

Σχολή Θετικών Επιστημών
Τμήμα Μαθηματικών
Εθνικόν και Καποδιστριακόν Πανεπιστήμιο Αθηνών

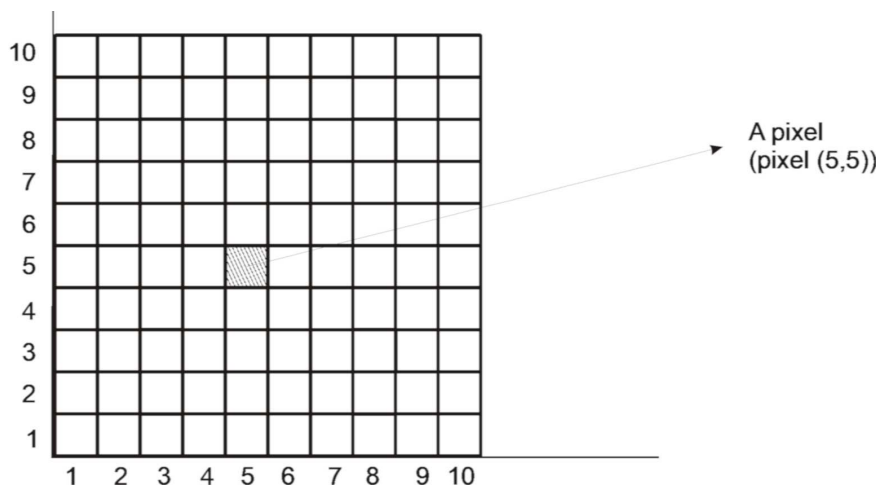
453. Γραφικά με Η/Υ

Σχεδιασμός Βασικών Σχημάτων
Ευθεία

Σχεδιασμός βασικών σχημάτων

Η συντριπτική πλειοψηφία των μεθόδων εξόδου που χρησιμοποιούμε στην εποχή μας, βασίζεται σε συσκευές που εφαρμόζουν μεθόδους Raster Scan. Με πιο απλά λόγια κάθε τι που θέλουμε να απεικονίσουμε το βλέπουμε σαν ένα είδος σύγχρονου ψηφιδωτού. Και τα αντικείμενα που θέλουμε να σχεδιάσουμε, είναι μια σύνθεση τέτοιων μικρών ψηφιδών, των pixel ή στα ελληνικά εικονοστοιχεία.

Τα pixel είναι μικρά τετράγωνα (όταν αναφερόμαστε σε οθόνες) και το πλήθος τους καθορίζει την ανάλυση της εικόνας μας. Μια εικόνα που έχει ανάλυση $m \times n$, αποτελείται από m γραμμές και n στήλες από τέτοια τετράγωνα. Για αυτό είναι και πολύ εύκολο να φανταστούμε ένα τέτοιο τετράγωνο σαν το στοιχείο ενός πίνακα, και τελικά να φανταζόμαστε κάθε εικόνα σαν έναν πίνακα.



Σχήμα 1: Μια οθόνη ανάλυσης 10x10 και ο προσδιορισμός ενός pixel σε αυτή

Ορισμός: Όταν αναφερόμαστε στη θέση ενός pixel με τις συντεταγμένες (x, y) θα εννοούμε ότι το κέντρο του τετραγώνου βρίσκεται σε αυτή τη θέση πάνω στο πλέγμα, που συνθέτει την οθόνη μας.

Παρατήρηση: Το **pixel** για τα γραφικά είναι ότι είναι το σημείο για τα μαθηματικά. Αν και πρακτικά έχει διαστάσεις, σε αντίθεση με το σημείο, θεωρούμε ότι είναι ισοδύναμο.

Με αυτόν τον τρόπο απεικόνισης, χρειαζόμαστε να σκεφτούμε τρόπους για να μπορέσουμε να σχεδιάσουμε ακόμα και απλά σχήματα, όπως ένα ευθύγραμμο τμήμα ή έναν κύκλο. Καθώς οι τρόποι που γνωρίζουμε από τα μαθηματικά δεν μπορούν να βρουν άμεση εφαρμογή.

Οι αλγόριθμοι αυτοί θα πρέπει να επιλέγουν το καλύτερο δυνατό pixel έτσι ώστε το τελικό αποτέλεσμα να είναι όσο πιο κοντά στο σχήμα που θέλουμε. Αλλά ταυτόχρονα ο υπολογισμός των pixel να γίνεται γρήγορα και χωρίς να χρειάζονται πολλές πράξεις ώστε να έχουμε σφάλματα λόγω της αριθμητικής του υπολογιστή και να χρησιμοποιούν ακεραίους.

Σχεδιασμός Ευθείας

Για τον σχεδιασμό ευθυγράμμων τμημάτων υπάρχουν πολλοί αλγόριθμοι που θα μπορούσαμε να χρησιμοποιήσουμε, αλλά πρέπει να επιλέξουμε κάποιον που δεν θα παρουσιάζει προβλήματα κατά την υλοποίησή του.

Αλγόριθμος Εξίσωσης Ευθείας

Για παράδειγμα θα μπορούσαμε να χρησιμοποιήσουμε την αλγεβρική εξίσωση της ευθείας για να προσδιορίσουμε ποια σημεία της ευθείας θα αντιστοιχούν σε pixel της οθόνης που θα επιλέξουμε να φωτίσουμε.

Αν θέλουμε να σχεδιάσουμε το ευθύγραμμο τμήμα $\overline{P_1P_2}$ με $P_1 = (x_1, y_1)$ και $P_2 = (x_2, y_2)$, τότε θα ισχύει :

$$\frac{y - y_1}{x - x_1} = \frac{y_2 - y_1}{x_2 - x_1} \Rightarrow y = \frac{y_2 - y_1}{x_2 - x_1}x + \frac{y_1x_2 - y_2x_1}{x_2 - x_1}$$

Και θέτοντας $S = \frac{y_2 - y_1}{x_2 - x_1}$ (κλίση) και $C = \frac{y_1x_2 - y_2x_1}{x_2 - x_1}$ έχουμε $y = Sx + C$. Και με βάση αυτή την εξίσωση γράφουμε τον ακόλουθο αλγόριθμο:

Βήμα 1: Διάβασε $P_1 = (x_1, y_1)$ και $P_2 = (x_2, y_2)$

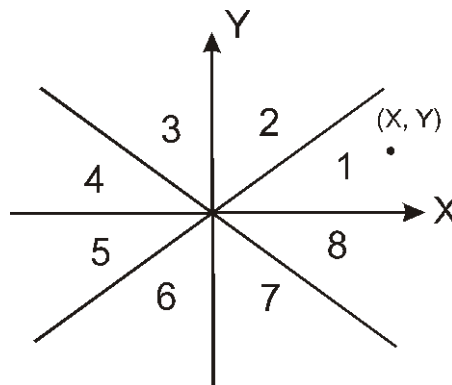
Βήμα 2: Υπολόγισε $S = \frac{y_2 - y_1}{x_2 - x_1}$ και $C = \frac{y_1x_2 - y_2x_1}{x_2 - x_1}$

Βήμα 3: Για x από x_1 έως x_2 με βήμα +1 . Υπολόγισε το $y = \text{round}(Sx + C)$ και φώτισε το pixel (x, y) .

Αν όμως χρησιμοποιήσουμε αυτόν τον αλγόριθμο θα παρατηρήσουμε ότι για ζεύγη σημείων το ευθύγραμμο τμήμα μας θα παρουσιάζει κενά και δεν θα είναι συνεχές. Επίσης κατά τον υπολογισμό των τιμών πραγματοποιούμε πολλαπλασιασμούς και διαιρέσεις που αυξάνουν την πολυπλοκότητα και εισάγουν και αριθμητικά σφάλματα στα δεδομένα μας. Για αυτούς τους λόγους η χρήση της αλγεβρικής εξίσωσης της ευθείας σε υπολογιστή είναι τελικά λάθος κίνηση. Ευτυχώς έχουν ήδη αναπτυχθεί άλλοι τρόποι που θα μας βοηθήσουν να υπολογίσουμε ποια πιξελ θα πρέπει να φωτίσουμε στην οθόνη για να σχηματιστεί το ζητούμενο ευθύγραμμο τμήμα. Ένα τέτοιος είναι ο αλγόριθμος ευθυγράμμων τμημάτων του Bresenham.

Αλγόριθμος Bresenham για ευθύγραμμο τμήμα

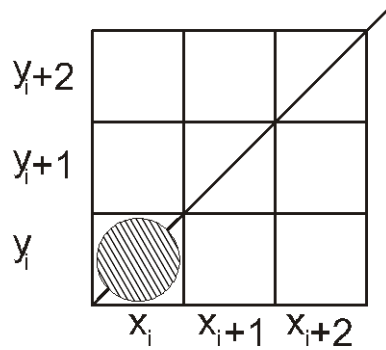
Το 1965 ο J.E. Bresenham παρουσίασε την πρώτη DDA μέθοδο (Digital Differential Analyzer) για το σχεδιασμό ευθυγράμμου τμήματος. Εκεί παρουσίαζε τη μορφή που έχει ο αλγόριθμός του για ευθύγραμμο τμήματα που ανήκουν στο πρώτο οκταμόριο. Δηλαδή που η γωνία που σχηματίζουν με τον οριζόντιο άξονα είναι από 0° έως 45° . Καθώς και τι μετατροπές που χρειάζονται να γίνουν για να μπορούμε να σχεδιάσουμε ευθύγραμμο τμήματα κάθε κλίσης.



Σχήμα 2: Τα οκταμόρια του επιπέδου

Προσδιορίζουμε το σχεδιασμό της ευθείας στο 1ο οκταμόριο και στη συνέχεια με κατάλληλη τροποποίηση, που θα την περιγράψουμε παρακάτω, μπορούμε να επεκταθούμε και στα άλλα οκταμόρια.

Έστω λοιπόν ότι θέλουμε να σχεδιάσουμε ένα ευθύγραμμο τμήμα μεταξύ των pixel στο 1ο οκταμόριο $P_1(x_1, y_1)$ και $P_2(x_2, y_2)$, όπου $x_i, y_i \in \mathbb{Z}, i = 1, 2$ και έστω ότι έχει φωτισθεί στην οθόνη το pixel (x_i, y_i)



Δεδομένου ότι ο σχεδιασμός του ευθυγράμμου τμήματος συνίσταται στον επιλεκτικό φωτισμό διαδοχικών pixels ώστε αυτά να σχηματίζουν ευθύγραμμο τμήμα, θεωρούμε ότι αφού βρισκόμαστε στο 1ο οκταμόριο εάν ήδη είναι φωτισμένο το pixel (x_i, y_i) τα επόμενα δυνατά προς φωτισμό pixels θα είναι τα $(x_i + 1, y_i)$ ή $(x_i + 1, y_i + 1)$.

Παρατηρούμε δηλαδή ότι οπωσδήποτε θα αυξηθεί κατά 1 η συντεταγμένη x και η συντεταγμένη y ή θα παραμείνει ίδια η x θα αυξηθεί κι αυτή κατά 1. Θα πρέπει λοιπόν να καθορίσουμε μία μεταβλητή σφάλματος, η οποία ανάλογα να αποφάνεται εάν θα αυξηθεί ή όχι η συντεταγμένη y .

Δημιουργία μεταβλητής σφάλματος e_i

Η εξίσωση μεταξύ των σημείων (x_i, y_i) και $(x_i + 1, y)$ είναι

$$y = s(x_i + 1) + c$$

$$s = \frac{y - y_i}{x_i + 1 - x_i} = \frac{\Delta y}{\Delta x} \leq 1$$

η κλήση της ευθείας, και

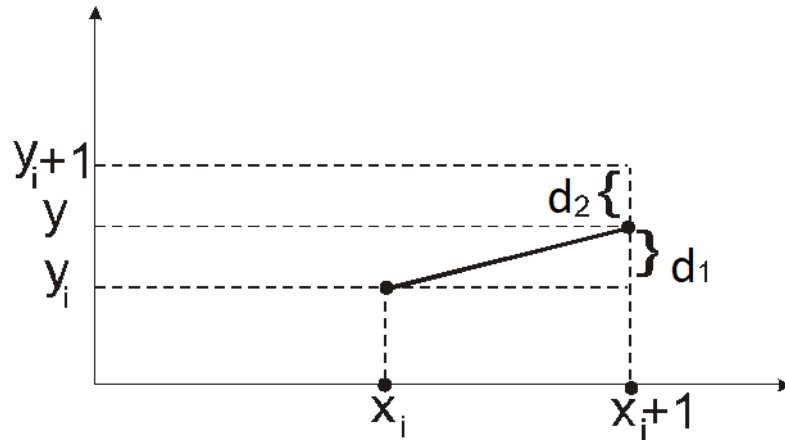
$$c = \frac{y_i(x_i + 1) - yx_i}{x_i + 1 - x_i}$$

Υπολογίζουμε τις αποστάσεις d_1, d_2 και τη διαφορά τους $d_1 - d_2$

$$d_1 = y - y_i = s(x_i + 1) + c - y_i$$

$$d_2 = (y_i + 1) - y = y_i + 1 - s(x_i + 1) - c$$

$$d_1 - d_2 = 2s(x_i + 1) - 2y_i + 2c - 1$$



Παρατηρούμε ότι ανάλογα με το πρόσημο της διαφοράς $d_1 - d_2$ μπορούμε να επιλέξουμε το επόμενο προς φωτισμό pixel. Εάν ονομάσουμε μεταβλητή σφάλματος e_i την ποσότητα $e_i = \Delta x(d_1 - d_2)$ τότε θα έχουμε :

e_i	Επόμενο προς φωτισμό pixel
< 0	$(x_i + 1, y_i)$
≥ 0	$(x_i + 1, y_i + 1)$

Προσπάθεια δημιουργίας αναδρομικού τύπου

Σε κάθε βήμα σχεδιασμού της ευθείας θα είναι καλό να μπορούμε να υπολογίζουμε την καινούργια μεταβλητή σφάλματος συναρτήσει της προηγούμενης, γι' αυτό προσπαθούμε να δημιουργήσουμε έναν αναγωγικό τύπο. Ισχύει ότι:

$$e_i = \frac{\Delta y}{s}(2s(x_i + 1) - 2y_i + 2c - 1) = 2\Delta y x_i - 2\Delta x y_i + c'$$

Όπου $c' = 2\Delta y + \Delta x(2c - 1)$

Για το e_{i+1} θα ισχύει ότι $e_{i+1} = 2\Delta y x_{i+1} - 2\Delta x y_{i+1} + c'$.

Τελικά $e_{i+1} - e_i = 2\Delta y(x_{i+1} - x_i) - 2\Delta x(y_{i+1} - y_i)$

Η αρχική τιμή για το e_1 υπολογίζεται από τη σχέση:

$$e_1 = 2\Delta y x_1 - 2\Delta x y_1 + 2\Delta y + \Delta x(2c - 1) = 2\Delta y - \Delta x$$

Παρατηρούμε ότι :

e_i	Επόμενη μεταβλητή σφάλματος e_2
< 0	$e_1 + 2\Delta y$
≥ 0	$e_1 + 2\Delta y - 2\Delta x$

Αυτά μας οδηγούν στο ακόλουθο αλγόριθμο για το 1ο οκταμόριο

```
input (x1,y1),(x2,y2)
dx = x2-x1;
dy = y2-y1
x = x1
y = y1
c1 = 2*dy
e = 2*dy-dx
c2 = e-dx
while x <= x2
    plot(x,y)
    x = x+1;
    if e < 0
        e = e + c1;
    else
        y = y+1;
        e = e + c2;
    end
end
```

Υπολογιστική Πολυπλοκότητα

Είναι φανερό η υπεροχή του αλγόριθμου Bresenham σε πράξεις, καθώς χρειάζονται μόνο προσθαφαιρέσεις και ένας πολλαπλασιασμός με το 2, πράξη που μπορεί να υποκατασταθεί με ένα αριστερό shift. Δεν απαιτούνται καθόλου διαιρέσεις.

Η επέκταση του αλγόριθμου για τα υπόλοιπα οκταμόρια γίνεται βασιζόμενη σε κάποιες παρατηρήσεις που είναι πολύ εύκολο να κάνουμε πάνω στην αρχική ιδέα.

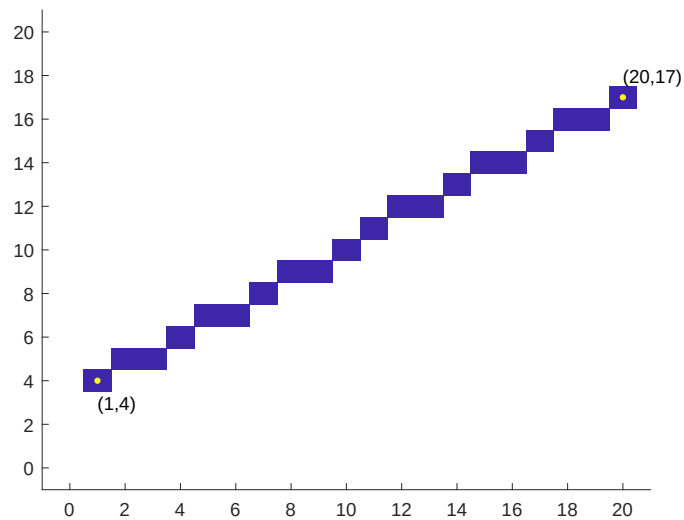
Παρατήρηση: Κάθε ευθύγραμμο τμήμα AB ορίζεται από τα σημεία $A = (x_1, y_1)$ και $B = (x_2, y_2)$. Αν θεωρήσουμε ότι το σημείο A ταυτίζεται με την αρχή των αξόνων και χωρίσουμε το επίπεδο σε οκτώ ίσα τμήματα, το τμήμα το περιέχει το σημείο B καθορίζει και το οκταμόριο που ανήκει το ευθύγραμμο τμήμα.

Παρατήρηση: Στην αρχική ιδέα του Bresenham παίζει σημαντικό ρόλο η σειρά που θα δώσουμε τα σημεία A και B. Δηλαδή ποιο είναι η αρχή και ποιο το τέλος του τμήματος. Για αυτό έχουμε 8 διαφορετικές περιπτώσεις. Όμως αν απλά θέλουμε να σχεδιάσουμε ένα ευθύγραμμο τμήμα. Δεν μας ενδιαφέρει από ποια μεριά θα ξεκινήσουμε τον σχεδιασμό του. Αλλά διαλέγουμε πάντα σαν αρχή το σημείο για το οποίο η συντεταγμένη $y = \min(y_1, y_2)$ τότε αντί για 8 περιπτώσεις έχουμε μόνο 4. Αφού τα οκταμόρια 5,6,7 και 8 ανάγονται στα 1,2,3 και 4 αντίστοιχα.

Παρατήρηση: Ο υπολογισμός των σημείων στο δεύτερο οκταμόριο γίνεται αν θεωρήσουμε ότι στη θέση των A και B, εισάγουμε τα σημεία $A_2 = (y_1, x_1)$ και $B_2 = (y_2, x_2)$ και αντί για να φωτίζουμε το pixel (x, y) που θα μας επιστρέφει ο αλγόριθμος εμείς φωτίζουμε το (y, x) .

Παρατήρηση: Για τα οκταμόρια 3 και 4, αρκεί να υποθέσουμε ότι θέλουμε να σχεδιάσουμε το συμμετρικό ευθύγραμμο τμήμα ως προς τον άξονα y και να βάλουμε τον αλγόριθμο του Bresenham να υπολογίσει για αυτό τα σημεία, αλλά εμείς θα φωτίζουμε τα συμμετρικά των υπολογισμένων.

Αφού λάβουμε υπόψιν μας αυτές τις παραδοχές, μπορούμε να υλοποιήσουμε τον ολοκληρωμένο αλγόριθμό.



Σχήμα 3: Παράδειγμα ευθύγραμμου τμήματος, με άκρα τα σημεία (1,4) και (20,17), υλοποιημένο σε Matlab

Εφαρμογές

1η) Να εφαρμοστεί ο αλγόριθμος της Εξίσωσης Ευθείας για τα ακόλουθα ζεύγη σημείων:

1. $P_1(1, 1), P_2(3, 2)$

2. $P_1(1, 1), P_2(4, 13)$

2η) Να εφαρμοστεί ο αλγόριθμός του Bresenham για το 1ο οκταμόριο για το ζεύγος $P_1(1, 1), P_2(3, 2)$.

3η) Να προσαρμόσετε τον αλγόριθμό του Bresenham έτσι ώστε να δουλεύει για το 2ο οκταμόριο και να το δοκιμάσετε για το ζεύγος $P_1(1, 1), P_2(4, 13)$

Στις επόμενες σελίδες μπορείτε να δείτε την δημοσίευση του J.E. Bresenham που πρωτοπαρουσίασε τον αλγόριθμό του.

An algorithm is given for computer control of a digital plotter.

The algorithm may be programmed without multiplication or division instructions and is efficient with respect to speed of execution and memory utilization.

Algorithm for computer control of a digital plotter

by J. E. Bresenham

This paper describes an algorithm for computer control of a type of digital plotter that is now in common use with digital computers.¹

The plotter under consideration is capable of executing, in response to an appropriate pulse, any one of the eight linear movements shown in Figure 1. Thus, the plotter can move linearly from a point on a mesh to any adjacent point on the mesh. A typical mesh size is 1/100th of an inch.

The data to be plotted are expressed in an (x, y) rectangular coordinate system which has been scaled with respect to the mesh; i.e., the data points lie on mesh points and consequently have integral coordinates.

It is assumed that the data include a sufficient number of appropriately selected points to produce a satisfactory representation of the curve by connecting the points with line segments, as illustrated in Figure 2. In Figure 3, the line segment connecting

Figure 1 Plotter movements

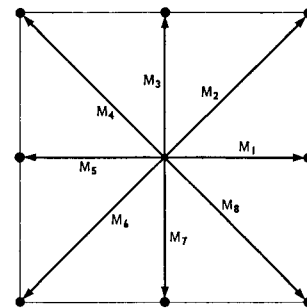
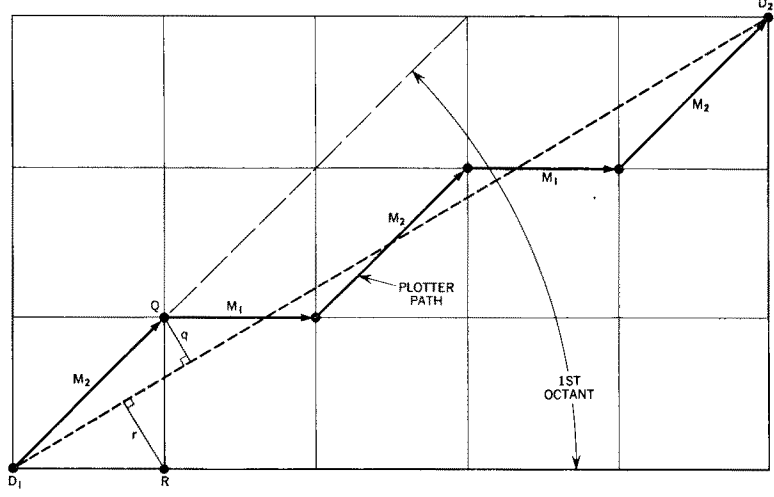


Figure 2 Curve defined by linear segments joining data points



Figure 3 Sequence of plotter movements



the two adjacent data points $D_1(x_1, y_1)$ and $D_2(x_2, y_2)$ is shown on the mesh, drawn on an enlarged scale. Also shown is the path actually taken by the plotter in accordance with the algorithm. In each instance, the mesh point nearest the desired line segment is selected. For example, since Q is closer to the line segment than R , Q is chosen as the second mesh point in the path taken by the plotter in approximating the desired segment joining D_1 and D_2 .

algorithm
for first
octant

In the first case to be considered, it is assumed that D_2 lies in the first octant, relative to a rectangular coordinate system obtained by translation of the origin to D_1 . It is apparent that the plotter movement can be accomplished by a sequence of moves involving only M_1 and M_2 , as illustrated in Figure 3.

In Figure 4, an (a, b) coordinate system obtained by translation of the origin to D_1 is shown. Consequently, the new coordinates of D_2 are $(\Delta a, \Delta b) = (x_2 - x_1, y_2 - y_1)$.

When the plotter has progressed to the point P_{i-1} , as indicated in Figure 4, the next movement is either M_1 (to the point R_i) if $r_i < q_i$, or M_2 (to the point Q_i) if $r_i \geq q_i$.

It follows from similar triangles that $r'_i - q'_i$ has the same sign as $r_i - q_i$. Since the segment $D_1 D_2$ lies in the first octant, $\Delta a > 0$. Thus, $\nabla_i = (r'_i - q'_i)\Delta a$ also has the same sign as $r_i - q_i$, and may be used for computational convenience in selecting the appropriate movement, either M_1 or M_2 . Later in the paper, ∇_i is shown to satisfy the recursive relation:

$$\left. \begin{aligned} \nabla_1 &= 2\Delta b - \Delta a \\ \nabla_{i+1} &= \begin{cases} \nabla_i + 2\Delta b - 2\Delta a & \text{if } \nabla_i \geq 0 \\ \nabla_i + 2\Delta b & \text{if } \nabla_i < 0 \end{cases} \end{aligned} \right\}, \quad (1)$$

where

$$\Delta a = x_2 - x_1, \quad \Delta b = y_2 - y_1. \quad (2)$$

The values of ∇_i computed by means of (1) and (2) are used to determine the movement of the plotter:

$$\text{if } \begin{cases} \nabla_i < 0, & \text{execute } m_1 \\ \nabla_i \geq 0, & \text{execute } m_2 \end{cases}, \quad i = 1, \dots, \Delta a, \quad (3)$$

where

$$m_1 = M_1 \quad \text{and} \quad m_2 = M_2. \quad (4)$$

Expressions (1), (2), (3), and (4) constitute the algorithm for the present case. For other octants, the right members of each equality in (2) and (4) must be modified.

Before indicating this modification, the recursive relation (1) is shown to hold. The notation employed in Figure 4 is as follows: $(a_{i-1}, \hat{b}_{i-1})$ is used to denote the coordinates of P_{i-1} . Consequently, the coordinates of R_i and Q_i are, respectively, $(a_i, \lfloor b_i \rfloor)$ and $(a_i, \lceil b_i \rceil)$, where " $\lfloor \cdot \rfloor$ " and " $\lceil \cdot \rceil$ " are used to denote the *floor* and *ceiling* operators.² Denoting the ordinate of S_i by b_i , the coordinates of S_i are (a_i, b_i) .

proof of
the recursive
relation

This notation is used to rewrite the expression for ∇_i :

$$\nabla_i = (r'_i - q'_i)\Delta a = [(b_i - \lfloor b_i \rfloor) - (\lceil b_i \rceil - b_i)]\Delta a.$$

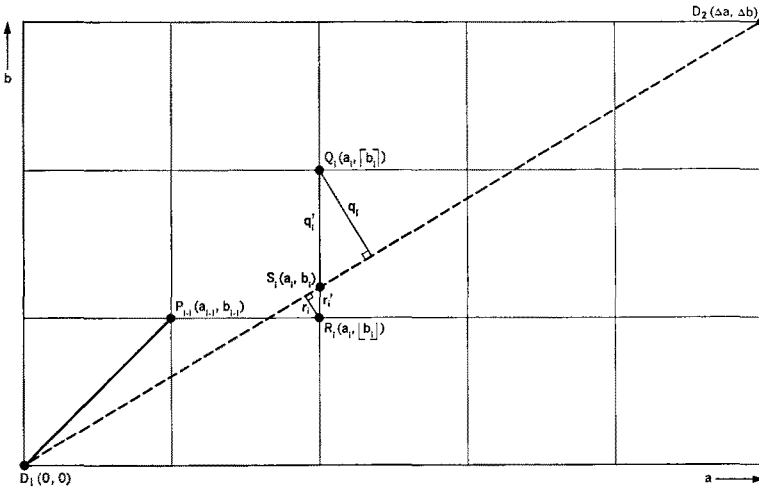
By noting that $b_i = (\Delta b / \Delta a)a_i$,

$$\nabla_i = 2a_i\Delta b - (\lfloor b_i \rfloor + \lceil b_i \rceil)\Delta a.$$

Since the line segment D_1D_2 lies in the first octant, $a_i = a_{i-1} + 1$. By definition, $\lceil b_i \rceil = \hat{b}_{i-1} + 1$ and $\lfloor b_i \rfloor = \hat{b}_{i-1}$. These relations are used to rewrite the latter expression for ∇_i in a form free of a_i and b_i :

$$\nabla_i = 2a_{i-1}\Delta b - 2\hat{b}_{i-1}\Delta a + 2\Delta b - \Delta a.$$

Figure 4 Notation for the algorithm



Applying the initial condition for the coordinates of P_0 , $a_0 = 0$ and $\hat{b}_0 = 0$,

$$\nabla_1 = 2\Delta b - \Delta a.$$

If $\nabla_i \geq 0$,

$$\hat{b}_i = \hat{b}_{i-1} + 1,$$

so that

$$\begin{aligned}\nabla_{i+1} &= 2(a_{i-1} + 1)\Delta b - 2(\hat{b}_{i-1} + 1)\Delta a + 2\Delta b - \Delta a \\ &= \nabla_i + 2\Delta b - 2\Delta a.\end{aligned}$$

If $\nabla_i < 0$,

$$\hat{b}_i = \hat{b}_{i-1},$$

so that

$$\begin{aligned}\nabla_{i+1} &= 2(a_{i-1} + 1)\Delta b - 2\hat{b}_i\Delta a + 2\Delta b - \Delta a \\ &= \nabla_i + 2\Delta b.\end{aligned}$$

Thus (1) has been shown to hold.

other
octants

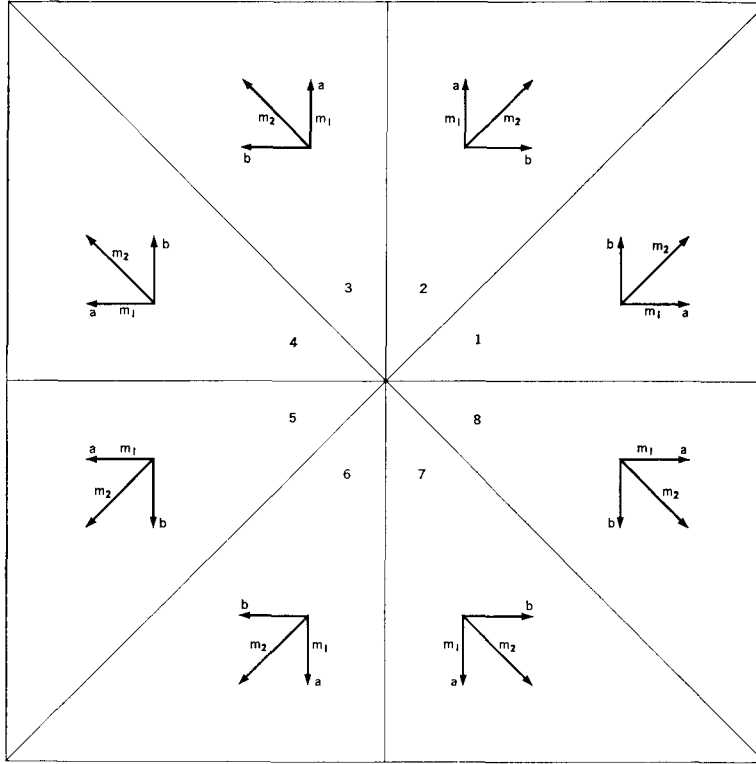
The second data point has been assumed to be in the first octant with respect to the first data point. If the second data point lies in another octant, an (a, b) rectangular coordinate system is again chosen with origin at the first data point, but with the axes oriented individually for each octant, as shown in Figure 5. Directions associated with the plotter movements m_1 and m_2 are also indicated. This information is summarized in the left columns of Table 1 together with the assignments made to m_1 and m_2 . Thus, the variants of Equations (1) and (4) have been specified for each of the eight octants, and the reader may verify that, in conjunction with (2) and (3) as previously stated, they comprise a correct formulation for the general case.

To complete the algorithm, a computational procedure is needed to determine the applicable octant for an arbitrary pair of data points so that the appropriate forms of (2) and (4) can be determined. The form of (2) depends on the sign of $|\Delta x| - |\Delta y|$.

Table 1 Determination of form of Equations 2 and 4

Δx	Δy	$ \Delta x - \Delta y $	OCT	Δa	Δb	X	Y	Z	m_1	F	m_2	G
≥ 0	≥ 0	≥ 0	1	$ \Delta x $	$ \Delta y $	1	1	1	M_1	(1, 0, 0, 0)	M_2	(1, 0, 0, 0)
≥ 0	≥ 0	< 0	2	$ \Delta y $	$ \Delta x $	1	1	0	M_3	(0, 1, 0, 0)	M_2	(1, 0, 0, 0)
≥ 0	< 0	≥ 0	8	$ \Delta x $	$ \Delta y $	1	0	1	M_1	(1, 0, 0, 0)	M_8	(0, 0, 0, 1)
≥ 0	< 0	< 0	7	$ \Delta y $	$ \Delta x $	1	0	0	M_7	(0, 0, 0, 1)	M_8	(0, 0, 0, 1)
< 0	≥ 0	≥ 0	4	$ \Delta x $	$ \Delta y $	0	1	1	M_5	(0, 0, 1, 0)	M_4	(0, 1, 0, 0)
< 0	≥ 0	< 0	3	$ \Delta y $	$ \Delta x $	0	1	0	M_3	(0, 1, 0, 0)	M_4	(0, 1, 0, 0)
< 0	< 0	≥ 0	5	$ \Delta x $	$ \Delta y $	0	0	1	M_5	(0, 0, 1, 0)	M_6	(0, 0, 1, 0)
< 0	< 0	< 0	6	$ \Delta y $	$ \Delta x $	0	0	0	M_7	(0, 0, 0, 1)	M_6	(0, 0, 1, 0)

Figure 5 Axes orientation



To find the form of (4), Boolean variables X , Y , and Z , corresponding to Δx , Δy , and $|\Delta x| - |\Delta y|$, are introduced. As shown in Table 1, these variables assume the value 0 or 1, depending on whether or not the correspondent is negative. To determine the assignment of m_1 , the function

$$F(X, Y, Z) = (XZ, Y\bar{Z}, \bar{X}Z, \bar{Y}\bar{Z}),$$

found by inspection of Table 1, is introduced. Correspondence between values assumed by F and the assignment of m_1 is indicated by columns headed F and m_1 . Similarly,

$$G(X, Y) = (XY, \bar{X}Y, X\bar{Y}, \bar{X}\bar{Y})$$

is used in conjunction with the G - and m_2 -columns of Table 1 to make the appropriate assignment to m_2 .

The algorithm can be programmed without the use of multiplication or division. It was found that 333 core locations were sufficient for an IBM 1401 program (used to control an IBM 1627). The average computation time between successive incrementations was approximately 1.5 milliseconds.

A functionally similar algorithm reported in the literature³ is described as requiring 513 core positions and 2.4 milliseconds between successive incrementations.⁴

concluding
remarks

ACKNOWLEDGMENT

The suggestions of the author's colleagues, D. Clark and A. Hoffman, were of considerable assistance.

CITED REFERENCES AND FOOTNOTES

1. This paper is based on "An incremental algorithm for digital plotting," presented by the author at the ACM National Conference at Denver, Colorado, on August 30, 1963.
2. The *floor* ($\lfloor _ \rfloor$) and *ceiling* ($\lceil _ \rceil$) operators are defined as follows: $\lfloor x \rfloor$ denotes the greatest integer not exceeding x , and $\lceil x \rceil$ denotes the smallest integer not exceeded by x . This notation was introduced by Iverson; see, for example: K. E. Iverson, "Programming notation in systems design," *IBM Systems Journal* **2**, 117 (1963).
3. F. G. Stockton, Algorithm 162, XMOVE PLOTTING, *Communications of the ACM* **6**, Number 4, April 1963. Certification appears in **6**, Number 5, May 1963.
4. F. G. Stockton, *Plotting of Computer Output*, Bulletin No. 139, California Computer Products, May 1963.