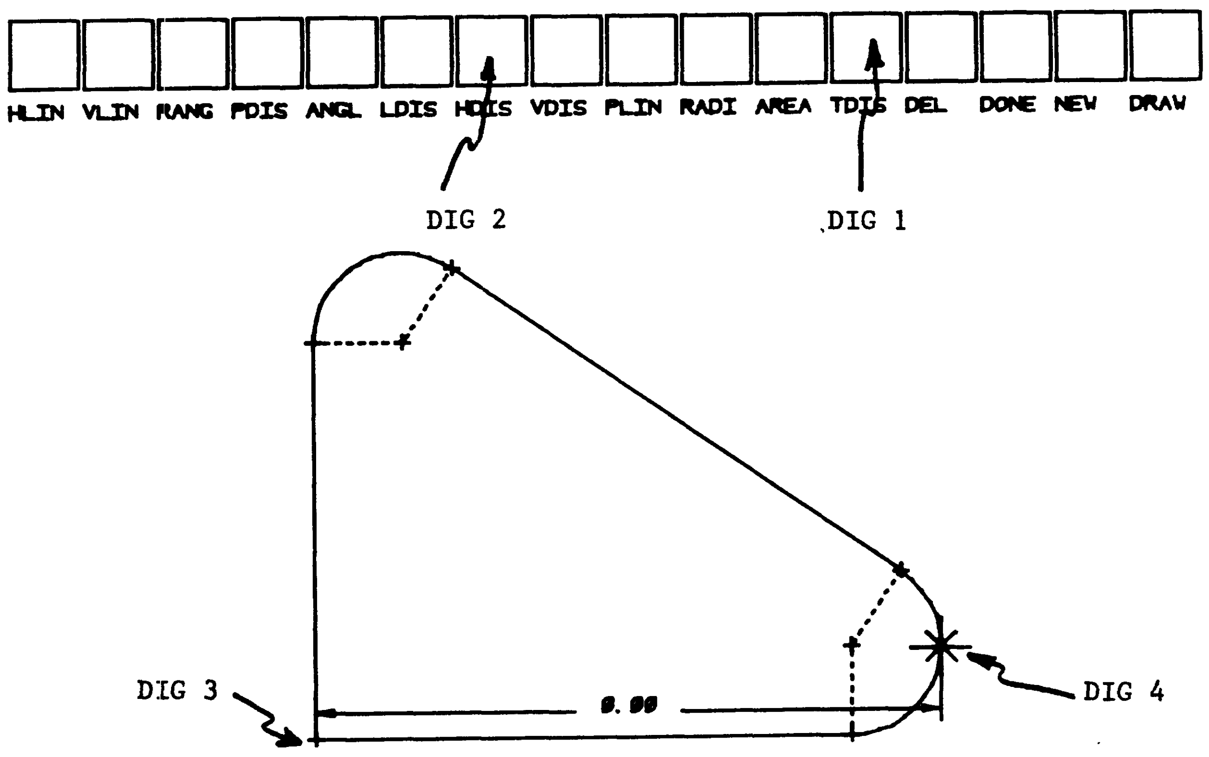


Figure D.28, Specification of a horizontal dimension between a point and an arc.

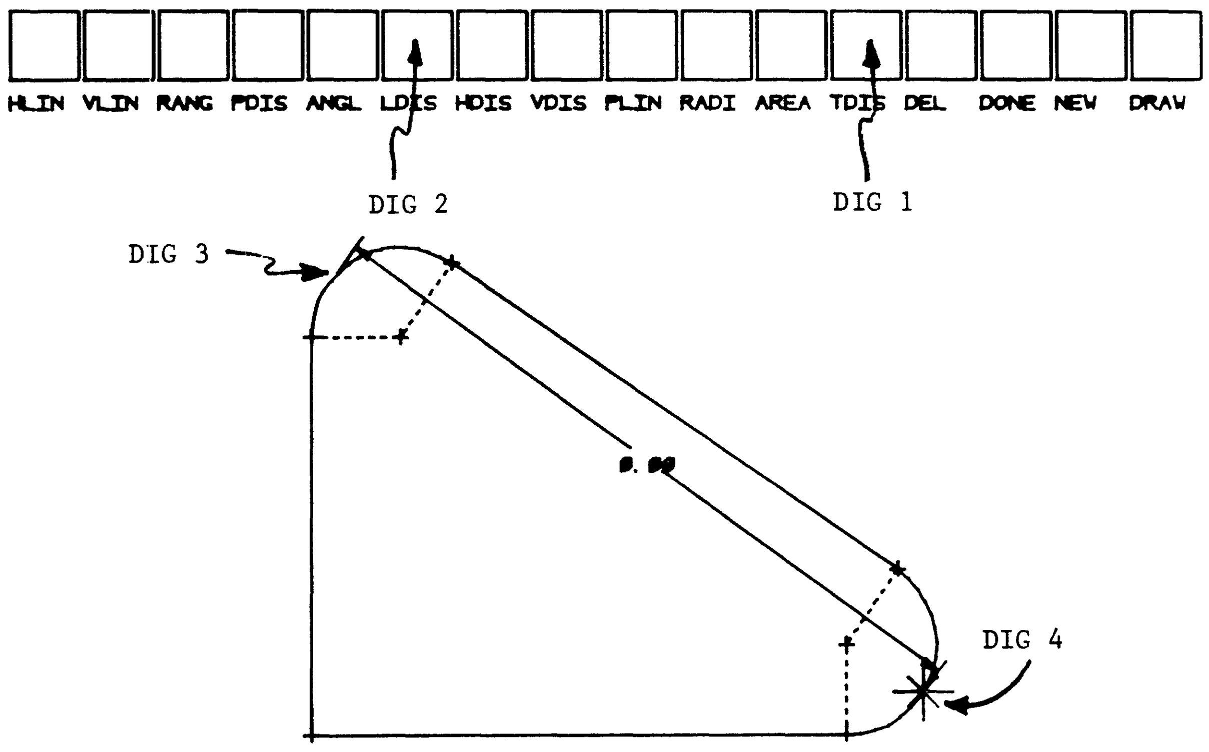


Figure D.29, Specification of a linear dimension between two arcs.

OTHER MENU COMMANDS

The DEL command:

This command is used to delete any constraint from the dimensioning scheme. If the constraint that is to be deleted has a number associated with it, all that is required is that the operator digitize within the DEL menu box and then digitize at the position of the number associated with the desired constraint (see Figure D.30).

If the constraint that is to be deleted is an implied constraint such as HLIN, VLIN, RANG, or PLIN, the first digitize must be within the DEL menu box, the second digitize within the menu box of the desired type of constraint and then the third digitize must be at one of the points involved with the constraint (see Figure D.31). If a mistake is made at any one of these digitizes, the deletion sequence must be restarted.

A plus sign is drawn over the dimensional constraint that was deleted to indicate that the deletion was successful. In order to update the display with all the deletions erased, the operator must digitize within the DRAW menu box. The deleted dimensional constraints will be erased.

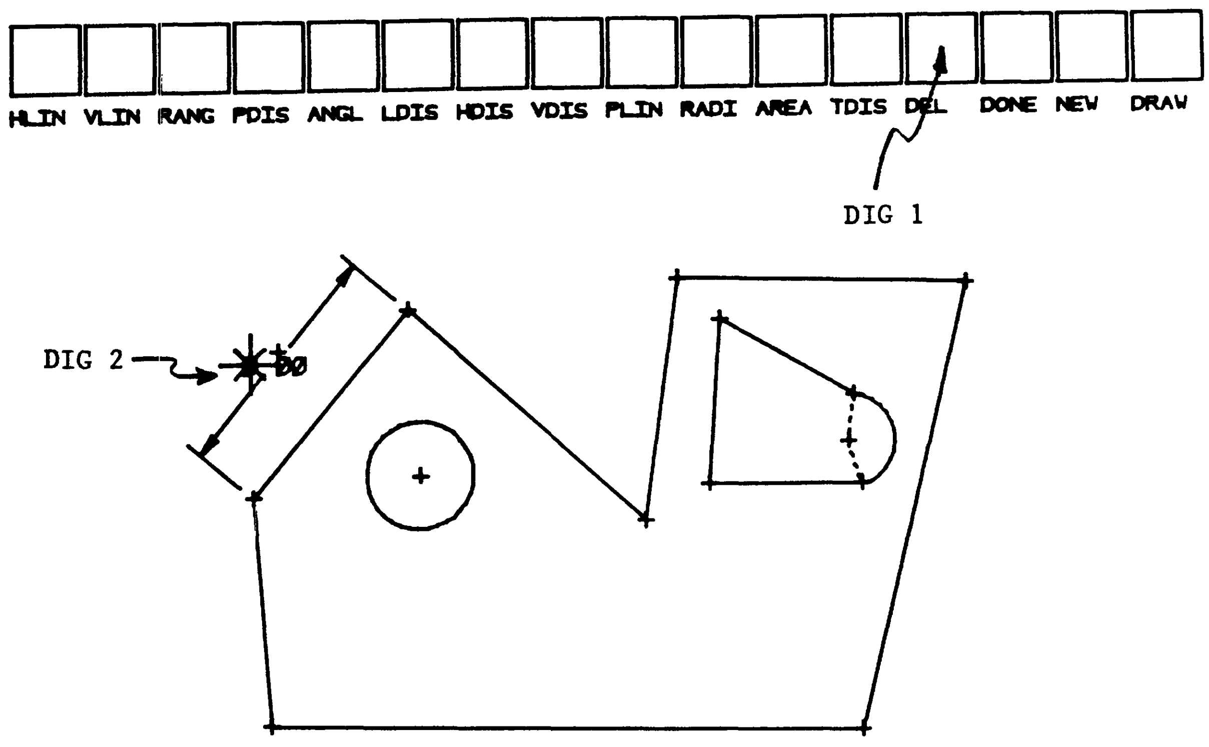


Figure D.30, Deletion of an explicit dimensional constraint.

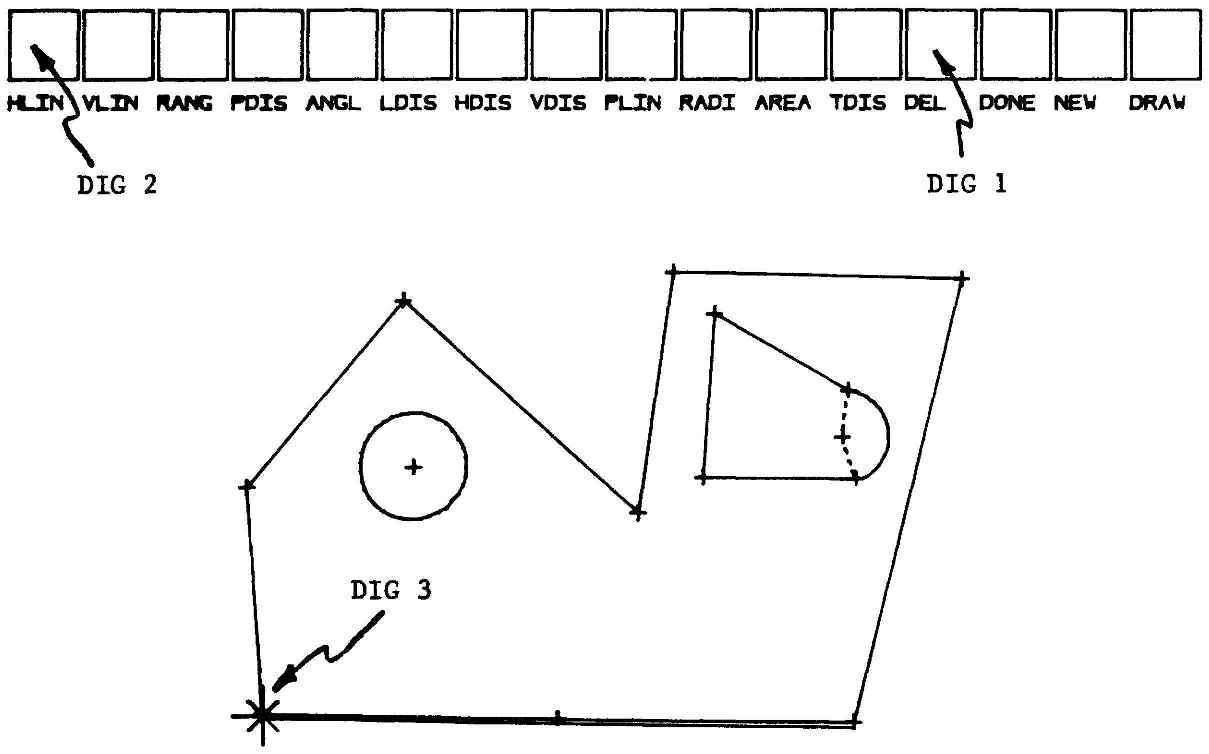
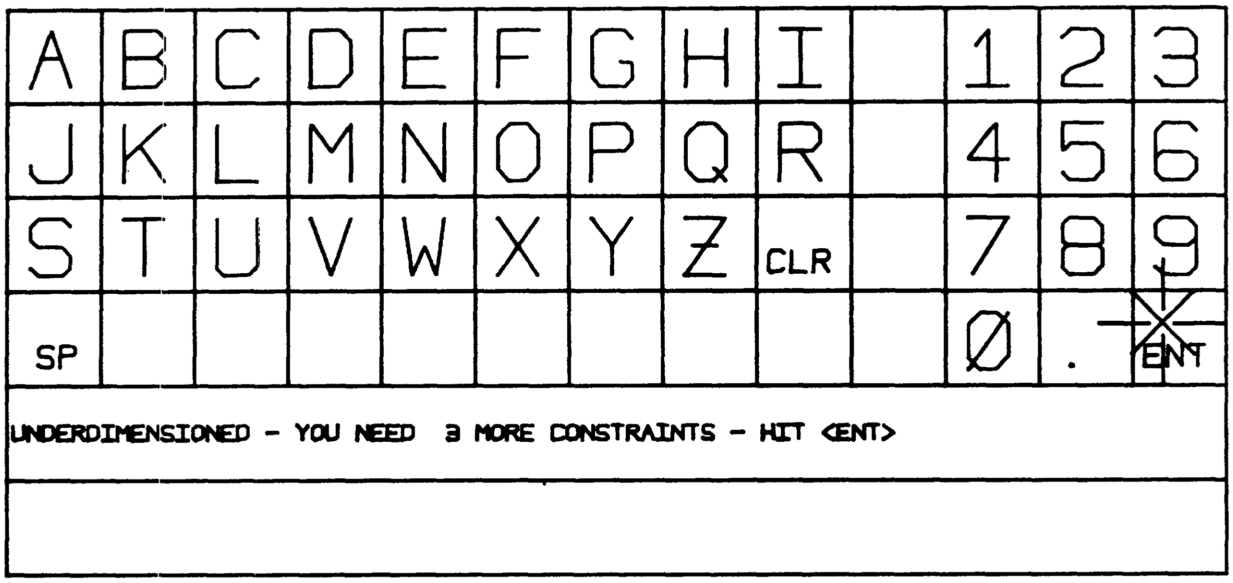


Figure D.31, Deletion of an implied constraint.

The DONE command;

This command allows the operator to leave the dimension specification routine and go to the dimensional modification routine. If the number of constraints remaining is not zero, the user is notified via the graphic keypad as in Figure D.32.



UNDERDIMENSIONED - YOU NEED 3 MORE CONSTRAINTS - HIT <ENT>

Figure D.32, Notification of over/under dimensioning.

If the part is over dimensioned, the user is not allowed to leave this routine. The operator is allowed to leave the routine if the part is sufficiently or under dimensioned (ie.- the number of constraints remaining is positive or zero).

The NEW command:

This command is used to shift control back to the sketch input section. A new part can now be drawn. Note- the current part will be destroyed in the process and cannot be recovered.

The DRAW command:

This command allows the user to redraw the current display. Any deletions that have been done since the last DRAW will be erased. If no deletions have occured, this command will have no effect on the display.

SECTION IV. Modification of Dimensional Values.

The purpose of this routine is to permit the operator to change the numeric values of the explicit dimensions. Program control is shifted to this section via two paths. The first path results from digitizing the "OLD" menu button in Figure D.2, a previously stored data set is read in and control is passed to this routine. The second path is directly from the constraint definition routine described in Section III.

When control is passed to the routine, the current shape, dimensions, command menu, and graphic keypad are displayed as in Figure D.33. If the shape was just sketched, the part shape is still "rough". In other words, the tangencies are not actually tangent, the horizontal lines are not horizontal, etc. (even though they were constrained to be tangent, horizontal, etc.). In order to "straighten up" the part shape, the operator must execute the numerical procedures (the Newton-Raphson method). This is accomplished by digitizing the "GO" menu button. The part should then be "straightened up" as in Figure D.34.

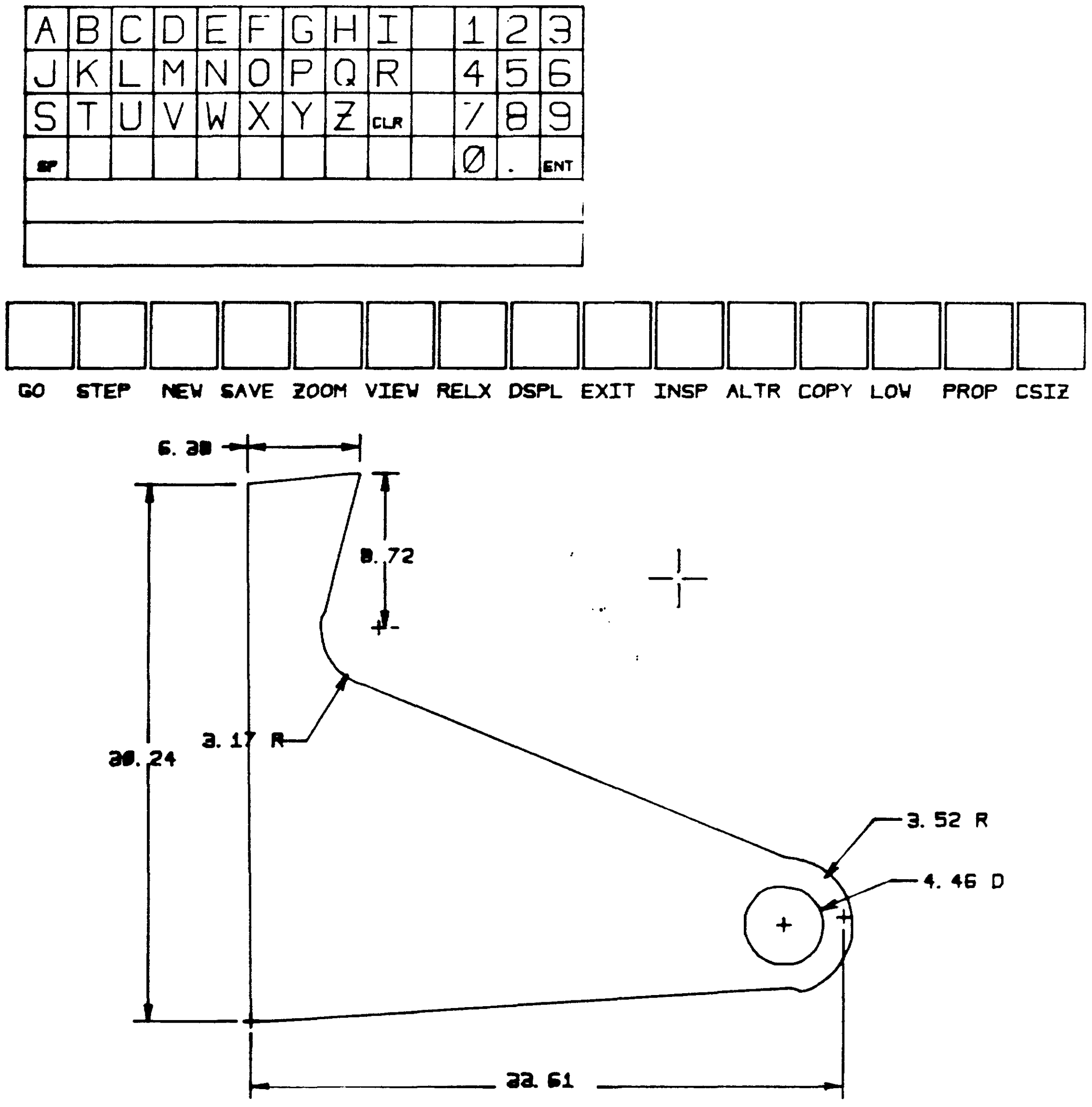

A B C D E F G H I 1 2 3
J K L M N O P Q R 4 5 6
S T U V W X Y Z CLR 7 8 9
SP Ø . ENT
GO STEP NEW SAVE ZOOM VIEW RELX DSPL EXIT INSP ALTR COPY LOW PROP CSIZ
6.30
0.72
3.17 R
30.24
3.52 R
4.46 D
32.61

Figure D.33, Initial display.

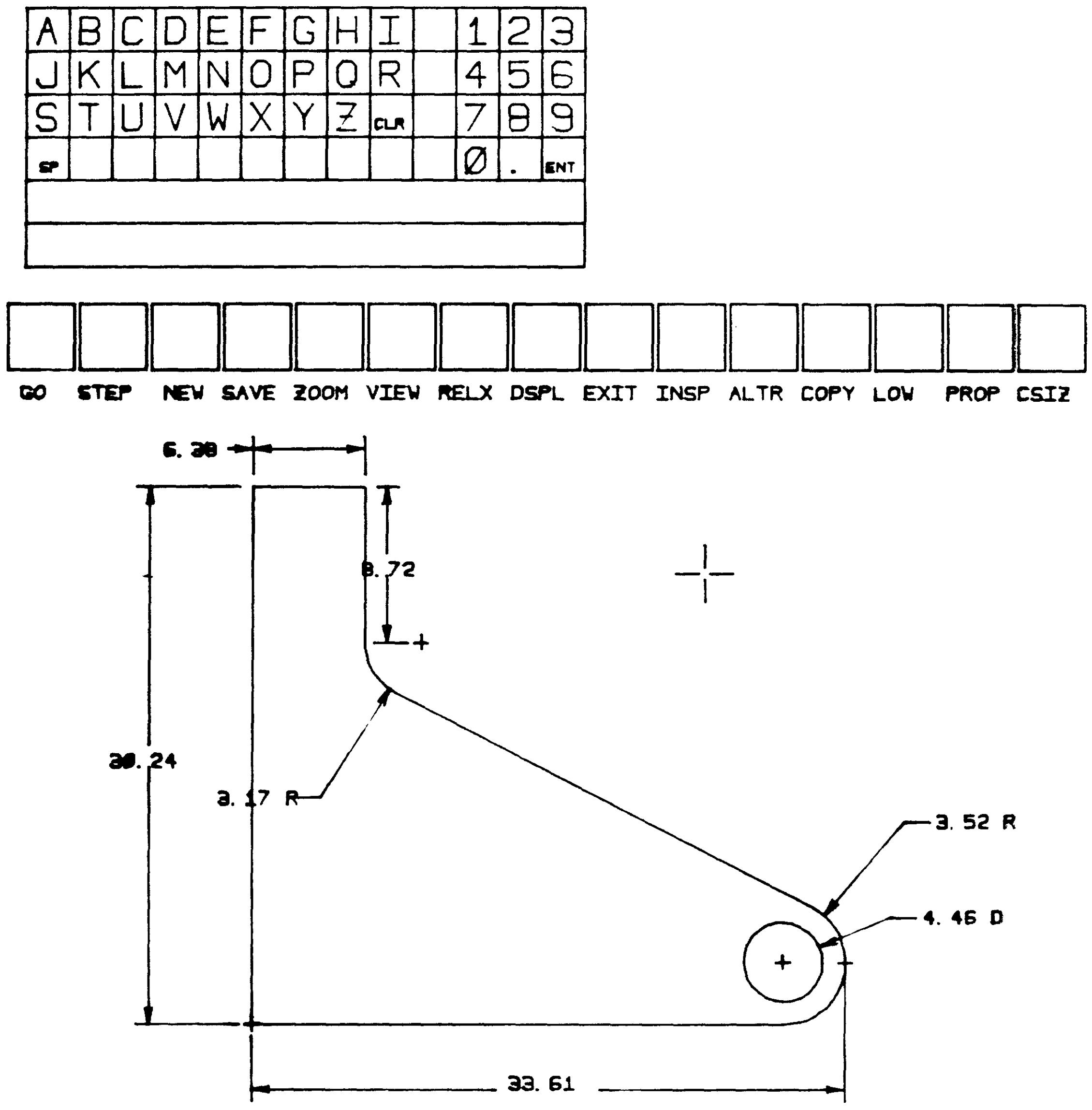

A B C D E F G H I 1 2 3
J K L M N O P Q R 4 5 6
S T U V W X Y Z CLR 7 8 9
SP Ø . ENT
GO STEP NEW SAVE ZOOM VIEW RELX DSPL EXIT INSP ALTR COPY LOW PROP CSIZ
6.38
8.72
30.24
3.17 R
3.52 R
4.46 D
33.61

Figure D.34, Part shape after being "straightened".

Changing the numeric values of a dimension:

This is the default function in this routine. If the system expects no other digitized input, the operator is able to change the numeric value of any dimension (including the area if it was constrained) by simply digitizing at the location of the number associated with the desired dimension (see Figure D.35).

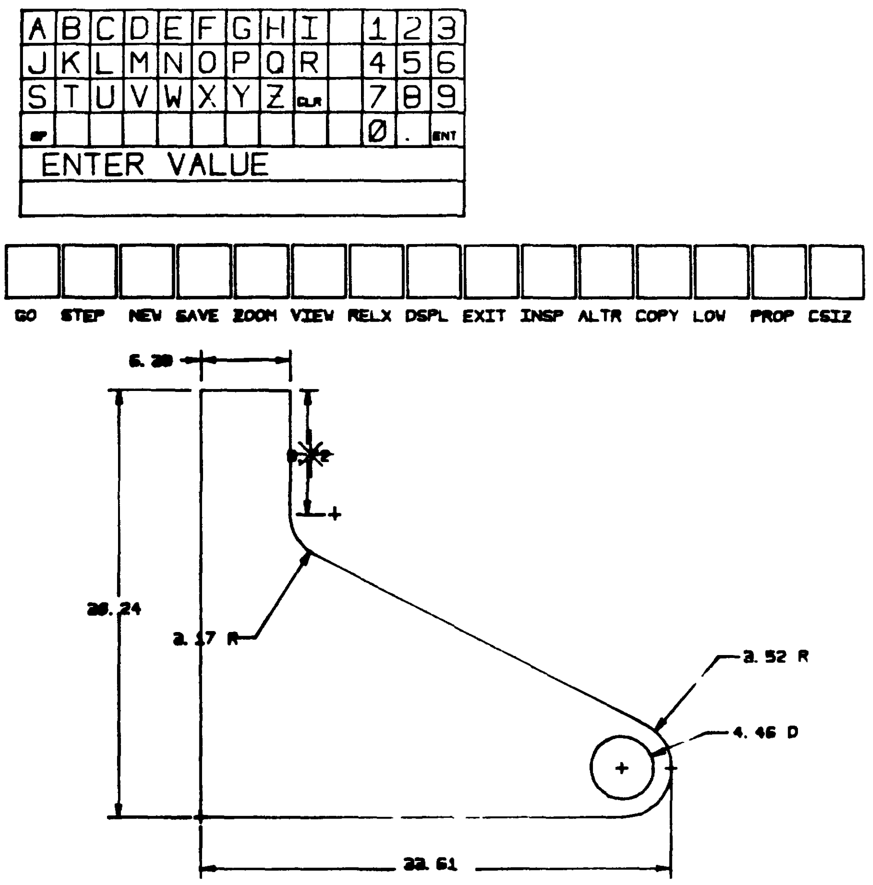


Figure D.35, Indicating which dimension is to be modified.

This indicates which dimension is to be modified. A prompt appears in the graphic keypad area (see Figure D.36) and the operator, using the pen, digitizes the graphic "keys" to specify the desired value for the dimension. The enter key, ENT, is digitized to terminate the data entry. The CLR key, is used to clear and re-input the number.

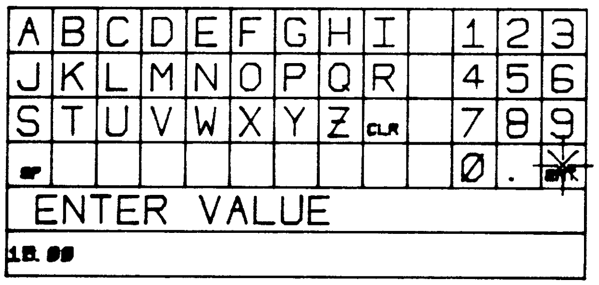


Figure D.36, Graphic keypad data entry.

The GO command;

When the "GO" menu button is digitized the Newton - Raphson method is executed to cause the current geometry to change in such a way that the geometry is compatible with the dimensional values. Therefore, with the previously made change in dimensions, when the GO menu button is digitized, the shape in Figure D.37 is produced. Note that the effect of the dimensional change was the correct alteration in geometry.

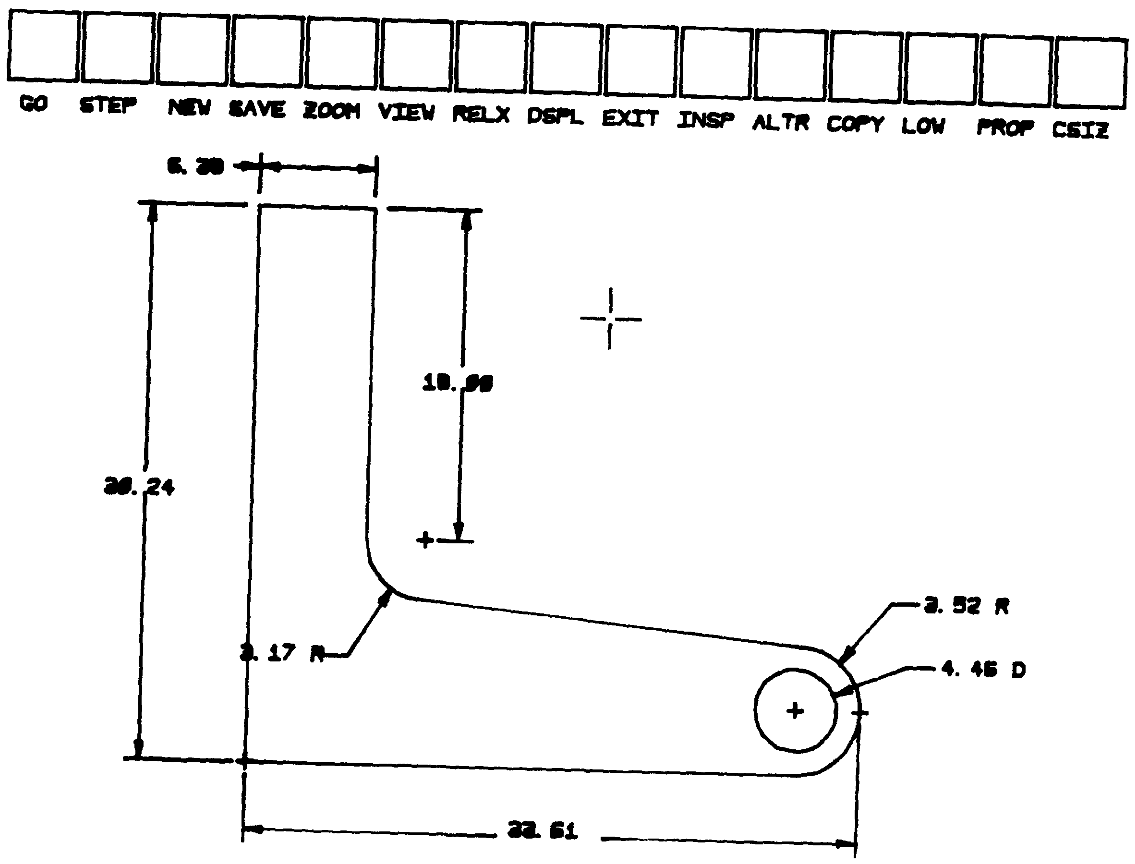


Figure D.37, Result of a dimensional change.

The use of the STEP, RELX and LOW commands:

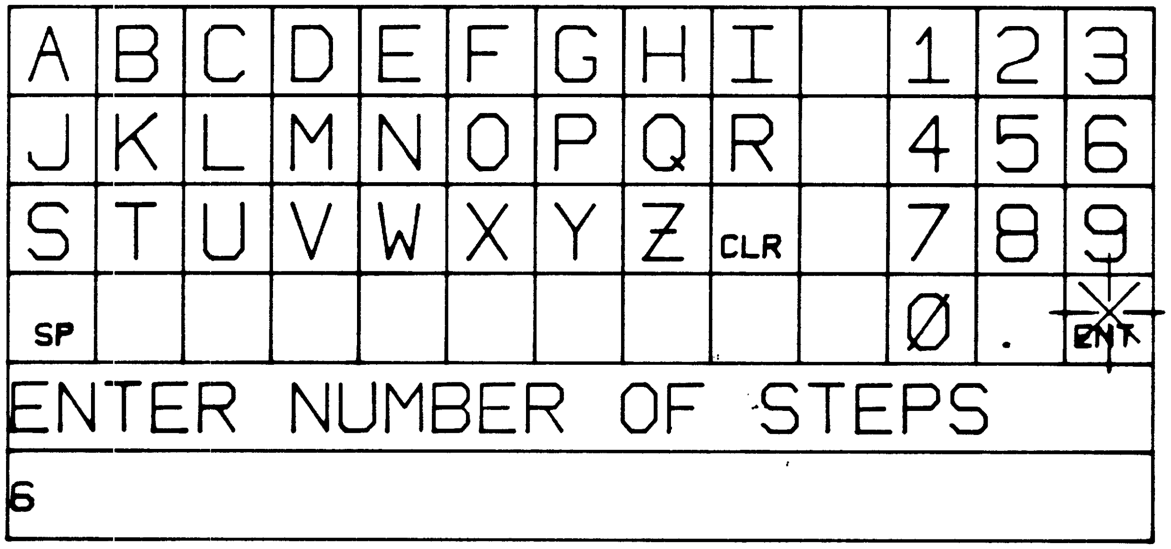
Unfortunately, the Newton - Raphson iteration does not always "work" the same for all shapes, dimensioning schemes and dimensional values. The numerical method used is an iterative method which causes the convergence of a set of equations to produce a new geometry based on the dimensional values. By the word "work", it is meant that for some shapes and dimensioning schemes, and depending on the size of the dimensional change, some modifications must be made to the iterative procedures. Only a qualitative description of the use of these techniques will be presented in this document. For an indepth treatment, the reader is advised to consult the author's Masters thesis.

The STEP command:

This is probably the most widely used of the three commands. It allows the user to specify how many increments will be made in the dimensional values to get from the current set of values to the final set of values. Therefore, the equivalent of changing a dimensional value from 10 units to 15 units in 5 steps and then digitizing the GO button, is to change the dimension from 10 to 11 (then press GO), then from 11 to 12 (then press GO) and so on, until the desired final dimension of 15 units is reached.

Thus it is desirable, if a lot of dimensions are being changed at once or if the desired dimensional value is very much different from the current dimensional value, to use a value for the number of steps between 5 and 10. Experience is neccessary to get a feel for the number of steps needed for any dimensional change.

In order to specify the number of steps, the user first digitizes the STEP menu button. Then a prompt is written in the graphic keypad area. The number of steps is then input using the graphic keypad and digitizing pen.



ENTER NUMBER OF STEPS

6

Figure D.38, Specifying the number of steps.

The default value for the number of steps is 1.0 and is reset after every pressing of the "GO" button.

The RELX command:

This command allows the operator to specify a "relaxation" factor to be used in the next execution of the Newton - Raphson method. If for some reason, the part shape and dimensioning scheme, tends to be unstable, a relaxation factor less than 1.0 in value may be specified (a good value might be around 0.7). The operator specifies the relaxation factor in the same way as the STEP factor. The RELX menu button is digitized and the relaxation factor is entered using the graphic keypad and digitizing pen. The default value of the relaxation factor is 1.0 and is reset each time the GO menu button is digitized.

The LOW command:

If the STEPS and RELX commands fail to cause the shape to converge, the operator may select a "low gear" method of solution which is much slower than the Newton - Raphson method but also much more stable. To indicate that the "low gear" numerical procedures are to be used, the operator simply digitizes the LOW menu button. A message is displayed in the graphic keypad area to indicate that the "low gear" option has been selected. The next time the GO button is digitized, the method of solution will use the "low gear" numerical method. The STEP and RELX parameters can also be set but it is advisable to set them both to 1.0 .

At the end of the iteration (when LOW is chosen), it is possible that the message "DID NOT CONVERGE - DO YOU WANT TO KEEP THIS SHAPE? (Y/N)" will be displayed in the graphic keypad area. If the shape looks as if it has converged, enter YES. This is due to a lack of accuracy at the end of the iteration. Then use the normal numerical method to get the solution to converge to the exact final shape.

Redundant and under dimensioned parts;

As explained in section II, it is not possible to enter this routine with an over dimensioned part. It is possible, however, to have a redundant or under dimensioned part. A redundant dimensioning scheme results when a section of the part is over - constrained while some other section is under-constrained. An under-dimensioned part simply lacks the correct number of dimensions.

The part shape in Figure D.39, is both under-dimensioned and redundantly dimensioned. When the "GO" button is digitized, the part shape will be straightened as much as possible. The redundant dimensions will blink and the under-constrained coordinates (X and/or Y) are indicated as in Figure D.40.

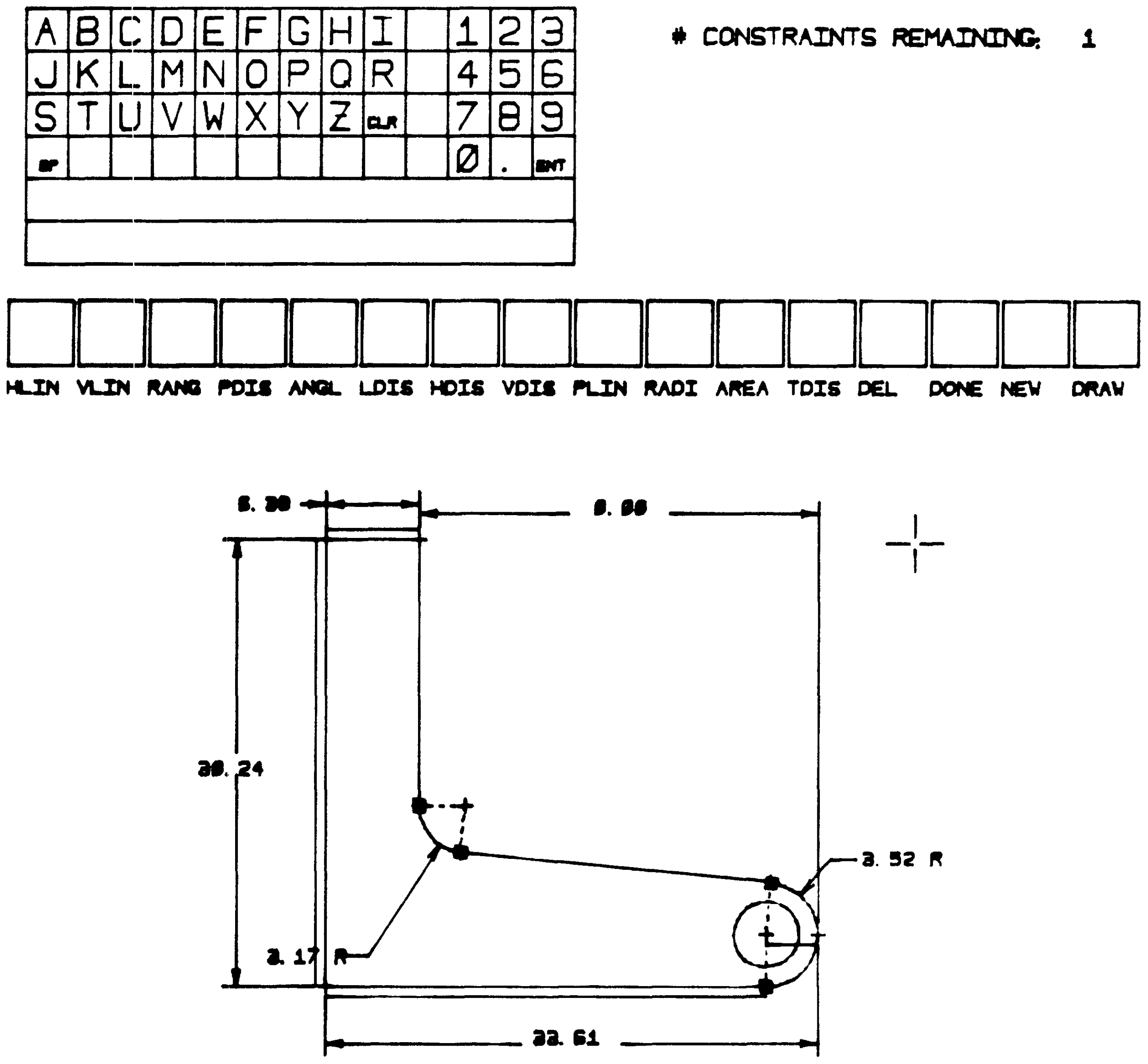

A B C D E F G H I 1 2 3
J K L M N O P Q R 4 5 6
S T U V W X Y Z CLR 7 8 9
BP Ø . ENT
# CONSTRAINTS REMAINING: 1
HLIN VLIN RANG PDIS ANGL LDIS HDIS VDIS PLIN RADI AREA TDIS DEL DONE NEW DRAW
6.30
8.00
30.24
3.52 R
3.17 R
22.61

Figure D.39, An under- and redundantly dimensioned part.

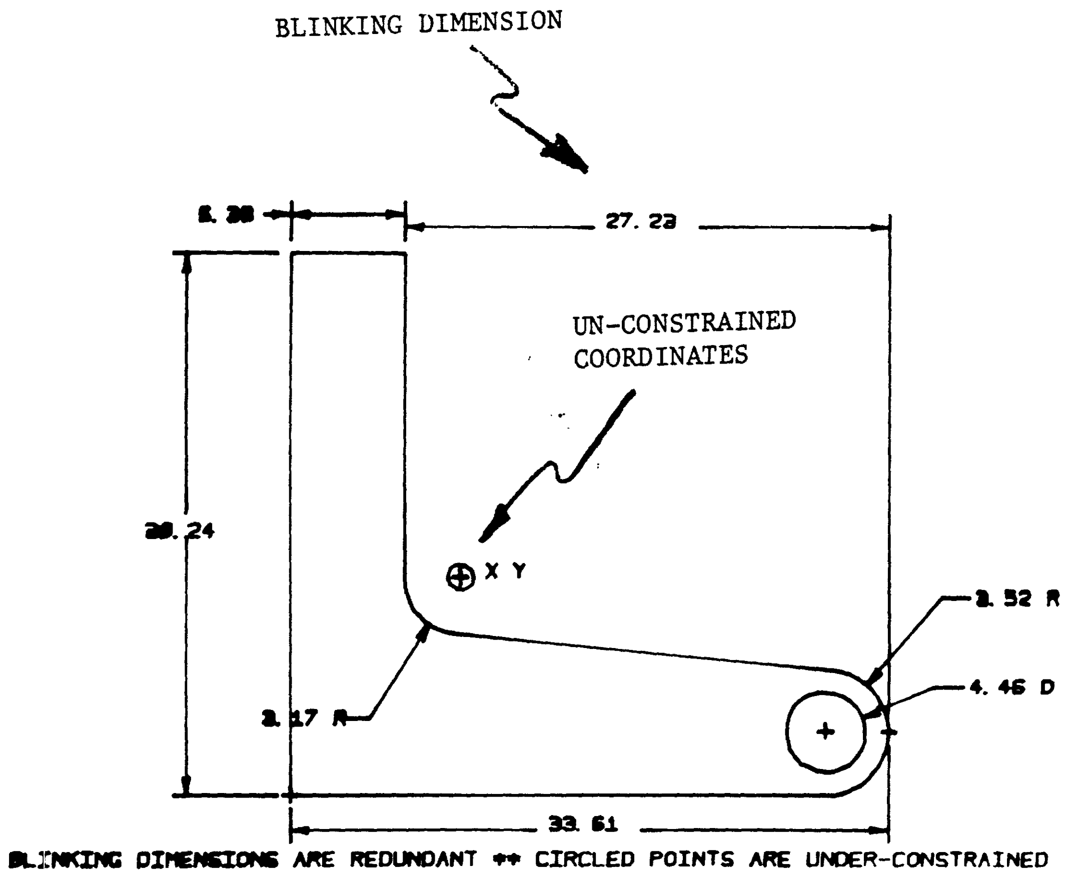


Figure D.40, Notification of redundancies and under-constrained points.

The NEW command:

When the operator digitizes the NEW menu area, Figure D.41 is displayed. The choice can then be made between modifying the current dimensioning scheme (EDIT), reading in a previously stored data set (OLD) or sketching in a new shape (DRFT). See section I for a detailed description of each of these commands.

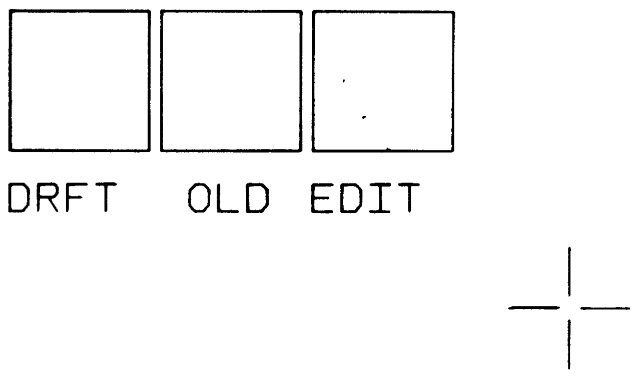
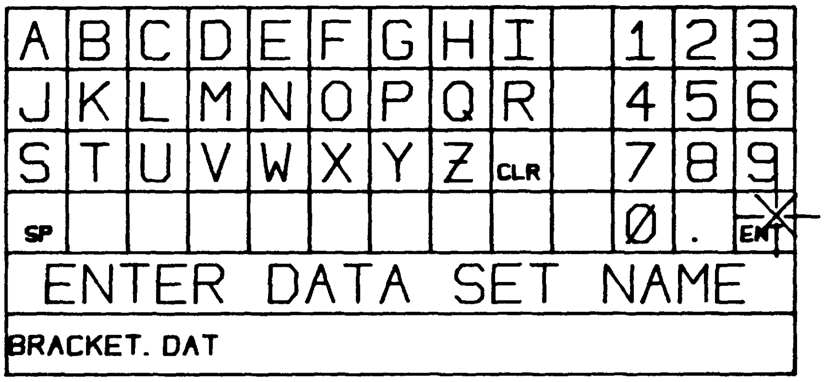


Figure D.41, Menu selection.

The SAVE command:

This allows the user to store the current data set on the disk. When the SAVE menu button is digitized, the user is prompted to enter the file name that the data set is to be stored under. The name is entered using the graphic keypad and the digitizing pen as shown in Figure D.42.



ENTER DATA SET NAME

BRACKET. DAT

Figure D.42, Specifying a file name for a SAVE command.

The ZOOM command:

The picture may be enlarged by digitzing the ZOOM menu button and then digitizing the opposite corners of a "window". If the digitizes shown in Figure D.43 are made, the area in the dotted box will be enlarged to fill the screen.

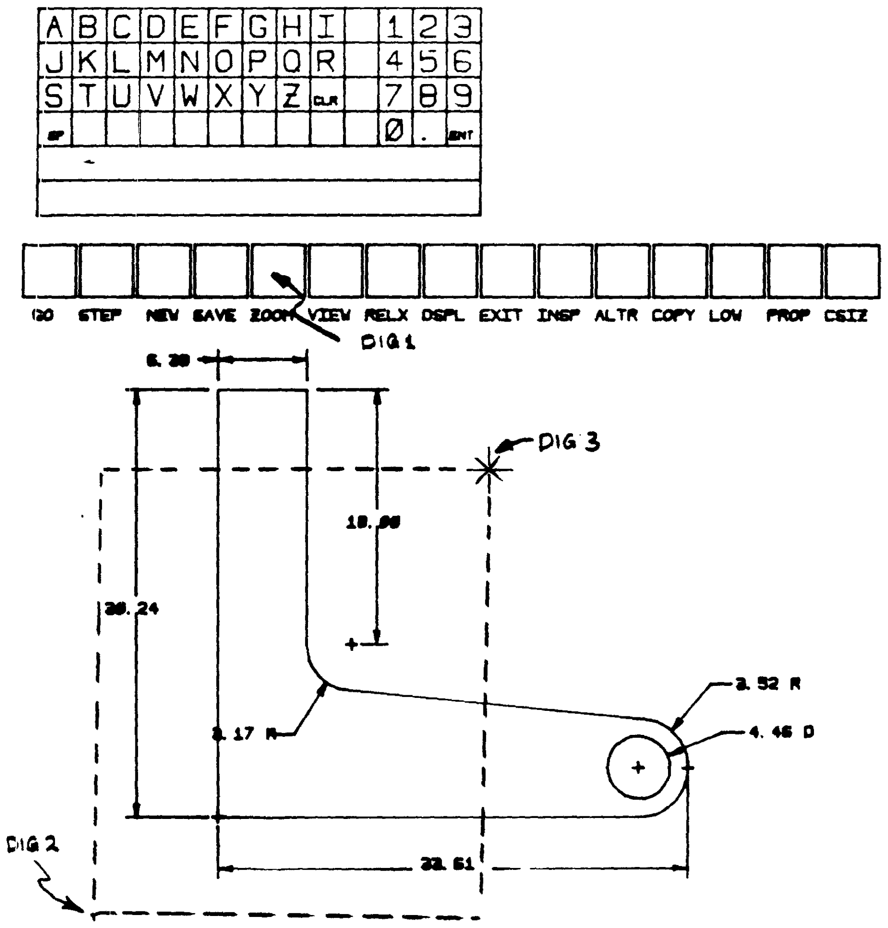


Figure D.43, Performing a ZOOM.

The VIEW command:

If this command is used, the scale of the drawing is changed so that all coordinates and dimension values are displayed. Since the positioning of the dimensions is related to the drawing scale, it is possible, that after the first VIEW command, not all the dimensions values will be visable. In this case, digitize the VIEW command again and this problem will be remedied.

The DSPL command:

This command simply "displays" the picture at the current scale with the current values of the dimensions. The geometry is not updated by using the DSPL command. This command is useful if, after many dimensional changes, the user wishes to see the potential dimensional values without using "GO".

The EXIT command:

This is used to exit the program. If by chance the user forgot to save the current part, the prompt "DO YOU WANT TO SAVE THE PART? (Y/N)" is displayed. If the user answers "YES", control is returned to the program and the user may execute any command. If "NO" is entered, the program terminates.

The INSP command:

This command is used to "inspect" the current value of any dimension. The operator simply digitizes the menu button and then digitizes the position of the number to be inspected. The value of the dimension is then displayed in the graphic keypad area to five decimal places.

The ALTR command:

This command is used to alter the placement of the dimensions. The dimensions are placed by starting at a certain location and searching in a certain direction until the dimension can be drawn such that no two dimensional values occupy the same place on the screen. For most dimension types, both the starting location and the search direction can be specified with the ALTR command.

The initial digitize is in the ALTR menu button, the next digitize is at the location of the value of the desired dimension. The next action depends on the type of the dimension selected.

1) For the PDIS constraint, the first digitize is the starting location and the second digitize specifies the search direction from the first digitize.

2) For LDIS constraints, no furthur digitizing is needed. The search simply starts from the opposite side of the line between the two constrained points.

3) For the HDIS and VDIS constraints, the operator is first asked which direction the search is to be made. This is via a graphic keypad entry. Then, the user must digitize the starting location of the search.

4) Angular dimensions may not be altered

5) For dimensions on the radius of an arc or the diameter of a circle, the only digitize needed will indicate the direction of the leader line from the center of the arc or circle.

The COPY command:

This command is used to make a hardcopy of the entire MEGATEK screen. The program creates a plot file named VARIAN.PLT in the user's directory on the VAX. This plot file can be plotted on any plotter at the JCF after exiting the program.

The PROP command:

This is used to compute and display the following information about the two dimensional part shape:

Perimeter;	Exterior perimeter of part.
Area;	Surface area of part.
Mx;	Moment of inertia about x-axis.
My;	Moment of inertia about y-axis.
Ix;	Second moment about x-axis.
Iy;	Second moment about y-axis.
Ixy;	Cross product of inertia.
X centroid;	X coordinate of center of mass.
Y centroid;	Y coordinate of center of mass.

The origin of the coordinate system is at the first point defined in the sketch input routine and is indicated by a plus sign.

The CSIZ command:

This command is used to alter the current character size. The operator is prompted to enter a number from 1 to 7. The value 1 results in the smallest character and the value 7 results in the largest character. The graphic key-pad and digitizing pen are used for this data entry.