Algorithm for drawing ellipses or hyperbolae with a digital plotter

By M. L. V. Pitteway*

An efficient algorithm is presented for drawing or displaying conic section curve segments, each incremental move being chosen by the computer to minimize the displacement from the intended curve. The inner cycle consists of three additions and one test for each move. Two further additions are required for display devices which do not accept incremental commands, and two further tests are required to detect possible changes of sector.

1. Introduction

In a recent paper (Bresenham, 1965) an algorithm is established for computer control of a digital plotter drawing straight lines. The object is to choose the combination of pen movements to produce the best possible approximation to a straight line between two points as shown in **Fig. 1,** without using more central processor time than necessary. For any specified straight line, only two of the eight possible pen movements will be required; the problem divides naturally into eight separate segments or "octants" of move combinations according to the direction of the line, and the treatment summarized below for the first octant can be extended to the other seven by including an initial test on the line gradient.

A flow chart for Bresenham's straight line algorithm is given in **Fig. 2** using the notation of this paper. Its validity is established as follows: After completing n moves, the pen will have reached the point n, j_n, where $0 \leqslant j_n \leqslant n$ and the unit of length is taken as the pen move 1. The next move is chosen as shown in **Fig. 3** to minimize the distance between the pen stopping place and the intended straight line. The smaller distance is

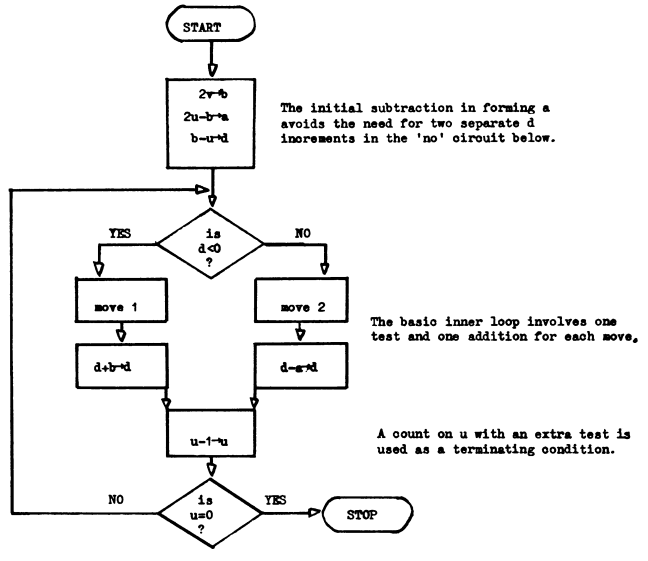

Fig. 2.—Flow chart of Bresenham's algorithm for choosing the moves for drawing a straight line in the first octant. The statement "$2v \rightarrow b$" has the same meaning as "$b := 2 \times v$" in ALGOL

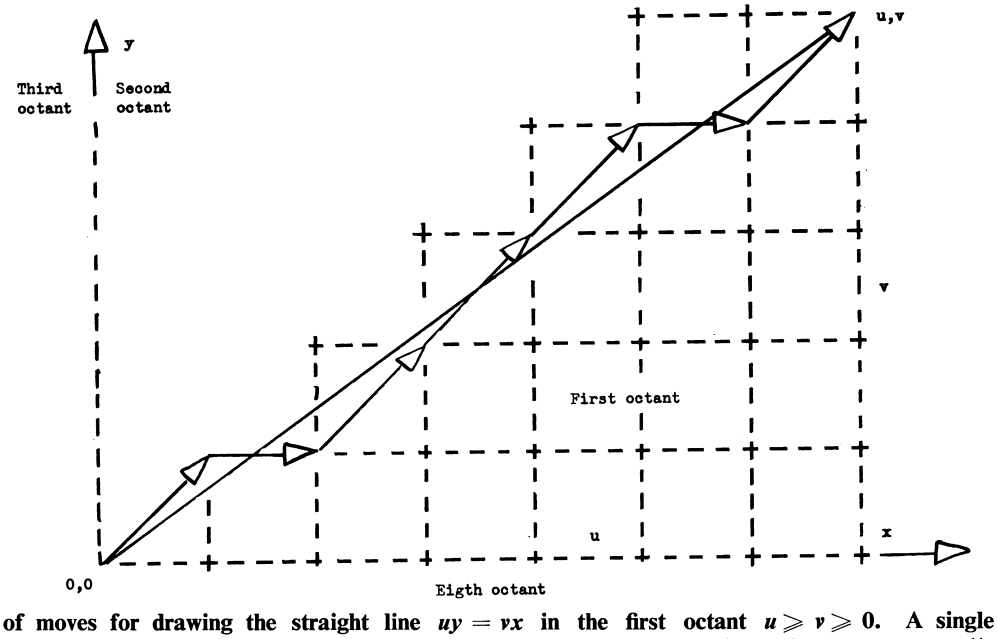

Fig. 1.—Choice of moves for drawing the straight line $uy = vx$ in the first octant $u \geqslant v \geqslant 0$. A single horizontal pen movement is taken to be the unit of length, and is referred to as "move 1". The diagonal pen movement "move 2" is automatically produced by a combination of horizontal and vertical movements

* *Computer Science Department, Brunel University, Woodlands Avenue, Acton, London W.*3.

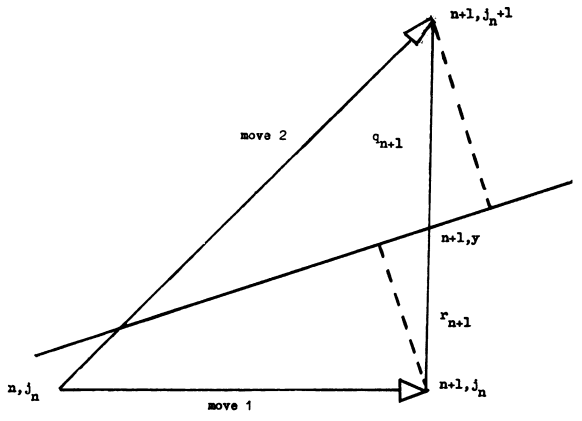

Fig. 3.—The best fit is determined by choosing the smaller of r_{n+1} and q_{n+1}. For a straight line, y at $x = n + 1$ is equal to vx/u

associated with the smaller of r_{n+1} and q_{n+1} (similar triangles), and the difference between these two quantities defines d for the test in Fig. 2:

$$\begin{aligned} d_{n+1} &= u(r_{n+1} - q_{n+1}) \\ &= 2v(n + 1) - u(2j_n + 1). \end{aligned} \tag{1}$$

u is positive, so cannot affect the sign of d. Equation (1) gives rise to the recurrence relations:

$$d_{n+1} = d_n + 2v - 2u(j_n - j_{n-1}) \quad \text{for } n \geqslant 1$$

and

$$d_1 = 2v - u. \tag{2}$$

The previous move gives $j_n - j_{n-1} = 0$ if $d_n < 0$, and $j_n - j_{n-1} = 1$ if $d_n \geqslant 0$, i.e. d is increased by $2v$ after move 1, and is decreased by $(2u - 2v)$ after move 2.

Before using the algorithm to draw a particular straight line, it is first necessary to move the ends of the line to the nearest grid point at which the pen can start or stop, as in Fig. 1. This slight displacement of the intended line can be avoided if a small residual displacement is introduced into the initial value of d. This correction term must also be included if the algorithm is to be resumed after an interruption. The previous argument can be repeated if the line is still supposed to pass through the point 0, 0 so that the pen mesh points are displaced to start from $0, \epsilon$; d_{n+1} becomes $2v(n + 1) - u(2j_n + 1 + 2\epsilon)$, where it is reasonable to suppose that $-1/2 \leqslant \epsilon < 1/2$. Alternatively, the line itself can be displaced from a 0, 0 mesh point origin by writing $y = y' - \epsilon$, so that its equation becomes $uy' - vx = u\epsilon = k/2$ (the constant $k/2$ is chosen to match the quadratic form (3) in the next section). The initial value of d is set up by writing $b - u + k \rightarrow d$ as the third statement after "start" in the Fig. 2 algorithm; no other part is affected by the change.

While the preceding arguments have been directed towards digital control of a digital plotter, a similar technique could be used for cathode ray tube display devices in which points are selected for brightening by a computer command specifying x and y coordinates. "Move 1" requires the adjacent point in the same row or column to be next selected, and "move 2" requires the move of a king piece in the game of draughts or checkers. Most devices of this type require absolute values for x and y instead of the incremental commands of a digital plotter, so an extra addition is required in the "no" circuit of Fig. 2 to keep track of the y coordinate, while the terminating count can be adjusted for x. Vector displays generate straight lines automatically, and similarly analogue techniques can be used to maintain curved displays (Dertouzos and Graham, 1966) except that it proves convenient to use mathematically more complex shapes than the quadratic forms of this paper. It is, however, possible that a special hardware implementation of part of the conic section algorithm could be useful in future applications; only additions and simple tests are involved in the inner loop, and it should be possible to drive these at the speeds required to set up a reasonable display.

2. Introduction to the quadratic form algorithm

The general equation for a conic section is given by the form:

$$\alpha y^2 + \beta x^2 + 2\gamma xy + 2uy - 2vx = k. \tag{3}$$

The minus sign is introduced with v to match the straight line notation of Section 1 in the special case $\alpha = \beta = \gamma = 0$, and the twos with γ, u and v to simplify the derivative equation (4). For simplicity k is taken to be zero in the main discussion, so that the intended curve passes through the starting point 0, 0; the effect of this residual displacement term is discussed at the end of Section 3. It is also assumed that the curve is initially directed in the first octant so that $0 \leqslant v \leqslant u$: the sign of the equation can be changed throughout if necessary, though strictly u and v both negative implies starting in the opposite direction, i.e. in the fifth or sixth octants.

Differentiating (3) with respect to x gives equation (4):

$$\frac{dy}{dx} = \frac{v - \beta x - \gamma y}{u + \gamma x + \alpha y}. \tag{4}$$

This form suggests that the constant b in the straight line algorithm should be replaced by the varying quantity $2(v - \beta x - \gamma y)$, and the constant a by $2(u + \gamma x + \alpha y) - b$; this is how the conic section algorithm operates. After each move x increases by 1, so b is decreased by 2β and a is increased by $(2\gamma + 2\beta)$. After move 2, y also increases by 1, so b is decreased by a further 2γ, and a is increased by a further $(2\alpha + 2\gamma)$. A change of octant is required when a or b become negative, and the starting values for a and b are carefully chosen to stop errors accumulating (Section 3).

The basic algorithm suggested by this procedure is shown in **Fig. 4,** where at the start of the run the constant k_1 should be set equal to 2β, k_2 and k_4 to $2\beta + 2\gamma$, and k_3 to $2\alpha + 2\beta + 4\gamma$. This is also made plausible by considering the limit if the step size of the plotter is supposed to become infinitely small. The "best fit" quality of the algorithm is not established until Section 3, but the following argument does show that Fig. 4 can be used as an alternative to the algebraic definition (3):

In making $N + M$ very small moves, the "yes" circuit in Fig. 4 may be passed N times, and the "no" circuit M times. The resulting move in the x direction $dx = N + M$, and $dy = M$, so $dy/dx = M/(N + M)$. Similarly, $da = k_2N + k_3M$ and $db = -k_1N - k_4M$; these can be integrated to give $a = k_2x + (k_3 - k_2)y + a_0$ and $b = -k_1x - (k_4 - k_1)y + b_0$, where a_0 and b_0 are constants. If it is assumed that a and b do not change appreciably during the move dx, the construction of the changes to d in Fig. 4 implies (by comparison with the straight line algorithm, for example) that $dy/dx = b/(a+b)$, which gives the differential equation:

$$\frac{dy}{dx} = \frac{b_0 - k_1x - (k_4 - k_1)y}{a_0 + b_0 + (k_2 - k_1)x + (k_1 - k_2 + k_3 - k_4)y}. \quad (5)$$

Equations (5) and (4) can be matched by choosing $a_0 + b_0 = 2u$ (this being just a convenient normalizing condition to match the straight line algorithm), $b_0 = 2v$, $k_1 = 2\beta$, $k_2 = k_4 = 2\gamma + 2\beta$ and $k_3 = 2\alpha + 2\beta + 4\gamma$ as expected. The full algorithm developed in the next section includes correction factors in the initial value of b, which is in fact $2v - \beta - \gamma$, but this has no affect in the very small step limit.

The algorithm of Fig. 4 can be generalized to include the case when k_2 is not equal to k_4. An extra degree of freedom is introduced into equation (3), but a detailed study of this generalized algorithm is beyond the scope of this paper.

3. The first octant

The "best" incremental approximation to the intended curve is defined in this paper by choosing each pen movement to give the smaller distance, measured in a direction parallel to the y axis, between the pen stopping point and the intended curve, i.e. the smaller of r_{n+1} and q_{n+1} again. The stopping point is not necessarily the closest to the curve if the gradient of the intended line changes appreciably during one increment, but it still seems a reasonable and convenient working choice.

Special case $\alpha = 0$

In order to avoid unnecessary complication at this stage, the special case when $\alpha = 0$ is considered first, so that equation (3) can be solved for y without introducing a square root:

$$y = \frac{x(v - \beta x/2)}{u + \gamma x}. \quad (6)$$

Divisions are avoided by introducing the multiplying factor $(u + \gamma x)$ into the definition of d replacing equation (1). This would cause the algorithm to fail if it was possible for $(u + \gamma x)$ to change sign, but when $\alpha = 0$ this implies an infinite gradient in equation (4), and this is not possible in the first octant considered here:

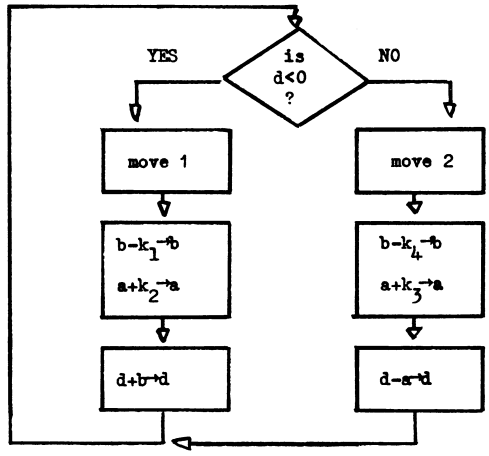

Fig. 4.—Algorithmic definition of a conic section (when $k_2 = k_4$) corresponding to equation 3. Each move involves just one test and three additions, so demands upon the central processor are modest

$$d_{n+1} = (r_{n+1} - q_{n+1})(u + \gamma n + \gamma)$$
$$= (2v - \beta n - \beta)(n + 1) - (u + \gamma n + \gamma)(2j_n + 1). \quad (7)$$

This leads to the recurrence relations:

$$d_{n+1} = d_n + 2v - \beta(2n+1) - \gamma(2j_n+1) - 2(u+\gamma n)(j_n - j_{n-1})$$
and
$$d_1 = 2v - u - \beta - \gamma. \quad (8)$$

Comparison with the straight line algorithm shows the form for b and a:

$$b = 2v - \beta(2n + 1) - \gamma(2j_n + 1)$$
and
$$a = 2(u + \gamma n) - b. \quad (9)$$

It is convenient to set up the algorithm as shown in Fig. 4 so that b and a are incremented immediately before making the change to d, so that the initial value of $b = 2v - \beta - \gamma$ and $a = 2u - b$. The full algorithm incorporating these terms is flow charted in **Fig. 5** (which also includes a term $\alpha/4$ and a residual displacement term k established below). The curvature terms k_1, k_2 and k_3 retain the values of Section 2, and the algorithm reduces to the expected straight line form of Fig. 2 when $\alpha = \beta = \gamma = 0$.

General case $\alpha \neq 0$

When $\alpha \neq 0$, equation (6) is replaced:

$$y = \frac{-u - \gamma x + \sqrt{[(u + \gamma x)^2 + \alpha(2vx - \beta x^2)]}}{\alpha}. \quad (10)$$

The positive square root is required to match (6) in the limit $\alpha \to 0$, and $y = 0$ at $x = 0$. The square root is inconvenient, and must be eliminated before a suitable

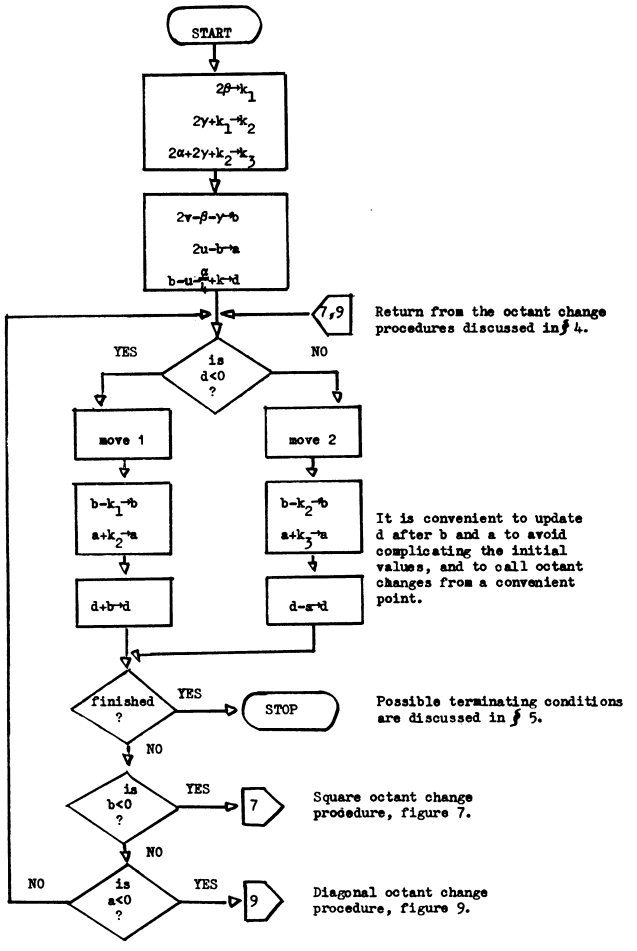

Fig. 5.—The complete conic section algorithm. Two additional tests are introduced to detect changes of octant, and some provision must be made in any particular implementation to terminate when the required arc is completed

generalization of (7) can be established. This can be done by restricting the investigation to the critical condition $d = 0$, which requires $r_{n+1} = q_{n+1}$ in Fig. 3, i.e. $2y - 2j_n - 1 = 0$. The square root part of y can then be taken over to the right hand side, both sides squared, and multiplication throughout by α gives the critical condition:

$$(2v - \beta x)x - (u + \gamma x)(2j_n + 1) - \alpha(j_n + 1/2)^2 = 0. \quad (11)$$

Comparison with the right hand side of equation (7) (replacing x by $n + 1$) suggests an additional term, $-\alpha(j_n + 1/2)^2$, in the definition of d_{n+1}. Effectively, the multiplying factor $(u + \gamma x)$ introduced into (7) is generalized and replaced by (12):

$$\tfrac{1}{2}\{\sqrt{[(u + \gamma x)^2 + \alpha(2vx - \beta x^2)]} + (u + \gamma x) + \alpha(j_n + 1/2)\}. \quad (12)$$

This multiplying factor is formed by changing the sign of the square root in $(r_{n+1} - q_{n+1})$ to establish a form corresponding to the difference of two squares, with a multiplying factor $\alpha/2$ introduced to match (7) in the limit $\alpha \to 0$. Again, the algorithm will fail if (12) changes sign, but this is also unlikely in the first octant unless the curve turns very sharp, a point discussed in Section 4.

The left hand side of (11) is the residue of the quadratic form (3) evaluated at the critical point $x, j + \frac{1}{2}$, and when taken as a modified definition of d_{n+1} applicable for any α, leads to the recurrence relations:

$$d_{n+1} = d_n + 2v - \beta(2n + 1) - \gamma(2j_n + 1) - (j_n - j_{n-1})(2u + 2\gamma n + \alpha[1 + j_n + j_{n-1}])$$

and
$$d_1 = 2v - u - \beta - \gamma - \alpha/4. \quad (13)$$

The algorithm requires b and a:

$$b = 2v - \beta(2n + 1) - \gamma(2j_n + 1)$$

and
$$a = 2(u + \gamma n + \alpha j_n) - b. \quad (14)$$

These formulae are implemented in Fig. 5. The division by 4 (a simple right shift two places on most binary computers) can be avoided if α is replaced by 2α, and logically β by 2β as well so that the twos can be dropped in the basic form (3); alternatively, the other initial values in the algorithm can be scaled up.

Residual displacement term

As in the cast of the straight line, the intended arc may not start from an exact mesh point. It is convenient to displace the curve rather than the mesh points in generalizing the analysis, so that the curve takes the form (3) with $k \neq 0$, though k should be small enough to make the starting point the same point that would have been reached by the algorithm in drawing the previous part of the curve. Following the previous analysis, the modified equations yield

$$d_{n+1} = (2v - \beta x)x - (u + \gamma x)(2j_n + 1) - \alpha(j_n + 1/2)^2 + k,$$

so the addition of k to the initial value of d is the only change required.

4. Octant changes

With the introduction of curvature it is possible for dy/dx to become greater than one, or to become negative, so that a change of octant may be required. A satisfactory algorithm must include provision to follow several changes of octant so that a complete ellipse, for example, can be drawn with the pen closing exactly to its starting point. The two possible types of octant change are considered separately. The simpler "square" change occurs if the gradient becomes negative; this calls for a change in the specification of move 2 and a reversal of the y direction. The "diagonal" octant change occurs when the gradient becomes greater than one, so that move 1 is changed. This involves an interchange of x and y, so that in the second octant the "best fit" is determined by measurement parallel to the x axis instead of y. In establishing the octant change procedures, only

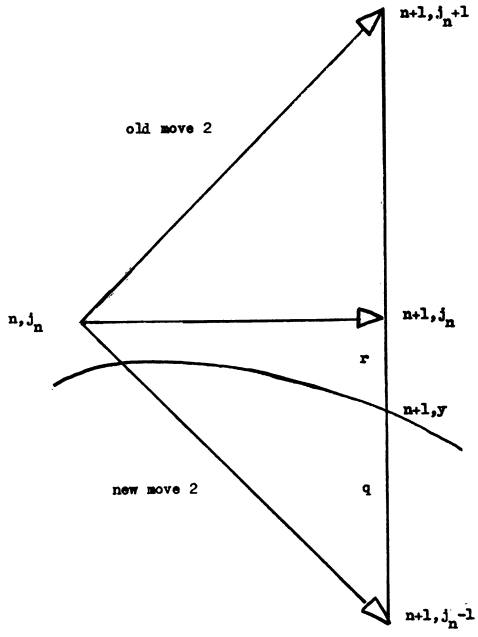

Fig. 6.—A square octant change occurs when *b* becomes negative, so that *dy/dx* becomes negative. The specification of move 2 is changed

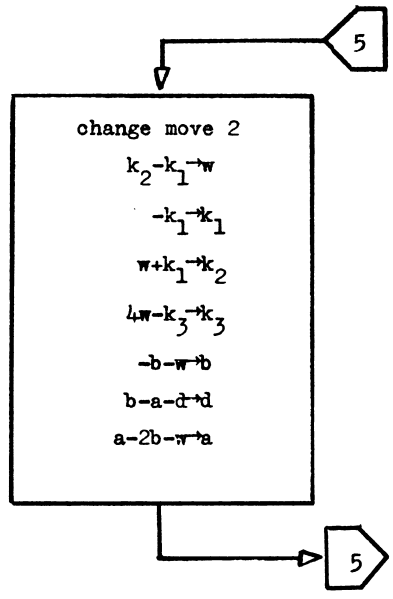

Fig. 7.—Algorithmic representation of a square octant change. Only local values are required, so the transformation can be repeated without further modification to the algorithm

the current values of k_1, k_2, k_3, b, a and d are involved; there is no recourse to the coefficients in the original equation, so a sequence of octant changes can be set up without further analysis. The new move 2 for a square octant change, or move 1 for a diagonal octant change, depends only upon the current octant.

Square octant change

A square octant change is required when b becomes negative, as shown in **Fig. 6.** The octant change is called from the Fig. 5 algorithm before the new (and in this case spurious) value of d is used to select a move. b is advanced before d in the algorithm, so the previous move from a valid d formed with a positive b has been completed. To avoid the confusion of an axis change, the previous analysis can be repeated, noting that $j_n = j_{n-1} - 1$ after making a new move 2. This gives the new recurrence relations, and the necessary starting values can be retrieved from the current values of k_1, k_2, k_3, b, a and d. The new required $d_{n+1} = (2j_n - 2y - 1)$ multiplied by a suitable factor, which in this case is obtained from (12) by changing the term $\alpha(j_n + \frac{1}{2})$ into $\alpha(j_n - \frac{1}{2})$.

The new $d_{n+1} = -(2v - \beta x)x + (u + \gamma x)(2j_n - 1) + \alpha(j_n - \frac{1}{2})^2$
$= -d - a - b - 2\gamma$ as already evaluated.

The new $b_{n+1} = -2v + \beta(2n + 1) + \gamma(2j_n - 1)$ and $a_{n+1} = 2u + 2\gamma n + 2\alpha j_n - b_{n+1}$, so the new $b = -b - 2\gamma$ as already evaluated, and the new $a = a + 2b + 2\gamma$ as already evaluated. The new $k_1 = -2\beta$, $k_2 = 2\gamma - 2\beta$ and $k_3 = 4\gamma - 2\alpha - 2\beta$, so that in effect α and β change sign.

The complete transformation can be achieved by the operations shown in **Fig. 7,** where for convenience w is introduced as workspace. The instruction order must be preserved, as for example the expression for d is formed using the new value of b and the old value of a. A double transformation returns to the original values for any k_1, k_2, k_3, b, a and d.

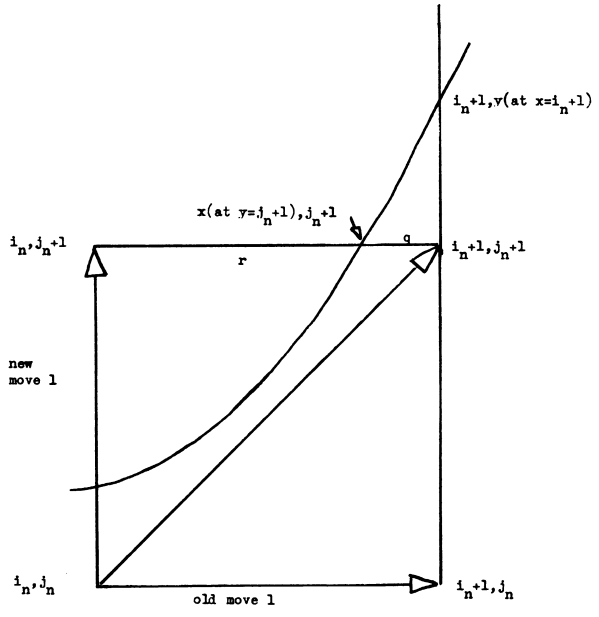

Fig. 8.—A diagonal octant change occurs when *a* becomes negative, so that *dy/dx* > 1. The specification of move 1 is changed, and the "best fit" now requires measurement parallel to the *x* axis

Diagonal octant change

The numeric value associated with the new move 1 after a diagonal octant change is move 2 minus the old

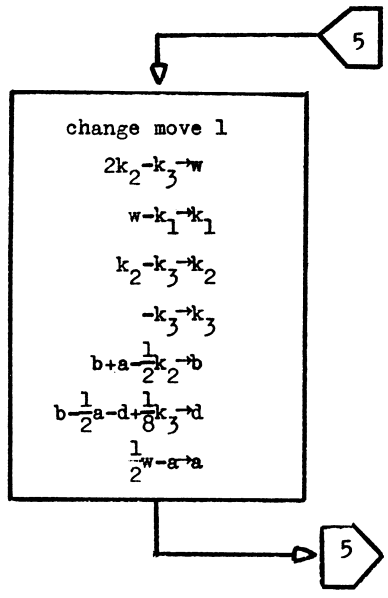

Fig. 9.—Algorithmic representation of a diagonal octant change. As in Fig. 7, only local values are required

move 1 for many computers, as the diagonal move is often specified by the addition of the numeric values of the two simultaneous square moves. After the change x is no longer incremented on every move, so it is necessary to replace n by i_n for x in the algebraic formulation; $i_n=n$, however, at the changeover point from the first octant. The geometry of the diagonal octant change is shown in **Fig. 8.** The notation requires care, because the spurious d already formed involves the value of y at $x = i_n + 1$, while the required d involves the value of x at $y = j_n + 1$.

In the new octant, $j_{n+1} = j_n + 1$, and i_n is increased only after a move 2. The required

$$d_{n+1} = [2x \text{ (at } y = j_n + 1) - 2i_n - 1]$$

multiplied by a suitable factor. Following the previous analysis with x and y interchanged, the required

$$d_{n+1} = (2u + \alpha y)y - (v - \gamma y)(2i_n + 1) + \beta(i_n + 1/2)^2$$

where $y = j_n + 1$ (not its meaning in Fig. 8). From this it can be deduced that

$$d_{n+1} = b + a/2 + 3\alpha/4 - \beta/4 + \gamma/2 - d$$

as already evaluated. From the new recurrence relations,

$$b_{n+1} = 2u + \alpha(2j_n + 1) + \gamma(2i_n + 1) = a + b + \alpha + \gamma$$

as already evaluated,

$$a_{n+1} = 2v - 2\beta i_n - 2\gamma j_n - b_{n+1} = -a - \alpha + \beta$$

as already evaluated,

$$k_1 = -2\alpha,\ k_2 = -2\alpha - 2\gamma \text{ and } k_3 = -2\alpha - 2\beta - 4\gamma.$$

The complete transformation can be achieved by the operations of **Fig. 9,** which again has the nil double transformation property. The division by powers of two can only be avoided by permitting the six quantities

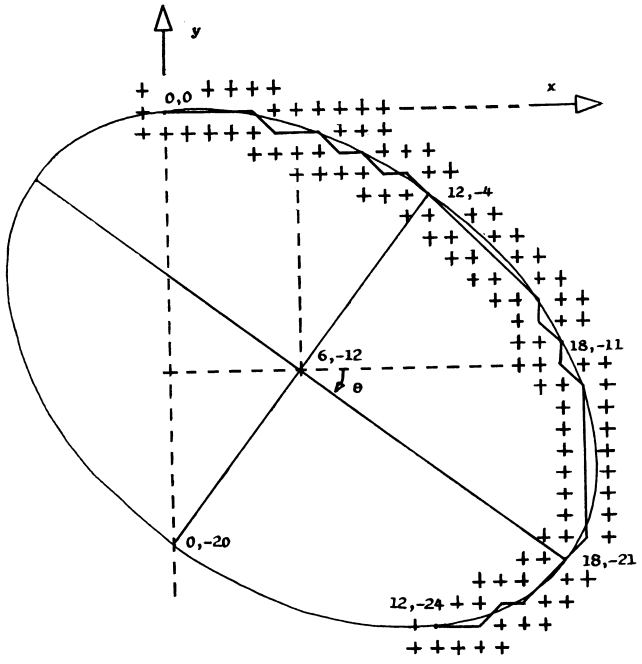

Fig. 10.—The algorithm's selection of moves to fit the ellipse $36y^2 + 29x^2 + 24xy + 720y - 60x = 0$. The minor axis length is 20, and the major axis has length 30 and is set at an angle θ to the x axis where $\sin\theta = 0{\cdot}6$. The ellipse passes through the mesh points indicated in the figure. The algorithm will complete the second half of the ellipse, restoring k_1, k_2, k_3, b, a and d to their original values

k_1, k_2, k_3, b, a and d to increase by 8 after every diagonal octant change, which is unsatisfactory. Right shifts are fast on many machines, and octant changes should occur far less often than regular moves, so it is not unreasonable to permit their use here.

An example

The complete algorithm with octant changes has been extensively checked by hand, and an Atlas Autocode program has been developed and run successfully on the KDF 9 in the Cripps Computing Centre, Nottingham University. **Fig. 10** shows a particularly simple ellipse:

$$36y^2 + 29x^2 + 24xy + 720y - 60x = 0. \quad (15)$$

This ellipse has a major axis length 30, a minor axis length 20, and if plotted symetrically about the origin ($x^2/9 + y^2/4 = 25$) passes through the four points $x = \pm 12$, $y = \pm 6$. The ellipse (15) passes through the origin with a gradient of 1/12, so it is possible to start in the first octant. Following the algorithm, the initial values of the six quantities involved are shown in **Table 1.** There is a change to the eighth octant after one move, to the seventh octant after another thirteen moves, to the sixth octant after another eleven moves (though this change could equally well be made after ten moves when it happens that $b = 0$), and to the fifth octant after another five moves. Then, after a further five moves, the pen reaches the mesh point 12, −24,

which is exactly half way round the ellipse; at this point, k_1, k_2, k_3, b, a and d all have the value they started with, so the process repeats exactly except that the moves occur in the fourth, third, second, and finally back to the first octant to close the ellipse and restore the original values again. The values of the six quantities after each octant change are shown in Table 1.

Table 1

	Initial	8	7	6	5
k_1	58	−58	72	−72	58
k_2	82	−34	48	−96	82
k_3	178	−82	82	−178	178
b	19	15	600	96	357
a	701	729	8	538	99
d	−350	−379	$484\frac{3}{4}$	$-472\frac{3}{4}$	151

Starting values in the other octants

In a general implementation, the algorithm must permit a start in any octant. This could be programmed through a series of consecutive octant changes at the start until an octant is found in which both b and a have positive initial values, but it is easy enough to write out all the possible starting values in full; these are presented in **Table 2**. The relevant octant can be selected by tests on u and v as follows:

If $|u| < |v|$, the initial octant is 2, 3, 6 or 7;

if $u < 0$, the initial octant is 3, 4, 5 or 6;

and if $v < 0$, the initial octant is 5, 6, 7 or 8.

In many applications, the coefficients of the quadratic form (3) will not be given outright. Instead, the curve may be required to pass through five given points, or the end points may be specified with starting and finishing gradients and some curvature parameter (Dertouzos and Graham, 1966). It may then prove more convenient to rewrite the first part of the algorithm to set up the initial values of k_1, k_2, k_3, b, a and d from the given parameters.

Degenerate case

The Fig. 5 algorithm assumes that the gradient changes reasonably slowly, so that only one octant change will be required at a time. It is interesting to study the behaviour of the algorithm in a degenerate case when the conic reduces to two straight lines:

$$(14y - 10x)(9y - 2x - 19) = 0. \qquad (16)$$

The two straight lines, both directed into the first octant, intersect in the region of the fifth pen movement. The intended line passing through the origin is $7y - 5x = 0$, and has been drawn out properly by the straight line algorithm in Fig. 1. The Fig. 5 algorithm follows the lower line after the intersection, as shown in **Fig. 11**. As the pen moves towards the intersection, b and a both decrease. After four moves, b becomes negative, and there is a square change into the eighth octant; b remains negative after the change, however, so the algorithm returns to the first octant after the next move, and this is repeated after seven moves.

In this particular case it can be argued that the algorithm has behaved reasonably in following a hyperbola formed by passing a plane very nearly through the apex of a cone with a wide angle, but the lines can also be formed from a narrow cone so that the intended "hyperbola" doubles back into the fourth octant towards the mesh point 0,2. In the degenerate case there is nothing to choose between the two interpretations, but a small change in the coefficients of equation (16) could prevent the factorization and make the intended line double back. A very small change in the coefficients will not affect the operation of the algorithm in selecting its discrete steps, so the drawn line will jump the gap to the other branch of the hyperbola; this behaviour must be expected when the gradient changes through several octants in one step. The algorithm can be modified to permit sharper turns (e.g. tests for b or a remaining negative at the end of an octant change) if precautions are taken to prevent looping, but a more detailed study is beyond the scope of this paper.

5. Termination

The terminating condition is not specified in Fig. 5, as this depends upon the particular application of the algorithm. For example, the computer commands for most digital displays require absolute values for the x and y coordinates of each point to be brightened; in this case, the algorithm must be extended to increment x (or y in octants 2, 3, 6 or 7) after move 1, and both x

Table 2

OCTANT	1	2	3	4	5	6	7	8
k_1	2β	-2α	2α	-2β	2β	-2α	2α	-2β
$k_2 - k_1$	2γ	-2γ	-2γ	2γ	2γ	-2γ	-2γ	2γ
$k_3 - k_2$	$2\alpha + 2\gamma$	$-2\beta - 2\gamma$	$2\beta - 2\gamma$	$-2\alpha + 2\gamma$	$2\alpha + 2\gamma$	$-2\beta - 2\gamma$	$2\beta - 2\gamma$	$-2\alpha + 2\gamma$
b	$2v - \beta - \gamma$	$2u + \alpha + \gamma$	$-2u - \alpha + \gamma$	$2v + \beta - \gamma$	$-2v - \beta - \gamma$	$-2u + \alpha + \gamma$	$2u - \alpha + \gamma$	$-2v + \beta - \gamma$
$a + b$	$2u$	$2v$	$2v$	$-2u$	$-2u$	$-2v$	$-2v$	$2u$
$d - b$	$-u - \frac{\alpha}{4} + k$	$-v + \frac{\beta}{4} - k$	$-v - \frac{\beta}{4} + k$	$u + \frac{\alpha}{4} - k$	$u - \frac{\alpha}{4} + k$	$v + \frac{\beta}{4} - k$	$v - \frac{\beta}{4} + k$	$-u + \frac{\alpha}{4} - k$

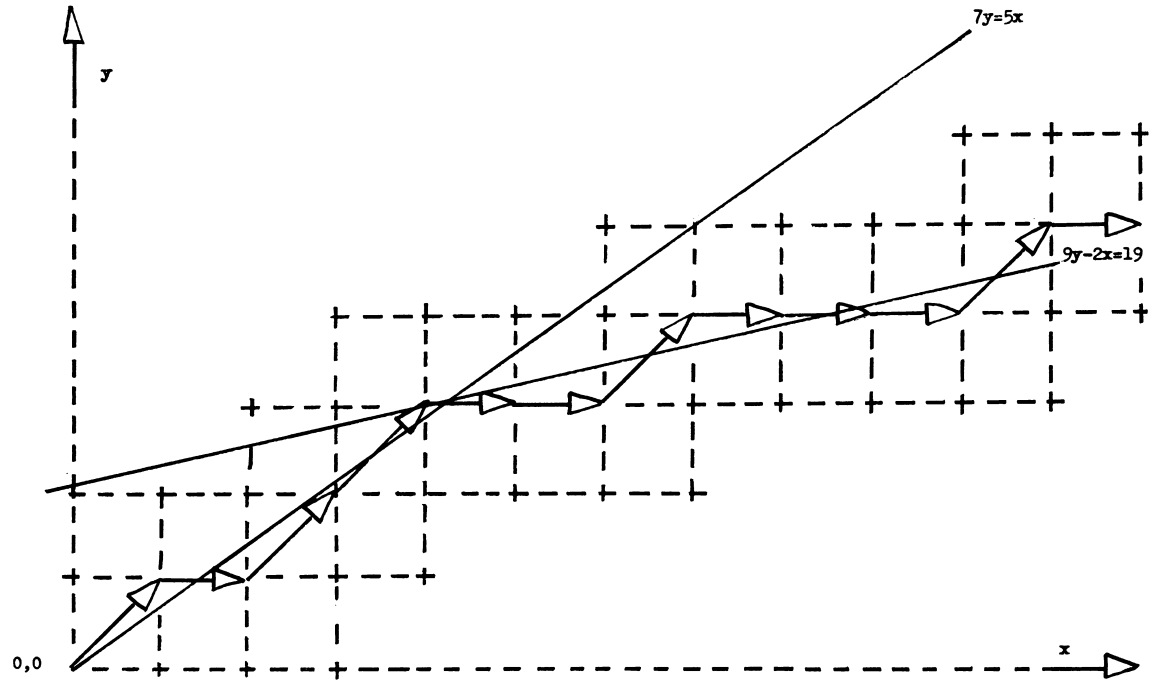

Fig. 11.—In this degenerate case, specifying two straight lines, the algorithm changes to the other line after the intersection

and y after move 2, and these values of x and y will be available for the terminating test.

In a purely incremental application, it may be possible to use a single count in the inner loop as in the straight line algorithm, if the terminating octant is known. If, for example, the final value of x is known to occur in octant 4, a test on x can be backed by an octant test to resolve the ambiguity. The final value must give x for octants 1, 4, 5 or 8, and y for octants 2, 3, 6 or 7.

As suggested in the Introduction, it might be worthwhile to implement the inner loop tests and additions in special purpose hardware. The octant change procedures are more complicated, so it may prove convenient to call the main computer when b or a becomes negative. This is reasonable for a high resolution display with a fine mesh, as the number of octant changes depends only upon the overall shape of the drawing. The main computer could also set up a terminating hardware count once the final octant has been reached. Some users may even want to stop at a zero or unit gradient, in which case the terminating count can be eliminated.

Acknowledgements

The author is indebted to the staff of the Cripps Computing Centre at Nottingham University, and in particular Dr. Ian Newman who prepared the Atlas Autocode program to test the algorithm presented in this paper, and Mr. Albert Nicholson for suggesting possible hardware implementations for digital displays.

References

BRESENHAM, J. E. (1965). Algorithm for computer control of a digital plotter, *IBM Systems Journal*, Vol. 4, p. 25.

DERTOUZOS, M. L., and GRAHAM, H. L. (1966). A parametric graphical display technique for on-line use, *AFIPS Conference Proceedings*, Vol. 29, p. 201.