Graphics and Image Processing

W. Newman*
Editor

A Linear Algorithm for Incremental Digital Display of Circular Arcs

Jack Bresenham
IBM System Communications Division

Circular arcs can be drawn on an incremental display device such as a cathode ray tube, digital plotter, or matrix printer using only sign testing and elementary addition and subtraction. This paper describes methodology for producing dot or step patterns closest to the true circle.

Key Words and Phrases: graphics, circle drawing, step generation, dot generation, incremental digital plotting, raster display, integer arithmetic, circle algorithm

CR Categories: 4.41, 8.2

Author's address: IBM System Communications Division, 1000 Westchester Avenue, White Plains, NY 10604.

* Editor when the paper was being processed; J. Foley is now editor of this department.

1. Introduction

This paper describes an algorithm for circular arc mesh point selection using incremental display devices such as a cathode ray tube or digital plotter. Error criteria are explicitly specified and both squared and radial error minimization considered. The repetitive incremental stepping loop for point selection requires only simple addition/subtraction and sign testing; neither quadratic nor trigonometric evaluations are required. When a circle's center point and radius are integers, only integer calculations are required.

The circle algorithm complements an earlier line algorithm described in [1, 2]. The algorithm's minimum error point selection is appropriate for use in numerical control, drafting, or photo mask preparation applications where closeness of fit is a necessity. Its simplicity and use only of elementary addition/subtraction allow its use in small computers, programmable terminals, or direct hardware implementations where compactness and speed are desirable.

The display devices under consideration are capable of executing, in response to an appropriate pulse, any one of the eight linear movements shown in Figure 1. Thus incremental movement is from a point on a mesh to any of its eight adjacent points on the mesh.

All generated data points must lie on mesh points and must consequently have integer display coordinates. It is assumed that by scaling and appropriate translation of axes, the circle is centered at the origin of a rectangular coordinate system whose units are those of the display device.

At each move, the algorithm choses a point so as to minimize the absolute difference between R^2 and the square of the radius to the point. In the Appendix it is shown that this also minimizes the linear difference between R and the radius itself when the circle is centered at a display mesh point and has an integer radius.

In this paper, the algorithm is developed for the case of clockwise movement from $(0, R)$ to $(R, 0)$ through the first quadrant. Requisite modifications for completing the full circular path are then indicated and the basic algorithm stated for tridirectional movement control by quadrant.

For analysis, the first quadrant arc of the circle given by

$$X^2 + Y^2 = R^2, \text{ where } X \triangleq \text{abscissa} \geq 0, \quad Y \triangleq \text{ordinate} \geq 0, \quad R \triangleq \text{an integer} \geq 1$$

will be used. The extension to complete the full circle will then be described as will the modifications required to accomodate an arbitrary arc of the general circle given by

$$(x - a)^2 + (y - b)^2 = r^2$$

with starting point (x_s, y_s) and terminating point (x_t, y_t) specified on circumference.

2. Analysis

In the first quadrant of a circle, y is a monotonically decreasing function of x. Clockwise movement in this quadrant can therefore be accomplished by a sequence of moves involving only m_1, m_2, and m_3.

When the display is at the point P_i, whose coordinates are (X_i, Y_i), the next movement is either m_1 to $(X_i + 1, Y)$ at 0°, m_2 to $(X_i + 1, Y_i - 1)$ at 315°, or m_3 to $(X_i, Y_i - 1)$ at 270°. The absolute difference between R^2 and the squares of the constrained radii of the three alternatives is minimized by determining the minimum of the following quantities:

$$| [(X_i + 1)^2 + Y_i^2] - R^2 |,$$
for m_1 movement to $(X_i + 1, Y_i)$,
$$| [(X_i + 1)^2 + (Y_i - 1)^2] - R^2 |,$$
for m_2 movement to $(X_i + 1, Y_i - 1)$,
$$| [X_i^2 + (Y_i - 1)^2] - R^2 |,$$
for m_3 movement to $(X_i, Y_i - 1)$.

The algorithm simplifies this threefold evaluation to consideration of only two points at each step by first observing the sign of the difference Δ_i:

$$\Delta_i = \{[(X_i + 1)^2 + (Y_i - 1)^2] - R^2\},$$

the difference between R^2 and the square of the radius to the diagonally adjacent point $(X_i + 1, Y_i - 1)$.

Figure 2 illustrates the five possibilities for the circle's intersection with the coordinate lines $X_i + 1$ and $Y_i - 1$ which must be considered when selecting the step for movement from point $P(X_i, Y_i)$. For clarity, subscripts for X_i and Y_i are dropped in developing the alternatives.

(a) If $\Delta_i < 0$, then $(X + 1, Y - 1)$ is in the interior of the true circle, i.e. 1 or 2 in Figure 2. The true circle passes between the points $(X + 1, Y)$ and $(X + 1, Y - 1)$, case 1, or between the points $(X + 1, Y + 1)$ and $(X + 1, Y)$, case 2.

In case 1 the closer of the two points can be found by observing the sign of the difference δ:

$$\delta = | [(X + 1)^2 + Y^2] - R^2 | - | [(X + 1)^2 + (Y - 1)^2] - \mathrm{R}^2 |.$$

Since in case 1 the constrained radius to $(X + 1, Y)$ exceeds or equals R while the constrained radius to $(X + 1, Y - 1)$ is less than R,

$$\{[(X + 1)^2 + Y^2] - R^2\} \geq 0 \text{ and}$$
$$\{[(X + 1)^2 + (Y - 1)^2] - R^2\} < 0.$$

Rewriting the definition of δ removing the absolute value expressions thus gives

$$\delta = \{[(X + 1)^2 + Y^2] - R^2\} + \{[(X + 1)^2 + (Y - 1)^2] - R^2\}.$$

Adding and subtracting $2Y - 1$ and collecting terms of $(X + 1)^2$ and $(Y - 1)^2$ gives

$$\delta = 2\{[(X + 1)^2 + (Y - 1)^2] - R^2\} + 2Y - 1.$$

Recalling the definition of Δ_i and substituting it into the previous equation yields

$$\delta = 2\Delta_i + 2Y_i - 1,$$

where $\delta \leq 0$ implies move m_1 and $\delta > 0$ implies move m_2. In case 2 the movement should be m_1. That in case 2, δ is always less than zero and hence forces the required m_1 move can be demonstrated as follows:

$$\delta = 2\Delta_i + 2Y_i - 1,$$
$$\delta = \{[(X + 1)^2 + Y^2] - R^2\} + \{[(X + 1)^2 + (Y - 1)^2] - R^2\}.$$

In case 2 the constrained radii to both $(X + 1, Y)$ and $(X + 1, Y - 1)$ are less than R:

$$\{[(X + 1)^2 + Y^2] - R^2\} < 0 \text{ and}$$
$$\{[(X + 1)^2 + (Y - 1)^2] - R^2\} < 0$$

so that here

$$\delta = -| [(X + 1)^2 + Y^2] - R^2 | - | [(X + 1)^2 + (Y - 1)^2] - R^2 |.$$

Thus $\delta < 0$ in all occurrences of case 2, so m_1 is correctly made.

(b) If $\Delta_i > 0$ then $(X + 1, Y - 1)$ is exterior to the true circle, i.e. cases 3 and 4 in Figure 2. The true circle passes between the points $(X + 1, Y - 1)$ and $(X, Y - 1)$, case 3, or between the points $(X, Y - 1)$ and $(X - 1, Y - 1)$, case 4.

In like manner, an alternate difference δ':

$$\delta' = 2\Delta_i - 2X_i - 1$$

yields a similar selection criterion; where $\delta' \leq 0$ implies move m_2 and $\delta' > 0$ implies move m_3.

(c) If $\Delta_i = 0$ then $(X + 1, Y - 1)$ is on the true circle, i.e. case 5, and the movement should be m_2. In

Fig. 1.

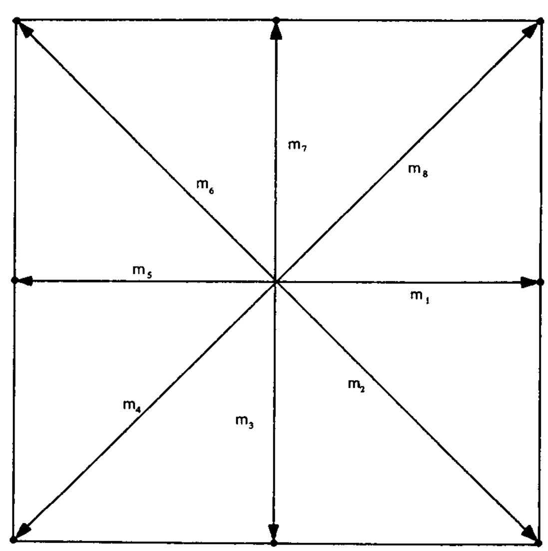
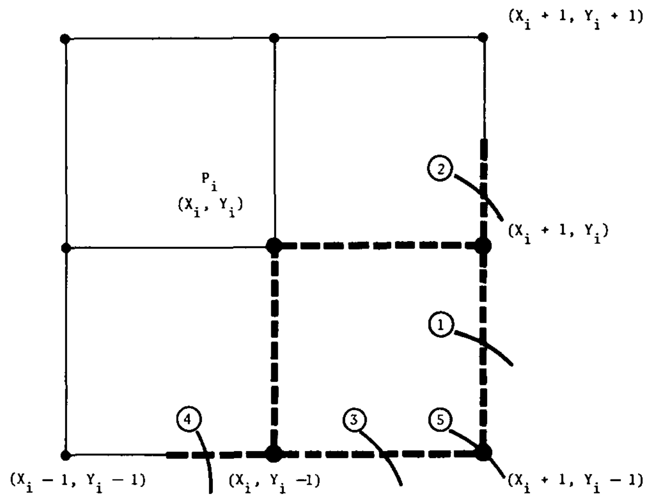


Fig. 2.



this case the above steps yield $\delta > 0$ and $\delta' < 0$ so a proper m_2 move is forced by either calculation.

By noting the expansions

$$(X+1)^2 = X^2 + 2X + 1, \quad (Y-1)^2 = Y^2 - 2Y + 1,$$

the difference Δ_i and the coordinates of the display point P_i are observed to have the following recurrence relations:

(i) For m_1 movement (i.e. $\Delta_i < 0$ and $\delta \leq 0$),

$$X_{i+1} = X_i + 1,$$
$$Y_{i+1} = Y_i,$$
$$\Delta_{i+1} = \Delta_i + 2X_{i+1} + 1.$$

(ii) For m_2 movement (i.e. $\Delta_i \leq 0$ and $\delta > 0$, or $\Delta_i \geq 0$ and $\delta' \leq 0$),

$$X_{i+1} = X_i + 1,$$
$$Y_{i+1} = Y_i - 1,$$
$$\Delta_{i+1} = \Delta_i + 2X_{i+1} - 2Y_{i+1} + 2.$$

(ii) For m_3 movement (i.e. $\Delta_i > 0$ and $\delta' > 0$),

$$X_{i+1} = X_i,$$
$$Y_{i+1} = Y_i - 1,$$
$$\Delta_{i+1} = \Delta_i - 2Y_{i+1} + 1.$$

If the circle is started at one of the four intersections of the true circle with the coordinate axes, there is no need to consider any second-order terms in the initial conditions. If the circle starts at the point ($X_0 = 0$, $Y_0 = R$), then

$$\Delta_0 = [(X_0 + 1)^2 + (Y_0 - 1)^2] - R^2 = 2 - 2R$$

and the full first-quadrant clockwise quarter arc will be complete when $Y_i = 0$ or, alternatively, when $Y_i < \frac{1}{2}$.

To complete the remaining three quarter arcs, the movement values m_1, m_2, and m_3 are respecified appropriately; the algorithm is reinitialized and is repeated with the same logic as before until the complete circle is drawn. When crossing a quadrant boundary the algorithm is reinitialized with

$$X_0 \leftarrow 0, \ Y_0 \leftarrow R, \ \Delta_0 \leftarrow 2 - 2R$$

or, alternatively, only current variables can be employed by reinitializing with

$$X_0 \leftarrow -Y_i, \ Y_0 \leftarrow X_i, \ \Delta_0 \leftarrow \Delta_i - 4X_i.$$

As is true for some plotters, let $M1$, $M2$, $M3$ denote, respectively, the appropriate m subscript for actual movement codes within a quadrant corresponding to normalized m_1, m_2, and m_3 first quadrant movement. With initial conditions of $M1_0 = 1$, $M2_0 = 2$, $M3_0 = 3$, one can observe that at each quadrant crossing the clockwise movement code reinitialization is

$$M1_{j+1} \leftarrow M3_j, \ M2_{j+1} \leftarrow M1_{j+1} + 1,$$
$$M3_{j+1} \leftarrow 8 \mid (M1_{j+1} + 2),$$

where $a \mid b$ is b modulo a.

Alternatively, movement can be coded as a 2-tuple of delta abscissa and ordinate (Δx, Δy) increments 1, 0, or -1. With initial conditions of $M1_0 = (1, 0)$, $M2_0 = (1, -1)$, and $M3_0 = (0, -1)$, one can observe that at each quadrant crossing the clockwise movement code reinitialization is

$$M1_{j+1} \leftarrow M3_j, \ M2_{j+1} \leftarrow (M2[2]_j, -M2[1]_j),$$
$$M3_{j+1} \leftarrow (M3[2]_j, -M3[1]_j).$$

To adapt the above basic algorithm[1] for arbitrary circular arcs having noninteger radii and center points, initialization must be considered in more detail. The general circle will be $(x-a)^2 + (y-b)^2 = r^2$ with a starting point (x_s, y_s) and terminating point (x_t, y_t) given on the circumference. Direction of rotation will be clockwise ($D = 1$) or counterclockwise ($D = -1$).

As the path to be calculated must consist of integer mesh points, it is necessary to determine the mesh points closest to the starting and terminating points. Any point (x, y) on the circumference lies within a unit square having mesh point corners of

$$C: \{(\lfloor x \rfloor, \lfloor y \rfloor) \ (\lfloor x \rfloor, \lceil y \rceil) \ (\lceil x \rceil, \lfloor y \rfloor) \ (\lceil x \rceil, \lceil y \rceil)\}$$

where $\lfloor x \rfloor$ is the greatest integer equal to or less than x and $\lfloor x \rfloor$ is the least integer equal to or greater than x. If x and/or y are integers, the unit square degenerates to a single point or a unit length line and the four member set C will contain only one or two unique elements. The closest mesh point (X', Y') is that point from C which minimizes the difference:

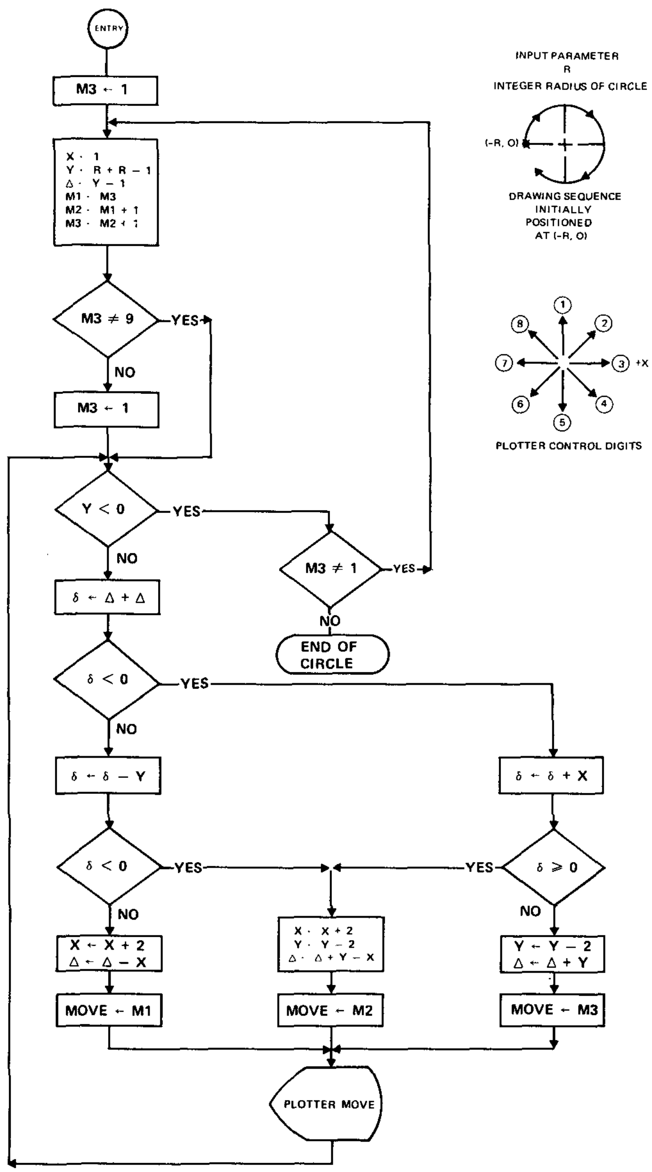
$$|(x' - a)^2 + (y' - b)^2 - r^2| \quad (x', y') \ni C.$$

Translating the closest mesh points to an origin coincident with the center of the circle then gives closest starting and terminating points of

$$\hat{X}_s = X_s' - a, \ \hat{Y}_s = Y_s' - b,$$
$$\hat{X}_t = X_t' - a, \ \hat{Y}_t = Y_t' - b.$$

[1] A flow chart for the basic integer, full circle case [3] which takes advantage of the control format codes for incremental digital plotting with the IBM 1627 is shown in Figure 3.

Fig. 3.



Since all calculations are based upon a standard first quadrant clockwise case, it is necessary to transform any other situation to the normalized case. Table I gives the transformation to obtain (X_s, Y_s) and (X_t, Y_t) from $(\hat{X}_s, \hat{Y}_s)$ and $(\hat{X}_t, \hat{Y}_t)$. Table I also gives a quadrant indicator, q, from which the number of quadrant crossings between the two points can be determined together with the initial movement codes associated with an arbitrary point.

The number of quadrant crossings is first calculated as follows:

$$Q^* = 4 \mid (q_t - q_s),$$

then, to consider $Q^* = 0$, the possible ambiguity is resolved by:

(a) $Q = Q^* - 1$ if $Q^* \neq 0$, or if $Q^* = 0$ and either $X_t \geq X_s$ and $Y_t < Y_s$ or $X_t > X_s$ and $Y_t \leq Y_s$;

(b) $Q = 3$ otherwise (i.e. $Q^* = 0$ and $X_t \leq X_s$ and $Y_t \geq Y_s$)

For simplicity, the above defaults to a full circle the case when the starting and terminating points differ yet have the same closest mesh point.

The final point has been selected when no further quadrant crossings remain (i.e. $Q_j < 0$) and $X_t \leq X_i$ and $Y_t \geq Y_i$.

3. Algorithm

Given the arc's starting and terminating points, (x_s, y_s) and (x_t, y_t), on the circle circumference together with the circle's center point (a, b) and direction of rotation (D), the display mesh point selection algorithm can be summarized as follows for the general case:

Notation

- X_i is the "constrained circle's" translated abscissa value at the ith step of normalized clockwise movement in the first quadrant.
- Y_i is the "constrained circle's" translated ordinate value at the ith step of normalized clockwise movement in the first quadrant.
- Q_j is the number of quadrants remaining to be traversed.
- $M1$ represents relative 0° movement for normalized clockwise, first-quadrant incrementation.
- $M2$ represents relative 315° movement for normalized clockwise, first-quadrant incrementation.
- $M3$ represents relative 270° movement for normalized clockwise, first-quadrant incrementation.
- Δ_i is the signed difference $\{[(X_i+1)^2 + (Y_i-1)^2] - R^2]\}$. The sign of Δ_i indicates whether the point (X_i+1, Y_i-1) is inside or outside of the true circle.
- δ is the signed difference $\{[(X_i+1)^2 + Y_i^2] - R^2\} + \Delta_i$. The sign of δ indicates which of the two points (X_i+1, Y) or (X_i+1, Y_i-1) is closest to the true circle.
- δ' is the signed difference $\{[X_i+(Y_i-1)^2] - R^2\} + \Delta_i$. The sign of δ' indicates which of the two points (X_i, Y_i-1) or (X_i+1, Y_i-1) is closest to the true circle.

Initialization

1. Determine closest mesh points (X_s', Y_s') and (X_t', Y_t') from (x_s, y_s) and (x_t, y_t) by finding their respective unit square corner mesh points from C_s and C_t which minimize

   $$\mid [(x'-a)^2 + (y'-b)^2] - [(x-a)^2 + (y-b)^2] \mid$$

   from $C(x', y')$: $\{(\lfloor x \rfloor, \lfloor y \rfloor)\ (\lfloor x \rfloor, \lceil y \rceil)\ (\lceil x \rceil, \lfloor y \rfloor)\ (\lceil x \rceil, \lceil y \rceil)\}$.

2. Translate to zero centered circle coordinates:

   $$\hat{X}_s = X_s' - a,\ \hat{Y}_s = Y_s' - b,\ \hat{X}_t = X_t' - a,\ \hat{Y}_t = Y_t' - b.$$

3. Transform $(\hat{X}_s, \hat{Y}_s)$ and $(\hat{X}_t, \hat{Y}_t)$ to normalized, first quadrant, clockwise coordinates per Table I to determine

   $X_s = X_0,\ Y_s = Y_0,\ q_s,\ M1_s = M1_0,\ M2_s = M2_0,\ M3_s = M3_0,$ and $X_t,\ Y_t,\ q_t$.

4. Calculate the number of quadrant crossings from $Q^* = 4 \mid (q_t - q_s)$ as

   $Q_0 = 3$, if $Q^* = 0$ and $X_t \leq X_s$ and $Y_t \geq Y_s$,
   $Q_0 = Q^* - 1$, otherwise.

5. Calculate the initial decision difference Δ_0 as

$$\Delta_0 = [(X_s+1)^2 + (Y_s-1)^2] - [(x_s-a)^2 + (y_s-b)^2]$$
$$= \{[(X_s'-a)^2 + (Y_s'-b)^2] - [(x_s-a)^2 + (y_s-b)^2]\} + 2(X_s - Y_s + 1).$$

6. Set the direction of rotation indicator as

$D = 1$, for clockwise rotation,
$D = -1$, for counterclockwise rotation.

Incremental Stepping Loop

1. If $Q_j \geq 0$, i.e. not in final quadrant, go to step 2. Otherwise:
   a. If $X_t > X_i$ or $Y_t < Y_i$, go to step 2.
   b. Otherwise, terminate.
2. If $Y_i \geq \frac{1}{2}$, i.e. quadrant not complete, go to step 3. Otherwise, reinitialize to

$$X_0 \leftarrow -Y_i, \quad Y_0 \leftarrow X_i, \quad \Delta_0 \leftarrow \Delta_i - 4X_i, \quad Q_{j+1} \leftarrow Q_j - 1$$
$$M1_{j+1} \leftarrow M3_j, \quad M2_{j+1} \leftarrow D \cdot (M2[2]_j, -M2[1]_j),$$
$$M3_{j+1} \leftarrow D \cdot (M3[2]_j, -M3[1]_j)$$

   and return to step 1.
3. a. If $\Delta_i \leq 0$, calculate

$$\delta \leftarrow 2\Delta_i + 2Y_i - 1,$$

   and if $\delta \leq 0$ move $M1$, if $\delta > 0$ move $M2$.
   b. If $\Delta_i > 0$ calculate

$$\delta' \leftarrow 2\Delta_i - 2X_i - 1;$$

   and if $\delta' \leq 0$ move $M2$, if $\delta' > 0$ move $M3$.
4. a. If movement was $M1$, then

$$X_{i+1} \leftarrow X_i + 1, \quad Y_{i+1} \leftarrow Y_i, \quad \Delta_{i+1} \leftarrow \Delta_i + 2X_{i+1} + 1.$$

   b. If movement was $M2$, then

$$X_{i+1} \leftarrow X_i + 1, \quad Y_{i+1} \leftarrow Y - 1,$$
$$\Delta_{i+1} \leftarrow \Delta_i + 2X_{i+1} - 2Y_{i+1} + 2.$$

   c. If movement was $M3$, then

$$X_{i+1} \leftarrow X_i, \quad Y_{i+1} \leftarrow Y_i - 1, \quad \Delta_{i+1} \leftarrow \Delta_i - 2Y_{i+1} + 1.$$

5. Return to step 1.

4. Remarks

Other incremental algorithms for displaying figures have been described elsewhere [1–12]. Pitteway [10] first published a general solution for simply displaying conic sections incrementally by differencing the general polynominal and using a bi-directional movement control by octant. In return for only a very slight overhead when crossing the additional four 45° octant boundaries, his bi-directional control methodology offers the most efficient inner stepping loop of any algorithm of which the author is aware. A variant of his bi-directional method for vertical error minimization can be used with the clockwise, first quadrant normalization technique described here for squared error minimization to also achieve a three addition inner loop for both square and diagonal moves. From 90° to 45° one uses the two variables $2(X - Y) + 1$ and $2X + 1$, then from 45° to 0° one tracks the variables $2(Y - X) - 1$ and $2Y - 1$. The square movement code is changed at 45° while the diagonal move is changed at 0° (i.e. at $2(X - Y) + 1 > 0$ and at $2Y - 1 < 0$). For implementation symmetry, the decision difference, d, is $d = \frac{1}{2}\delta$ between 90° and 45° and is $d = -\frac{1}{2}\delta'$ between 45° and 0°.

Addressing each conic section separately, Metzger [9] provided a set of compact algorithms for incremental display, though, in all but the straight line case, quadratic or square root calculation is required.

Jordan, Lennon, and Holm [8] have unified with very good clarity a generalized but efficient solution for incrementally displaying, with arbitrary step sizes (Δx, Δy) explicitly covered, any curve possessing continuous derivatives. As special cases, they describe a tri-directional movement control polynominal display algorithm functionally comparable to that of Pitteway and a circle display algorithm functionally comparable to the one presented here. Though comparable, the Jordan and Pitteway algorithms do differ in error criteria in that Jordan minimizes function residue or, for circles, squared error while Pitteway minimizes vertical or horizontal error.

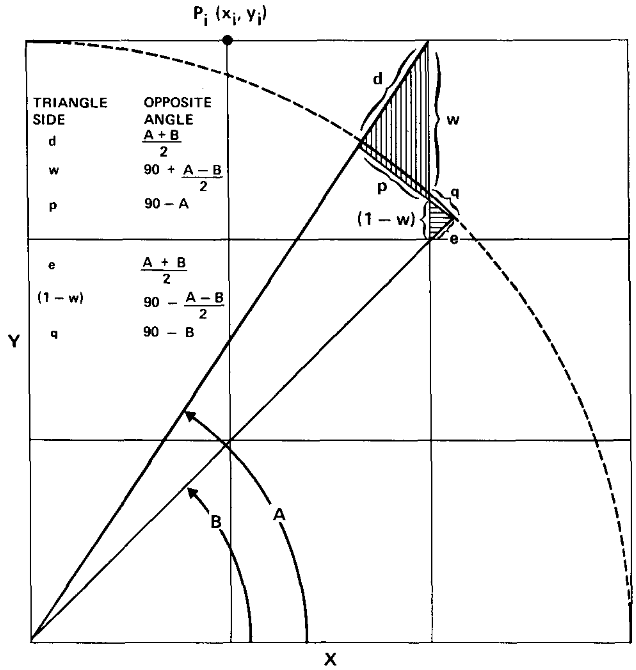
A linear algorithm using only integer calculation has been described by Denert [7] for polygon approximation of circles. In two short notes [11, 12] Pitteway has briefly discussed variants and an alternative to the Denert algorithm using [1, 2, 10].

Cohen [4, 5] has presented a method of generating a sequence of regularly spaced points on a circle using the iteration $P_{(i+1)} = TP_i$ where T is a 2×2 matrix. Successive points then are connected by straight line segments to approximate the curve. For other conic sections, the method generates variable density points

Table I. Transformation Table: Normalize to Standard Clockwise First Quadrant Case.

Quadrant	Rotation	$\hat{X}$	$\hat{Y}$	Index	X	Y	q	$M1$	$M2$	$M3$
III	CCW	<0	<0	0	$\|\hat{Y}\|$	$\|\hat{X}\|$	3	0, −1	1, −1	1, 0
II	CCW	<0	≥0	1	$\|\hat{X}\|$	$\|\hat{Y}\|$	2	−1, 0	−1, −1	0, −1
IV	CCW	≥0	<0	2	$\|\hat{X}\|$	$\|\hat{Y}\|$	0	1, 0	1, 1	0, 1
I	CCW	≥0	≥0	3	$\|\hat{Y}\|$	$\|\hat{X}\|$	1	0, 1	−1, 1	−1, 0
III	CW	<0	<0	4	$\|\hat{X}\|$	$\|\hat{Y}\|$	2	−1, 0	−1, 1	0, 1
II	CW	<0	≥0	5	$\|\hat{Y}\|$	$\|\hat{X}\|$	3	0, 1	1, 1	1, 0
IV	CW	≥0	<0	6	$\|\hat{Y}\|$	$\|\hat{X}\|$	1	0, −1	−1, −1	−1, 0
I	CW	≥0	≥0	7	$\|\hat{X}\|$	$\|\hat{Y}\|$	0	1, 0	1, −1	0, −1

Fig. 4.



with points most closely spaced in those portions of the curve having the greatest curvature.

In personal communications [March and April, 1975], Pitteway has shown the author an alternative interpretation which provides additional insight into the difference between squared and vertical error minimization. The expressions for δ and δ' can be refactored as follows:

$$\delta = 2\{(X_i + 1)^2 + (Y_i - \tfrac{1}{2})^2 - (r^2 - \tfrac{1}{4})\},$$
$$\delta' = 2\{(X_i + \tfrac{1}{2})^2 + (Y_i - 1)^2 - (r^2 - \tfrac{1}{4})\},$$

which is twice the residue of a circle of radius $(r^2 - \frac{1}{4})^{\frac{1}{2}}$ evaluated respectively at the points $(X + 1, Y - \frac{1}{2})$ and $(X + \frac{1}{2}, Y - 1)$. The algorithm essentially is using the decision rule for squared error minimization:

If $\delta \leq 0$, move m_1
If $\delta' > 0$, move m_3
Otherwise, move m_2

The Pitteway algorithm, applied to circles, effectively evaluates these "step and a half" residues using a radius of r rather than $(r^2 - \frac{1}{4})^{\frac{1}{2}}$ and applies the same decision rule. In implementation, the difference between vertical and squared criteria amounts only to a bias of $\frac{1}{4}$ in the initial value of δ.

When a circle's center point and radius are limited to only integer values, identical display points will be selected regardless of whether one minimizes squared, vertical, or radial error. Depending upon which error criterion is used, different display points occasionally will be selected in the general noninteger case. The two circles $(x - 4.53)^2 + (y + 3.6)^2 = (10)^2$ and $x^2 + y^2 = (4.925)^2$ illustrate the differences. In the first example, clockwise movement from (14, −2) will be to (15, −3) by either vertical or radial criteria but will be to (14, −3) by the squared criteria of the Jordan or Bresenham algorithms. In the second example, clockwise movement from (1, 5) will be to (2, 4) by either squared or radial criteria but will be to (2, 5) by the vertical criteria of the Pitteway or Metzger algorithms.

Appendix

It remains to be shown that minimizing the difference between the squares of the true and constrained radii also minimizes the linear difference between the two radii. Figure 4 shows the case in which the circle passes between the points $(X + 1, Y)$ and $(X + 1, Y - 1)$.

Using the notation in Figure 4, application of the trigonometric law of sines for p and q and the law of cosines for d^2 and e^2 gives

$$d^2 = w^2 + p^2 - 2\,pw \cos \tfrac{1}{2}\,(A + B),$$

$$d^2 = w^2 + p\left[w \frac{\sin\,(90 - A)}{\sin\,[90 + \frac{1}{2}(A - B)]} - 2w \cos \tfrac{1}{2}(A + B)\right]$$

$$d^2 = w^2 + pw\left[\frac{\cos A - 2\cos \frac{1}{2}(A + B) \cos \frac{1}{2}(A - B)}{\cos \frac{1}{2}(A - B)}\right],$$

$$d^2 = w^2 - pw \frac{\cos B}{\cos \frac{1}{2}(A - B)},$$

$$d^2 = w^2\left[1 - \frac{\cos A \cos B}{\cos^2 \frac{1}{2}(A - B)}\right]. \quad (1)$$

$$e^2 = (1 - w)^2 + q^2 - 2q\,(1 - w) \cos \tfrac{1}{2}(A + B),$$

$$e^2 = (1 - w)^2 + q\left\{(1 - w) \frac{\sin\,(90 - B)}{\sin\,[90 - \frac{1}{2}(A - B)]} - 2(1 - w) \cos \tfrac{1}{2}(A + B)\right\},$$

$$e^2 = (1 - w)^2 + q(1 - w) \cdot \left[\frac{\cos B - 2\cos \frac{1}{2}(A + B) \cos \frac{1}{2}(A - B)}{\cos \frac{1}{2}(A - B)}\right],$$

$$e^2 = (1 - w)^2 - q(1 - w)\left[\frac{\cos A}{\cos \frac{1}{2}(A - B)}\right],$$

$$e^2 = (1 - w)^2\left[1 - \frac{\cos A \cos B}{\cos^2 \frac{1}{2}(A - B}\right]. \quad (2)$$

From (1) and (2)

$$d^2 + e^2 = [w^2 + (1 - w)^2]\left[1 - \frac{\cos A \cos B}{\cos^2 \frac{1}{2}(A - B)}\right] \quad (3)$$

Now $90° > A > B \geq 0°$ and $1 \geq w \geq 0$. Hence

$$1 \geq [w^2 + (1 - w)^2] \geq \tfrac{1}{2}$$

and

$$1 > \left[1 - \frac{\cos A \cos B}{\cos^2 \frac{1}{2}(A - B)}\right] > 0$$

so that

$$1 > d^2 + e^2 > 0. \tag{4}$$

In a like manner, (4) can be derived for the δ' situation in which the point $(X + 1, Y - 1)$ and the line segment d' are exterior to the circle and the point $(X, Y - 1)$ and the line segment e' are interior to the circle.

Now

$$\delta = \{[(X+1)^2 + Y^2] - R^2\} + \{[(X+1)^2 + (Y-1)^2] - R^2\}.$$

Hence from Figure 4

$$\delta = [(R+d)^2 - R^2] + [(R-e)^2 - R^2],$$
$$\delta = 2R(d-e) + (e^2 + d^2),$$

so that

$$d^2 + e^2 = \delta - 2R(d-e). \tag{5}$$

From (4) and (5) then

$$1 > \delta - 2R(d-e) > 0$$

and

$$\delta/2R > (d-e) > (\delta - 1)/2R. \tag{6}$$

Therefore

$$\text{if } \delta \le 0 \text{ then } (d-e) < 0 \text{ hence } d < e, \tag{7}$$
$$\text{if } \delta \ge 1 \text{ then } (d-e) > 0 \text{ hence } d > e. \tag{8}$$

When δ is integer valued only, one has the simple decision rule

$$\text{if } \delta \le 0 \text{ then } d < e, \tag{9}$$
$$\text{if } \delta > 0 \text{ then } d > e. \tag{10}$$

In a like manner, (9) and (10) can be shown to hold for the δ' situation in which the point $(X + 1, Y - 1)$ and line segment d' are exterior to the true circle.

The circle algorithm thus minimizes both the linear difference between the true and constrained radii and the difference between the squares of the true and constrained radii when the circle has an integer radius and integer center point. From eqs. (4) and (5) and the sum of the square roots of eqs. (1) and (2) (i.e. $0 < d + e < 1$), it can be shown that if $\delta \le 0$ then $d < \frac{1}{2}$, and if $\delta \ge 1$ then $e < \frac{1}{2}$, which agrees with the maximum radial error found experimentally in [8] and [9].

When δ can assume noninteger values in the range $0 < \delta < 1$, radial and squared criteria need not coincide as one quickly can verify when stepping clockwise from (1, 5) on the circle $x^2 + y^2 = (4.93)^2$. Should decision rule (10) be used when $0 < \delta < 1$, e will be selected when in fact d possibly could be the lesser of the two radial error measures. The difference should be negligible for larger radii since, for $0 < \delta < 1$, one can observe from equations (1), (2), (4), and (5) that $e < 1/2 + 1/4r$ and $|d - e| < 1/2r$.

When incrementally displaying the circle $(x - 4.53)^2 + (y + 3.6)^2 = (10)^2$, movement clockwise from (14, −2) or counterclockwise from (15, −4) to the point (14, −3) provides an example of minimum squared error coinciding with a nonminimum radial error, or normal distance to the curve, in excess of $\frac{1}{2}$ unit. At (14, −3) the squared error is $\simeq$ 9.96 while radial error is $\simeq$0.511. At (15, −3) the squared error is $\simeq$9.98 while radial error is $\simeq$ 0.487. One also can observe in this case that the selected points for the full circle do not exhibit the quadrant to quadrant (or octant to octant) symmetry found when a circle's center point and radius are integers.

Acknowledgments. The suggestions, comments and insights of M.L.V. Pitteway, W.M. Newman, and the ACM referees were of considerable assistance. Helpful also were various discussions with D. van Alderwerelt, F.K. Allen, H.P. Barnett, B.M. Burke, D. Clark, and C.L. Fisher of IBM. R.L. Pratt's June 1965 communication concerning the comparable forcing situation overlooked in [2] led to recognition of cases 2 and 4 shown in Figure 2. W.M. Newman suggested Figure 2 during the review process.

Received June 1974; revised September 1975

References

1. Bresenham, J.E. An incremental algorithm for digital plotting. Presented at ACM Nat. Conf. (Aug. 1963).
2. Bresenham, J.E. Algorithm for computer control of a digital plotter. *IBM Systems J. 4*, 1 (1965), 25–30.
3. Bresenham, J.E. A linear, incremental algorithm for digitally plotting circles. Tech. Rep. No. TR02.286, IBM General Products Div., San Jose, Calif. Jan 27, 1964.
4. Cohen, D. On linear difference curves. Proc. Int. Symp. CG-70, Vol. I, Brunel U. Uxbridge, England, April 1970.
5. Cohen, D. Incremental methods for computer graphics. Tech. Rep. ESD-TR69-193 Harvard U., Cambridge, Mass., April 1969.
6. Danielsen, P.E. Incremental curve generation. *IEEE Trans. Computers C-19*, 9 (Sept. 1970), 783–793.
7. Denert, E. A method for computing points on a circle using only integers. *Comptr. Graphics and Image Processing 2*, 1 (Aug. 1973), 83–91.
8. Jordan, B.W., Lennon, W.J., and Holm, B.C. An improved algorithm for the generation of nonparametric curves. *IEEE Trans. Computers C-22*, 12 (Dec. 1973), 1052–1060.
9. Metzger, R.A. Computer generated graphic segments in a raster display. Proc. AFIPS 1969 SJCC, AFIPS Press, Montvale, N.J., pp. 161–172.
10. Pitteway, M.L.V. Algorithm for drawing ellipses or hyperbolae with a digital plotter. *Comptr. J. 10*, 3 (Nov. 1967), 282–289.
11. Pitteway, M.L.V. Integer circles, etc.—three move extension of Bresenham's algorithm. *Comptr. Graphics and Image Processing 3*, 3 (Sept. 1974), 260–261.
12. Pitteway, M.L.V. Integer circles—some further thoughts. *Compt. Graphics and Image Processing 3*, 3 (Sept. 1974), 262–265.