



Title	ラインプリンターとデジタル・グラフィックス
Author(s)	吉田, 勝行
Citation	大阪大学大型計算機センターニュース. 1973, 10, p. 53-70
Version Type	VoR
URL	<a href="https://hdl.handle.net/11094/65191">https://hdl.handle.net/11094/65191</a>
rights	
Note	

*The University of Osaka Institutional Knowledge Archive : OUKA*

<https://ir.library.osaka-u.ac.jp/>

The University of Osaka

## ラインプリンターとデジタル・グラフィックス

大阪大学教養部図学教室

吉 田 勝 行

その昔、不老不死の秘法を教える男がいるという話を聞いて、その国の王様は家来を弟子入りさせ、その秘法を習わせたのですが、家来がまだ習いおえないでいるうちに、その仙人を自称する男は死んでしまいました。

王様がおこって家来に言うには、

『おまえがぐずぐずしていたために、せっかくの秘法が手に入らなくなってしまったではないか。』

そしてその家来を、罰として死刑にしてしまったのだそうです。

一番大切な自分自身の体を不老不死に出来ないでいて、他人である王様を不老不死にするような方法など、あろうはずもないと思われるにもかかわらず、そんないいかげんなうたう文句の存在を信じて大切な家来を殺してしまうなど、あさはかのきわみですが、『設計図の作成や図形処理に使える。』がうたい文句になっているデジタルプロッターやC R Tを実際に設計を行う過程等で使ってみて、『不便だ。使いものにならない。』等とこぼしている人は、まずこれらの機械を作り出した会社なりメーカーが、自社の製品を設計する過程等でそれらをちゃんと使いこなしているかどうかを調べてごらんになるべきで、それをしないでいてぐちだけをこぼしているというのでは、この王様を笑う資格はないようです。

C R Tやプロッターは、精度の良い図を数枚かいてながめるのには確かに適しているのですが、設計過程や論文をまとめる過程での図の使われ方というものを考えてみますと、最終段階で図の清書をやらせる以外に、これら機械の適性を生かす道はないように思えます。

たとえば、ある条件のもとで可能な住宅の平面図（ラインプラン）を電算機に作成させますと、たちどころに数百種類位作ることが出来ますが、この中から設計を進めて行くのに適当なラインプランを選ぶについて、図-1のようにプリントアウトしたのでは、これをもとに図を手がきでもしないかぎり検討のしようもありませんし、さりとてデジタルプロッターで数百種類もの図をえがくのは、時間の点でむりがあります。

また論文をまとめる場合は、調査や実験の結果から大量のグラフを作成し、それらをながめて考察をすすめるわけですが、C R Tで1枚ずつグラフをえがき出してはながめるというより、なんらかの方法で紙の上にプリントされたグラフを同時にずらりと並べてながめた方が、はるかに能率よく結論を導びくことが出来るにちがいません。

したがって、このように図として概略がつかめればよく、精度はそれほど要求されないが、数が多いという場合には、C R Tやデジタルプロッターは、適当な装置ではありません。む

しろ日頃つかいながっているラインプリンターが、この目的にも適切であるようです。図-2は、図-1の数値を図化してラインプリンターでプリントアウトしたもので、図-1は、折れ線グラフをラインプリンターでプリントアウトしたものです。いずれの図にも、図の内容がもつ特徴がよく表われており、検討用の図としては、これで十分のようです。

SQR ( 1 ) = .0	SQR ( 27 ) = 2.9999999999
SQR ( 2 ) = .0	SQR ( 28 ) = 2.0000000000
SQR ( 3 ) = .0	SQR ( 29 ) = 1.5000000000
SQR ( 4 ) = 1.5000000000	SQR ( 30 ) = 1.5000000000
SQR ( 5 ) = 2.0000000000	SQR ( 31 ) = 1.0000000000
SQR ( 6 ) = 2.9999999999	SQR ( 32 ) = 2.9999999999
SQR ( 7 ) = 2.9999999999	SQR ( 33 ) = 2.9999999999
SQR ( 14 ) = .0	SQR ( 40 ) = 2.9999999999
SQR ( 15 ) = 2.9999999999	SQR ( 41 ) = 2.9999999999
SQR ( 16 ) = 5.9999999998	SQR ( 42 ) = 2.0000000000
SQR ( 17 ) = 5.9999999998	SQR ( 43 ) = 2.0000000000
SQR ( 18 ) = 2.9999999999	SQR ( 44 ) = 2.9999999999
SQR ( 19 ) = .0	SQR ( 45 ) = 4.0000000000
SQR ( 20 ) = 4.0000000000	SQR ( 46 ) = 4.0000000000

図-1 住宅のラインプランを数値でプリントアウトした結果の一例

HEYASU= 7 ( 4\* 4)

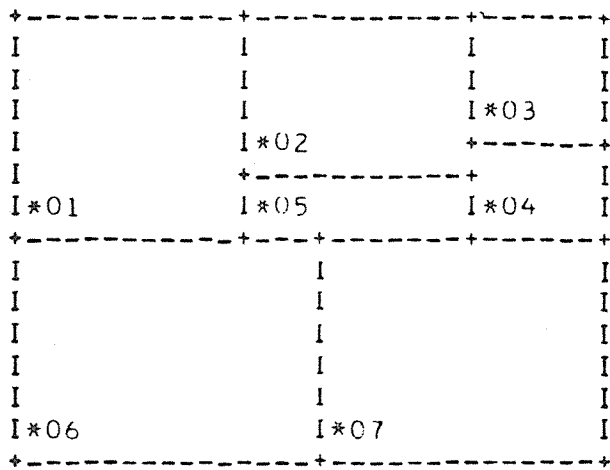


図-2 住宅のラインプランの一例（図-1の数値を図化したもの）

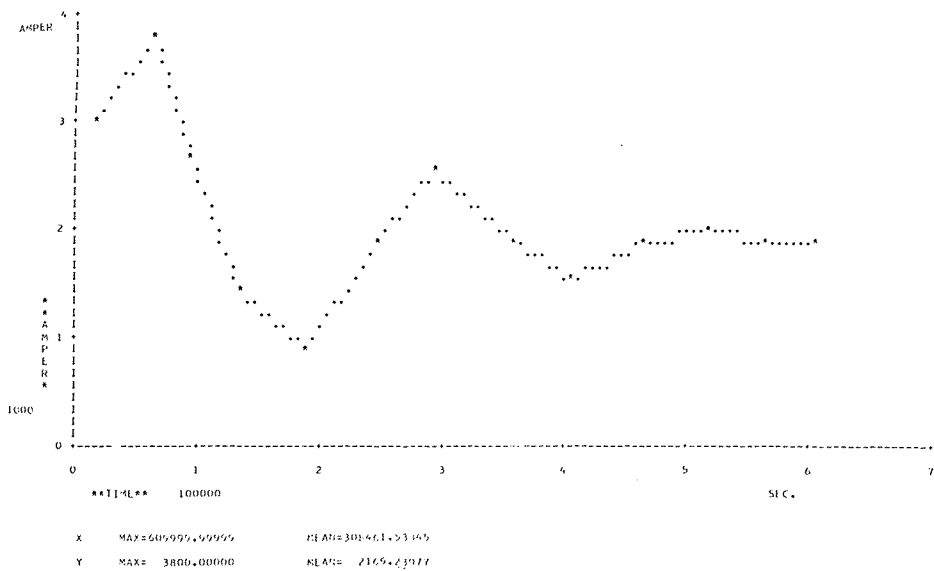


図-3 ラインプリンターでプリントアウトした折れ線グラフの一例

そこで、図形をアウトプットする際の端末装置として、ラインプリンターをお使いになることを、おすすめしたいと思います。

なおここで、タイトルにのべたデジタルグラフィックスというのは、デジタルなとびとびの要素で構成されている図形を、デジタルな手段で画く場合に必要な概念や手法をとりあつかうもので、ラインプリンターを用いて図を画く手法を開発する上で、基礎となる概念や手法等が、こういった方面から導入することが出来るのではないかと考えております。

ここでは、ラインプリンターで図をえがく際に、どのような問題があるかについて、ごく基礎的な部分を述べることにいたします。

## 1. ラインプリンターによる図の書き方の基礎

ラインプリンターのプリント用紙1頁には、縦60個、横133個までの活字を打つことが出来るようです。しかしながらこれを全部画面として使用した場合、プログラムミスでわずかに画面限界を越えた場合でも、エラーとして処理が中断されてしまう可能性があるのです。私は図-4のように縦56個、横130個までの範囲を画面として常用することにしております。

活字の大きさは、図-4に示すように、縦が1/6インチ、横が1/10インチです。したがって、画面の左上隅を原点とし、図のように原点と原点から縦1インチ、横1インチの位置に各々\*印をプリントするためのプログラムは、次のようになります。

```
STRING GAMEN (56,130)
```

```
D0 50 I=1 ,56
```

```
D0 50 J=1 ,130
```

50 GAMEN (I,J) = 1 H $\Delta$  (△印はブランク)

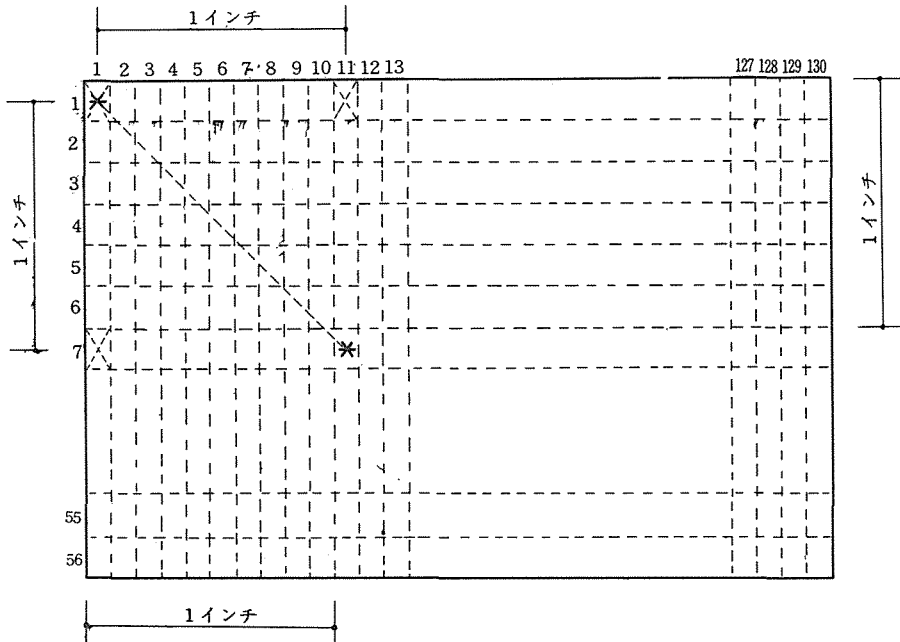
```
GAMEN ( 1 , 1 ) = 1 H*
GAMEN ( 7 , 11 ) = 1 H*
```

WRITE ( 6 , 100 ) ( ( GAMEN (I,J) , J= 1 , 130 ) , I= 1 , 56)

100 FORMAT ( 1H1 , 56 ( 130 A 1 , / , 1 H $\Delta$  ) )

STOP

END



図－4 ラインプリンターのプリント用紙上に設定出来る画面と、活字の大きさ

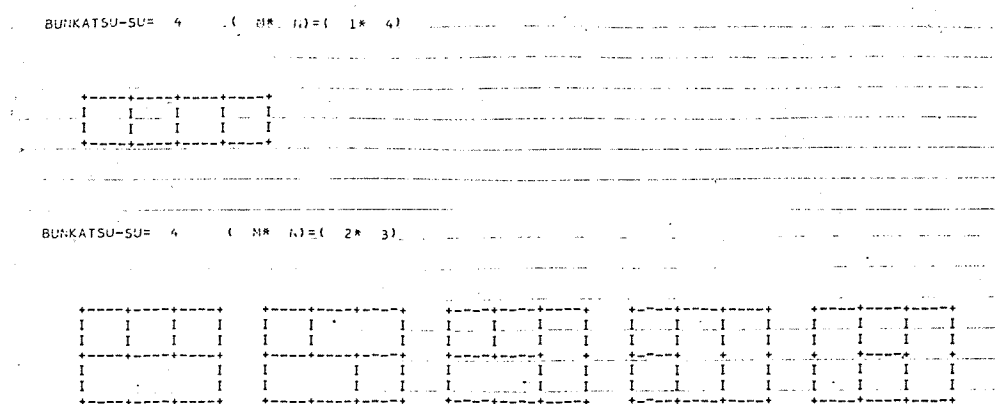
このプログラムの、2重枠でかこんだ部分に、自分の目的にあったステートメントを書きこめば、自分の求める図形が画けるわけです。

なおWRITE文をこのようにすれば、ラインプリンターは、1頁分を一挙にプリントし、プリントし終れば、それをもとにもどしてさらにその上に重ねてプリントするということが不可能です。それゆえ、例えば図－5の長方形分割図のように、1まとまりの図を横にいくつか並べてコンパクトにおさめたい場合には、計算結果をまずGAMEN内にたくわえ、1頁分がたまったところでプリントアウトさせるよう、プログラムを工夫する必要があります。

## 2. 直線を描くプログラムを組む際に誤りやすい点について

その昔、ある男が自分の奥さんに、ズボンを作ってくれるようにたのみました。奥さんがたずねていうのに、

『今度はどんなのにいたしましょうか。』



図一5 長方形分割図（4分割の場合）

『今はいているのと同じにしてくれ。』

そこで奥さんは新しいズボンをぬいあげると、それをあちこち引きさき、つぎをあててはき古したズボンと同じにしてから旦那にわたしたのだそうです。

この奥さんのゆうずうのなさにはあきれたものですが、コンピューターはプログラムを、この奥さんのように解釈しますから、プログラマたるものは、このゆうずうのなさを相手に、日頃悪戦苦闘をしているわけで、直線をラインプリンターで画くという、割合簡単なことでさえ、目的をとげることの出来るプログラムが作成出来るようになるまでに、以下に述べるような試行錯誤の段階が、誰にも必ずあるようです。

たとえば図一4において、GAMEN ( 1 , 1 ) と GAMEN ( 7 , 11 ) を直線で結ぶことを考えてみます。次にあげる2つのアルゴリズムは、どなたでもすぐに思いつかれるものでしょう。

#### I 直線からのはずれの大きさの限界値をあらかじめ設定しておく方式

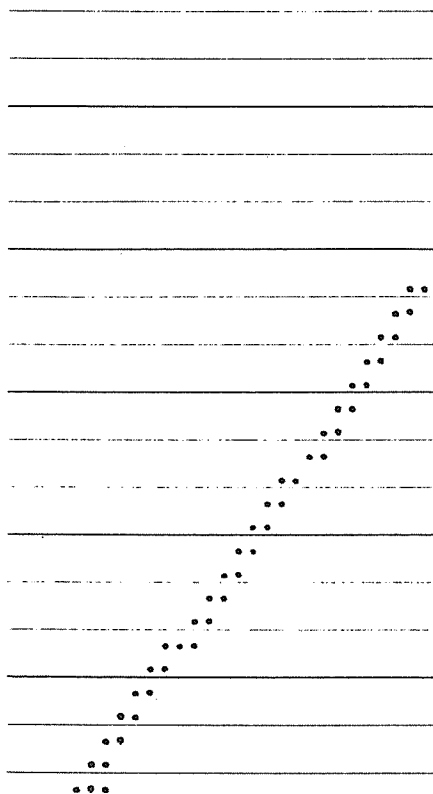
これは、GAMEN ( 1 , 1 ) ~ GAMEN ( 1 , 11 ) , GAMEN ( 2 , 1 ) ~ GAMEN ( 2 , 11 ) , ----- , GAMEN ( 7 , 1 ) ~ GAMEN ( 7 , 11 ) の各点から、GAMEN ( 1 , 1 ) より GAMEN ( 7 , 11 ) に向って引いた直線（図中では破線で表示）までの距離を各々計算し、この距離があらかじめ設定した限界値より小さい点をつらねて直線とする方式です。

#### II 縦列に必ず1個\*印をわりふる方式。

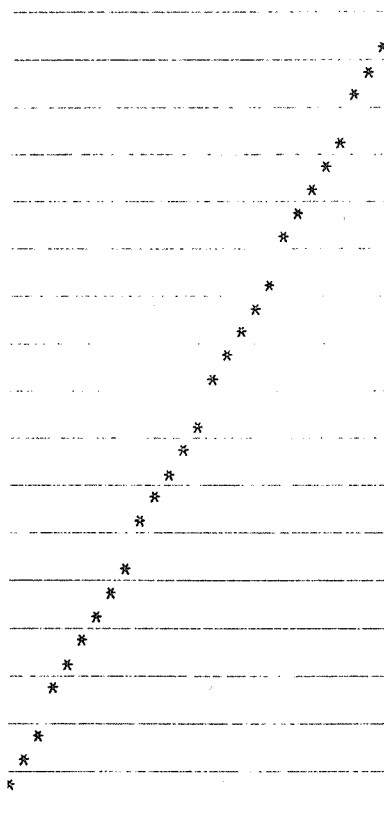
これは、任意の縦列たとえば例を4列目にとりますと、GAMEN ( 1 , 1 ) と GAMEN ( 7 , 11 ) を結ぶ直線は、GAMEN ( 1 , 4 ) , GAMEN ( 2 , 4 ) , ~ GAMEN ( 7 , 4 ) の各マス目のうち、GAMEN ( 3 , 4 ) を通ることが計算によりわかりますから、このGAMEN ( 3 , 4 ) に\*印を入れるというように考えて直線をえがく方式です。

いずれの方式も見正しように見えるのですが、実際にプログラムを組んで実行してみると、I の方式では限界値が大きいと図一6 (イ) のように線が太くなり、限界値が小さすぎると線が消えてしまっとうまく行きません。

II の方式では、直線の勾配が10 / 6 より大きくなりますと、縦列に1個\*印をわりふると



(イ) 直線からのはずれの指定が大きすぎた場合の一例



(ロ) 縦例に1個\*印をわりふった場合の一例

図—6 ラインプリンターでかいた直線の失敗例

いう条件であるために、図—6 (ロ)の例のように線がとびとびになってしまいます。もっと勾配の大きいGAMEN (7, 1) とGAMEN (1, 2) を結ぶ直線を考えてみますと、この方式ではGAMEN (7, 1) とGAMEN (1, 2) に\*印が入るだけですから、もはや直線を描いたとは言えなくなってしまいます。

こうしてみると、自分の願っていることを命令として正確に表現することは、いがいにむずかしいものです。

さきの旦那も表現が意をつくしたものでなかったために、ボロボロのズボンをはくはめになったのです。

旦那はこう命令すべきだったのでしょうか。

『寸法と形だけは、今はいているズボンと同じにしておくれ』

### 3. 直線を描くプログラム

2.で述べたことから考えて、直線を描くためのアルゴリズムとしては、次の2つの方式が考えられます。

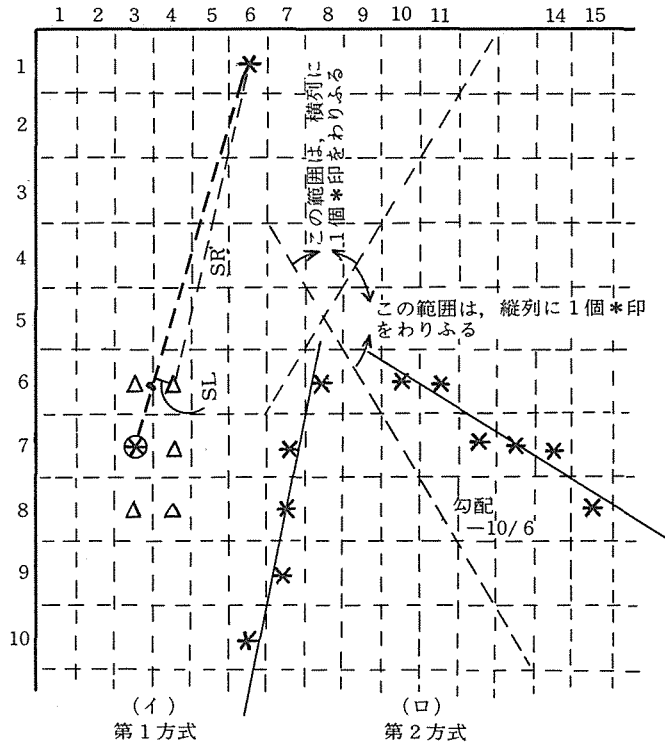


図-7 直線を描く2つの方式

# I 直線をたどってゆく方式。

これは、たとえば図-7(イ)のようにGAMEN (7, 3) からGAMEN (1, 6) までは直線で引く場合を考えますと、次のように直線の近傍のマスをたどりながら画いてゆく方式です。

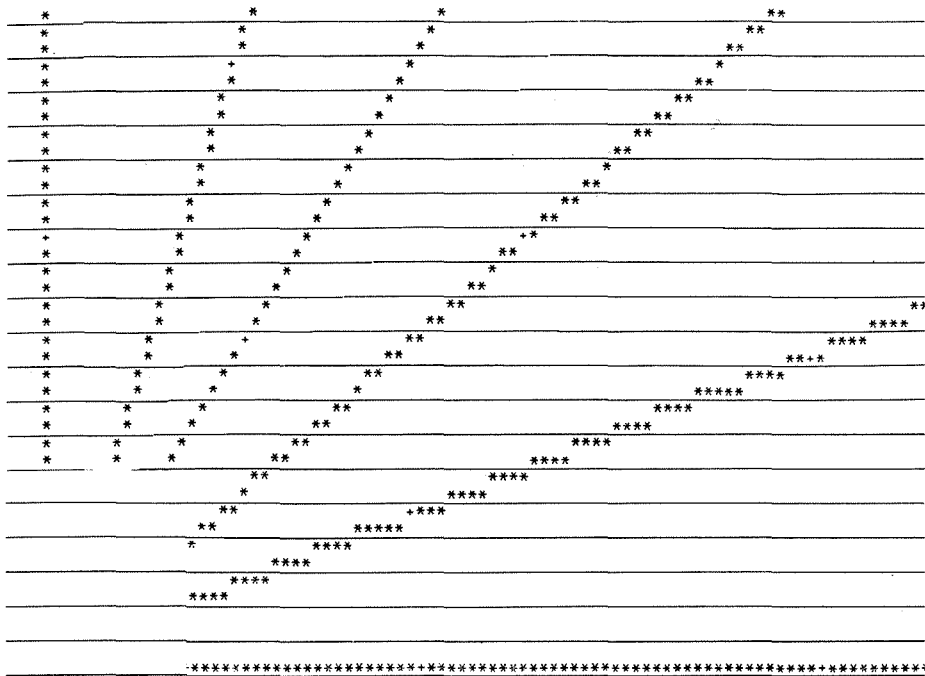
- まず始点GAMEN (7, 3) の周囲にある△印で示した5つの点について、その点から直線までの距離 $SR$ と、その点から終点GAMEN (1, 6) までの距離 $SL$ を各々計算します。
- $SL$ 最小の位置へ\*印を入れます。図-7の例であれば、GAMEN (6, 4) に\*印を入れるわけです。
- $SL$ が最小で、しかも $SR$ が最小である点にコントロールを移します。この例であれば、GAMEN (6, 4) にコントロールを移すわけです。
- コントロールが移った点GAMEN (6, 4) の周囲5つの点について、(a)のように $SR$ と $SL$ を計算します。そして $SL$ 最小であるGAMEN (5, 4) に\*印を入れ、 $SR$ 最小であるGAMEN (5, 4) にコントロールを移します。
- このように次々にコントロールを移しては、その周囲の点の $SR$ と $SL$ を計算して、\*印を入れ、次にコントロールを移す点を求めるといふぐあいにして、終点GAMEN (1, 6) に至るわけです。



## Ⅱ 縦列または横列に1個\*印をわりふる方式

これは、2.のⅡの方式を改良したもので、図一7(ロ)に示すように、あらかじめ直線の勾配を計算し、勾配の絶対値が10/6以下ならば縦列に1個\*印を割りふり、10/6より大きい場合は横列に1個\*印を割りふるというようにして、直線をえがいていく方式です。

この2つの方式のいずれを用いても、ラインプリンターにより直線が画けます。図一8にその1例を示します。



図一8 色々な勾配の直線

またアルゴリズムの面白さから考えると、Ⅰの方式の方がⅡより面白いと考えられるので、ここではⅠの方式によるプログラムを、図一9に示しておきます。まだ未整理のままですので、読みづらいと思いますが、なにかの御参考になれば幸いです。

なおこの方式はⅡの方式にくらべ、若干計算時間が長くなるようです。

## 4. 立体の透視図の書き方

立体の透視図を画く場合、画面と視点を図一10のように想定すると、立体上の1点 $P_1(x_1, y_1, z_1)$ と視点 $E(x_e, y_e, z_e)$ を結ぶ視線 $P_1E$ は、次の式で表わすことが出来ます。

$$\frac{x - x_e}{x_1 - x_e} = \frac{y - y_e}{y_1 - y_e} = \frac{z - z_e}{z_1 - z_e} \quad \text{----- (1)}$$

画面は $y = \alpha$ ですから、この画面と視線 $P_1E$ の交点 $P_p$ の座標 $(x_p, y_p, z_p)$ は、 $y_p = \alpha$ を(1)式に代入して整理することにより、次の(2)式でえられます。

MAIN PROG. : KAIJPG REV# : SOURCE PROGRAM LIST COMPILE DATE: 03/06/73(13:16'58") PAGE: 001  
MOD4EX203 FORTRAN 700 : 030 OPTIONS: DBG,AST,HOPSE,HOOVF,DLR,SRC,NOOBJ,HQMAP,HQCR5,WRRN,HOPCH

ISN	LABEL	FORTRAN STATEMENT	LINE#
0001		COMMON/G2/GAPEN(56,130)/OT/FIG	DM 10
0002		STRING GAPEN,FIG	DM 20
0003		DO 100 I=1,56	DM 30
0004		DO 100 J=1,130	DM 40
0005	100	GAPEN(I,J)=IH	DM 50
0006	200	READ(2,300)IX1,IY1,IX2,IY2,FIG,IO	DM 60
0007	300	FORMAT(4I3,A1,I3)	DM 70
0008		CALL DGLINE(IX1,IY1,IX2,IY2)	DM 80
0009		WRITE(3,400)((GAPEN(I,J),J=1,130),I=1,56)	DM 90
0010	400	FORMAT(1H1,56(130A1),/1H1)	DM 100
0011		IF(10,LE,999)GO TO 200	DM 110
0012		STOP	DM 120
0013		END	DM 130

(IX1, IY1) は始点の座標  
(IX2, IY2) は終点の座標  
FIG は \*印等の記号  
IQ は一番終りのカードの  
み 999 とする

SUBROUTINE : KENT REV# : SOURCE PROGRAM LIST COMPILE DATE: 03/06/73(13:16'58") PAGE: 001  
MOD4EX203 FORTRAN 700 : 030 OPTIONS: DBG,AST,HOPSE,HOOVF,DLR,SRC,NOOBJ,HQMAP,HQCR5,WRRN,HOPCH

ISN	LABEL	FORTRAN STATEMENT	LINE#
0001		SUBROUTINE KENT(I5,IT)	SKN 10
0002		COMMON/G2/GAPEN( 56,130)/OT/FIG	SKN 20
0003		STRING GAPEN,FIG	SKN 30
0004		IF(.NOT.(I5,GE,1.AND,15,LE, 56).AND.(IT,GE,1.AND,IT,LE,130)))	SKN 40
0005		GO TO 50	SKN 50
0006	50	GAPEN(I5,IT)=FIG	SKN 60
0007		END	SKN 70

図がプリンター用紙 1 頁に入っているかどうかを判定し、入っている部分のみを画くサブルーチン

図-9 直線を画くプログラム (その 1)

SUBROUTINE : DGLINE REV# : SOURCE PROGRAM LIST COMPILE DATE: 03/06/73(13:16'58") PAGE: 001  
MOD4EX203 FORTRAN 700 : 030 OPTIONS: DBG,AST,HOPSE,HOOVF,DLR,SPC,NOOBJ,HQMAP,HQCR5,WRRN,HOPCH

ISN	LABEL	FORTRAN STATEMENT	LINE#
0001		SUBROUTINE DGLINE(IX1,IY1,IX2,IY2)	SDG 10
0002		DIMENSION TSL(5),TSLMN(5),ITSLMN(5),TTL(5)	SDG 20
0003		SR(IYY,IXX)=FLOAT(1ABS((IYT-IY0)+IXX-(IXT-IX0)*IYY*(IXT-IY0-IX0	SDG 30
0004		1 IYI))	SDG 31
0005		SL(IYY,IXX)=SORT((FLOAT(IYT-IYY)/ 6.0)*2*(FLOAT(IXT-IXX)/10.)*2)	SDG 40
0006		DEBUG TRACE	
0007		TRACE TTL,TSL,IXSG,IYSG,TSLMN,ISGL	SDG 50
0008		IF(IX1,NE,IX2)GO TO 100	SDG 60
0009		IF(IY1,LT,IY2)GO TO 90	SDG 70
0010		IY0=IY2	SDG 80
0011		IYT=IY1	SDG 90
0012	90	IY0=IY1	SDG 100
0013		IYT=IY2	SDG 110
0014	95	ISG=IY0	SDG 120
0015	97	CALL KENT(15G,IX1)	SDG 130
0016		ISG=15G+1	SDG 140
0017		IF(15G,LE,IYT)GO TO 97	SDG 150
0018		RETURN	SDG 160
0019	C	*****	SDG 170
0020	100	CALL CLOCK(KT1)	SDG 180
0021		CONTINUE	SDG 190
0022		IF(IX1,GT,IX2)GO TO 150	SDG 200
0023		IX0=IX1	SDG 210
0024		IY0=IY1	SDG 220
0025		IXT=IX2	SDG 230
0026		IYT=IY2	SDG 240
0027		GO TO 200	SDG 250
0028	150	IX0=IX2	SDG 260
0029		IY0=IY2	SDG 270
0030		IXT=IX1	SDG 280
0031		IYT=IY1	SDG 290
0032	200	CALL KENT(IY0,IX0)	SDG 300
0033		IXSG=IX0	SDG 310
0034		IYSG=IY0	SDG 320
0035	300	TSL(1)=SR(IYSG-1,IXSG)	SDG 330
0036		TTL(1)=SL(IYSG-1,IXSG)	SDG 340
0037		TSL(2)=SR(IYSG-1,IXSG+1)	SDG 350
0038		TTL(2)=SL(IYSG-1,IXSG+1)	SDG 360
0039		TSL(3)=SR(IYSG,IXSG-1)	SDG 370
0040		TTL(3)=SL(IYSG,IXSG-1)	SDG 380
0041		TSL(4)=SR(IYSG,IXSG+1)	SDG 390
0042		TTL(4)=SL(IYSG,IXSG+1)	SDG 400
0043		TSL(5)=SR(IYSG+1,IXSG)	SDG 410
0044		TTL(5)=SL(IYSG+1,IXSG)	SDG 420
0045		JJ=1	SDG 430
0046		TSLMN(IJJ)=1000000.	SDG 440
0047		DO 400 I=1,5	SDG 450

SL と SR の計算

図-9 (その 2)

ISN	LABEL	FORTRAN STATEMENT	LINE#
0047		IF(TIL(I).GT.TILMX)GO TO 400	SDG 441
0048		IF(TSL(I).GT.TSLMR(JJ))GO TO 400	SDG 450
0049		IF(TSL(I).EQ.TSLMR(JJ))GO TO 350	SDG 460
0050		JJ=1	SDG 470
0051		TSLMR(JJ)=TSL(I)	SDG 480
0052		ITSLMR(JJ)=1	SDG 490
0053		ISGL=1	SDG 500
0054		GO TO 400	SDG 510
0055	350	CONTINUE	SDG 520
0056		JJ=JJ+1	SDG 530
0057		TSLMR(JJ)=TSL(I)	SDG 540
0058		ITSLMR(JJ)=1	SDG 550
0059		IF(TIL(I).GE.TIL(ISGL))GO TO 400	SDG 560
0060		ISGL=1	SDG 570
0061	400	CONTINUE	SDG 580
0062		DO 510 I=1,JJ	SDG 590
0063		IJKL=ITSLMR(I)	SDG 590
0064		GO TO (501,502,503,504,505),IJKL	SDG 600
0065	501	CALL KENT(IYSG-1,IXSG)	SDG 610
0066		GO TO 510	SDG 620
0067	502	CALL KENT(IYSG-1,IXSG+1)	SDG 630
0068		GO TO 510	SDG 640
0069	503	CALL KENT(IYSG,IXSG+1)	SDG 650
0070		GO TO 510	SDG 660
0071	504	CALL KENT(IYSG+1,IXSG+1)	SDG 670
0072		GO TO 510	SDG 680
0073	505	CALL KENT(IYSG+1,IXSG)	SDG 690
0074	510	CONTINUE	SDG 700
0075		GO TO (601,602,603,604,605),ISGL	SDG 710
0076	601	IYSG=IYSG-1	SDG 720
0077		GO TO 610	SDG 730
0078	602	IXSG=IXSG+1	SDG 740
0079		IYSG=IYSG-1	SDG 750
0080		GO TO 610	SDG 760
0081	603	IXSG=IXSG+1	SDG 770
0082		GO TO 610	SDG 780
0083	604	IXSG=IXSG+1	SDG 790
0084		IYSG=IYSG+1	SDG 800
0085		GO TO 610	SDG 810
0086	605	IYSG=IYSG+1	SDG 820
0087	610	IF(IYSG.EQ.IYT.AND.IXSG.EQ.IXT)GO TO 700	SDG 830
		C CALL CLOCK(KT2)	SDG 840
		C IF(KT2-KT1.LT.100)GO TO 300	SDG 850
		C WRITE(3,620)KT1,KT2	SDG 860
	620	FORMAT(1H,10HCPTIME OUT,1H,1H,1B)	SDG 870
0088		GO TO 300	SDG 880
0089	700	CALL KENT(IYT,IXT)	SDG 890
0090		RETURN	SDG 900
0091		END	SDG 900

SLとSRの最小な  
点を求めるプログラム

図-9 (その3)

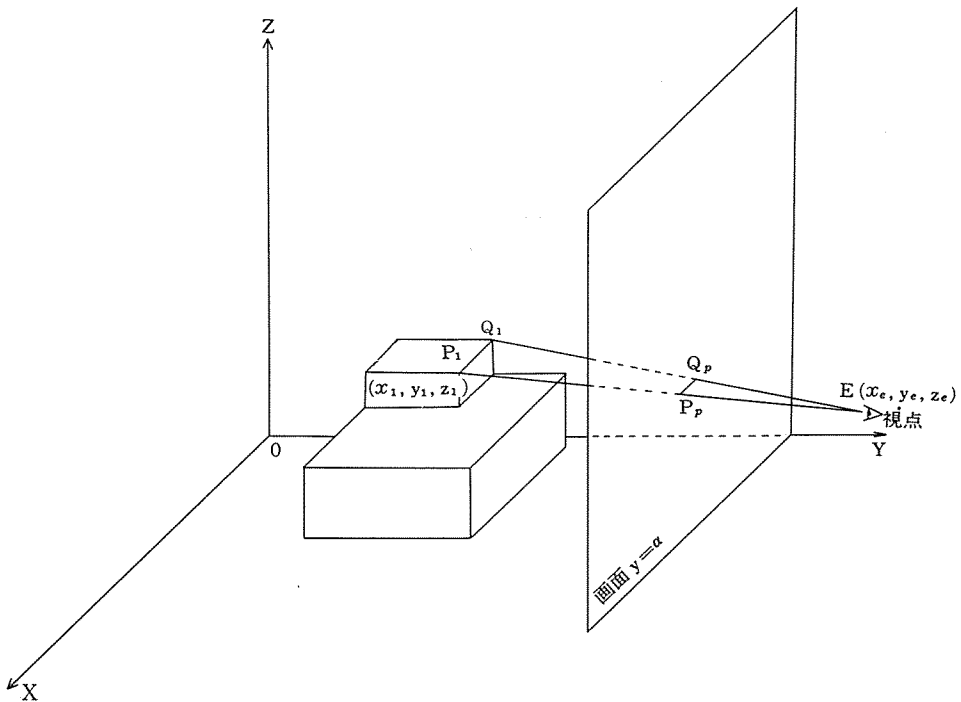


図-10 立体の透視図の考え方

$$\left. \begin{aligned} x_p &= (\alpha - y_e) * \frac{x_1 - x_e}{y_1 - y_e} + x_e \\ z_p &= (\alpha - y_e) * \frac{z_1 - z_e}{y_1 - y_e} + z_e \end{aligned} \right\} \dots\dots\dots (2)$$

したがって透視図を画くには、立体の各頂点について、(2)式より $x_p$ と $z_p$ をそれぞれ計算しておき、GAMEN (56, 130) を原点として、各頂点を $x_p$ と $z_p$ の値から整数化（この場合、 $x$ 方向は活字10個で1インチ、 $z$ 方向は活字6個で1インチという関係を使う。）してGAMENマトリックスの中にわりつけ、わりつけた頂点同志を3.の直線を書くサブルーチンで結ぶようプログラムを作成すればよいわけです。図-11は、このようにして画いた透視図の1例です。

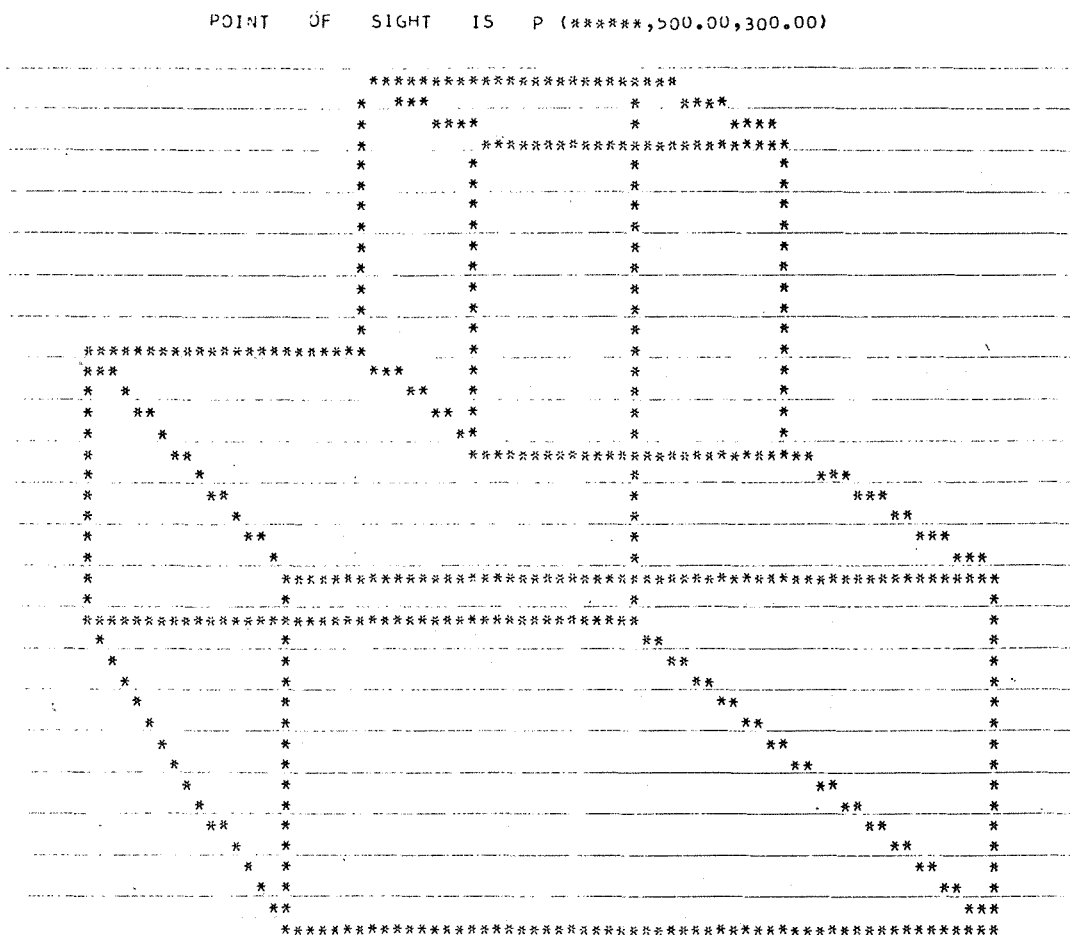


図-11 立体の透視図

なお整数化して頂点をGAMENマトリックスに割りつける時、一度GAMEN (56, 130)を原点と定めたら、それを動かさないことが重要です。なぜなら、GAMEN (56, 130)から2 cm Z 方向へ行った点はGAMEN (51, 130)になりますが、同じくGAMEN (56, 130)から1 cm Z 方向へ行き、さらにそこを新しい原点として1 cm Z 方向に行った点は、整数化の際の切り捨てが累積する為にGAMEN (52, 130)となり、さきと同じ点にならないからです。

## 5. 一般曲線の書き方

曲線を描く場合にも、2.でのべたような失敗が起りやすいことに御注意いただきたいと思います。図-12は、円の失敗例ですが、なにがまずかったかについては、もうおわかりいただけるものと思います。

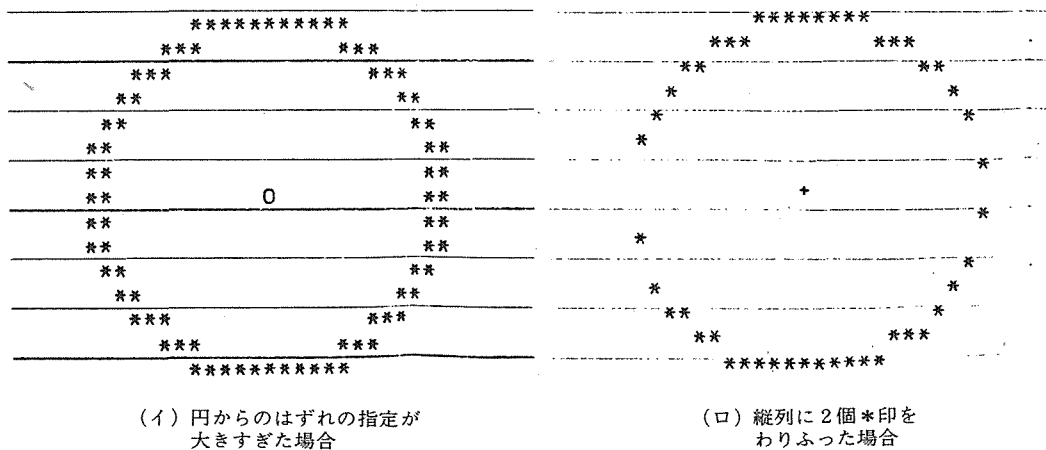


図-12 円の失敗例

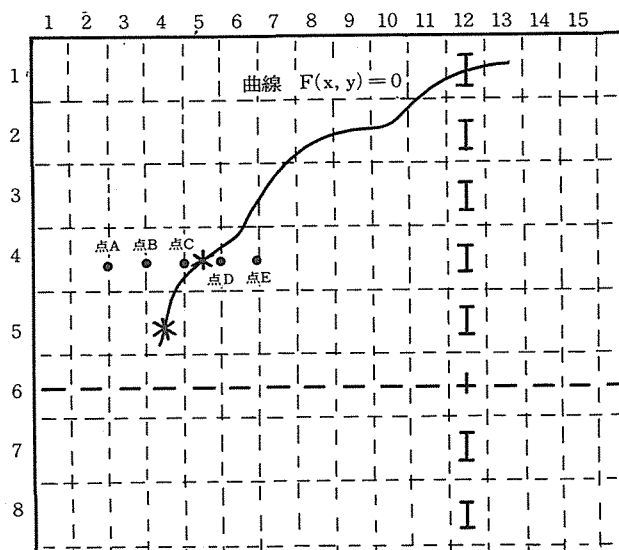
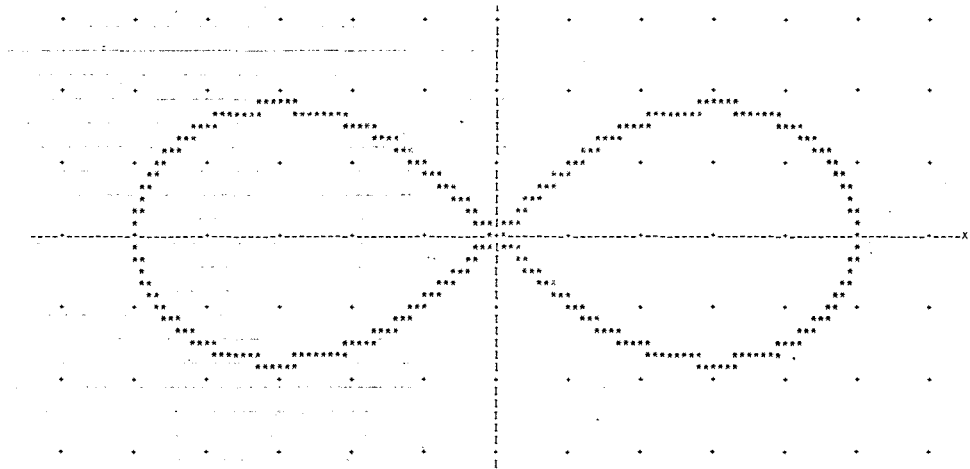


図-13 一般的な曲線の概略の書き方

$F(X, Y) = 0$  と陰関係で表わされるような一般的な曲線の概略の形を画くには、3.のIの方式と同じように線をたどって書いて行く方式以外に、図一13に示すような方式が考えられます。

これは、たとえば上から4行目を例にとると、まず原点をGAMEN (6, 12) として点A, B, C, D, E, -----等の座標を求め、この座標値を  $F(X, Y)$  に代入してその符号の正負を判定し、点Cと点Dの間で符号が変わっていることを判定して、GAMEN (4, 5) に\*印を入れるというようにして曲線を書いて行く方法です。



図一14 レムニスケート

図一14は、次のステートメント関数で表わされるレムスケートのグラフを、このような方法で画いたものです。

$$F(X, Y) = (X^{**2} + Y^{**2})^{**2} - X^{**2} + Y^{**2}Z \text{-----}(3)$$

この方式は、あくまでも全体をおおずかみにして画くために有効なのであり、細い変化をみたり、特異点の付近を細く画くのにには適していないということに御注意いただき大いと思います。

## 6. ラインプリンターによる図的表示の可能性について

以上図形をラインプリンターで画くための基礎的な事柄について述べてきましたが、ここではさらに、図的表示としてどのようなことが可能であるかについてのべてみたいと思います。

### I 群衆流動の状態表示

図一15は、群衆流動のシミュレーション結果を、図としてラインプリンターでアウトプットしたもので、人間の逃げて行く様、障壁の前のたまり具合等が1目で読みとれます。

### II マルチプリンティング

FORMAT文内に1 H+をうまく書きこんで、ラインプリンターに重ね打ちをやらせると、図をくっきりとさせたり、濃淡をつけたりすることが可能です。図一16は、記号を変えたり、

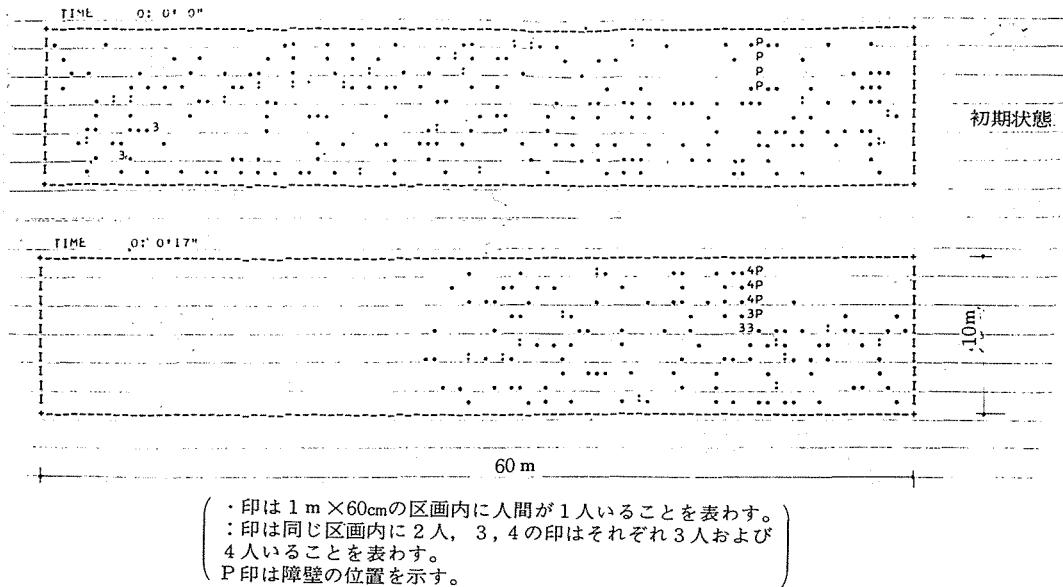


図-15 群衆流動のシミュレーション結果の表示

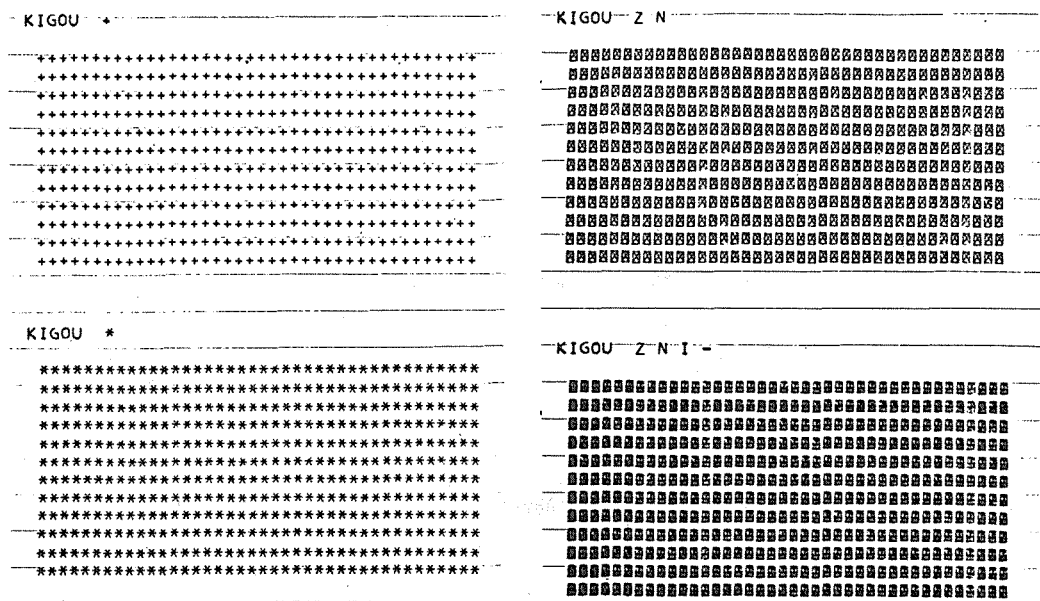
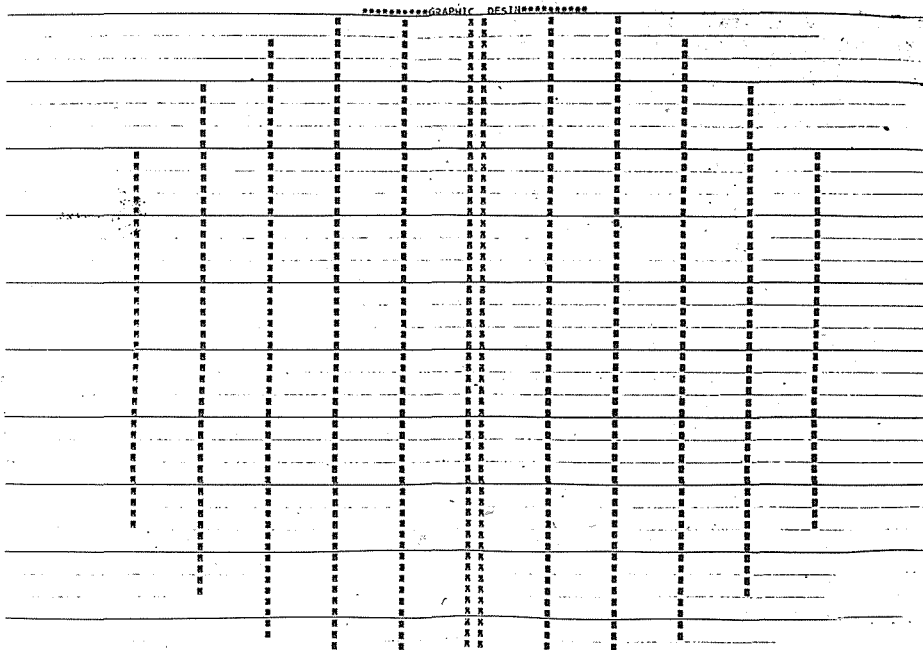


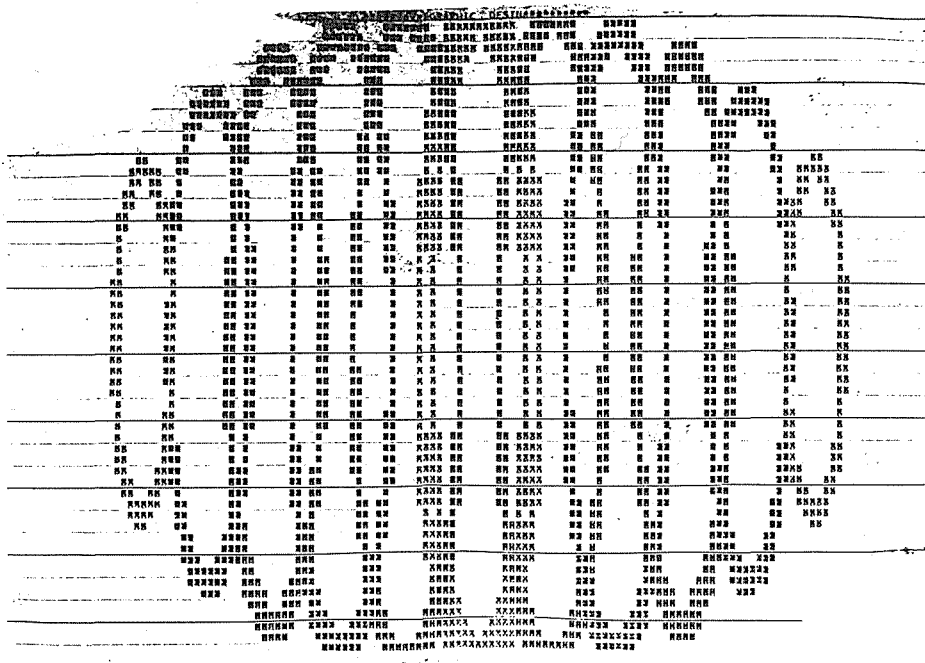
図-16 記号を変えたり、重ね打ちをすることによりえられる  
4段階の濃度

重ね打ちをすることによりえられた、4段階の濃淡です。記号の組み合わせを変えて重ね打ちをしますと、同じ回数の重ね打ちであっても、もち味が違って来ようようです。

図-17は回転楕円体を輪切りにしたものを、正面から見たり、やや斜めから見たりした結果を画いたもので、重ね打ちにより、図がはっきりとえがけています。



図一17 (イ) 回転楕円体を輪切りにしたものを、正面からながめて投象



図一17 (ロ) 回転楕円体を輪切りにしたものを、斜めからながめて投象



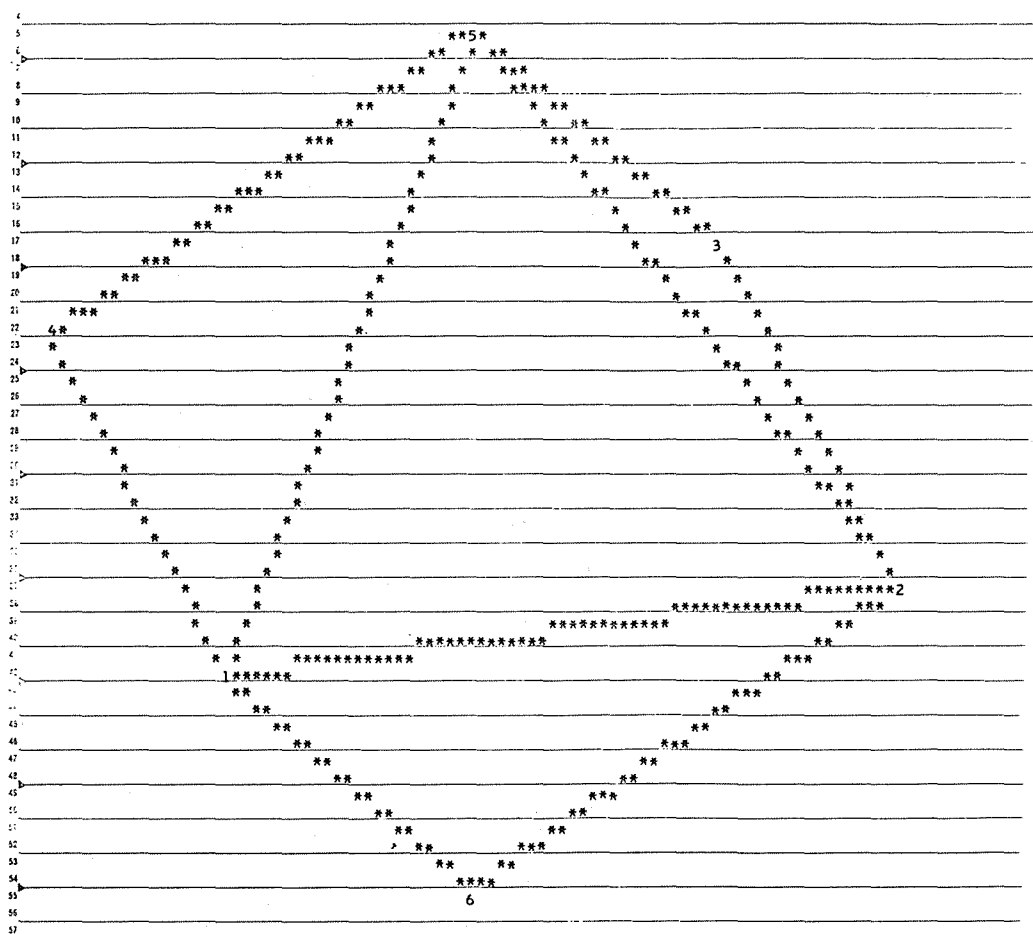


図-18 8面体の投象

図-18は8面体のみえない線を消して投象したのですが、これにすこし手を加えることにより、みえない線を十印等で薄くうち、みえる線を重ね打ちで濃く打つプログラムの開発が可能です。

### Ⅲ コンピューターアート

コンピューターを使って何か美しいと感じられるようなものが出来ないだろうかと考えていたわけですが、図-19は、まず乱数を使って点を9つ画面上に発生させ、それらを適宜直線で結んだもので、動きのあるいいコンポジションになっているようです。

図-20は、円や楕円等の曲線を、乱数を使って構成し、重ね打ちの機能をフルにを使ってプリントアウトしたもので、なかなか味のあるいい絵になっています。なお、この絵を画くについて、人間は乱数の初期値を1個定めるだけで、あとはプログラムが各種の決定を自動的にに行い、画いてゆくようにしてあります。

こうしてみると、機械に美しいものを作成させるということも、決して不可能ではないようです。

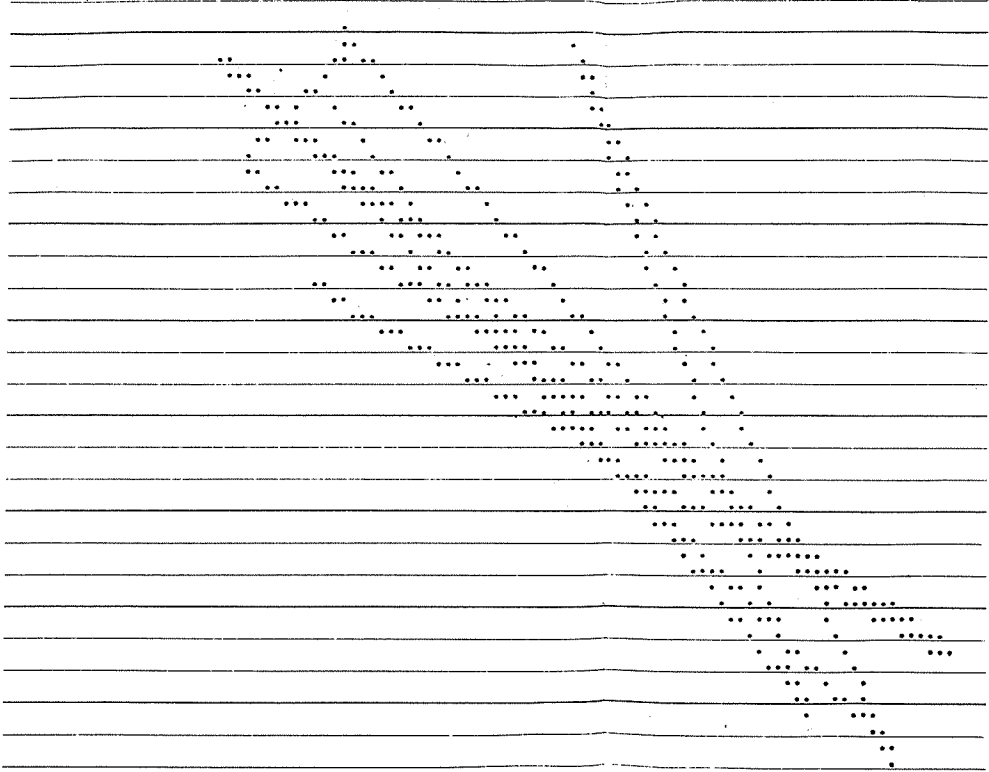


図-19 直線によるコンポジション

以上、私のところで開発した結果をもとに、ラインプリンターで図を画くための基礎的な問題や可能性について、おおざっぱにのべてみました。

ここにあげた図のおおくは、教養部で私のやっている低学年ゼミを受講された、笹耕治、百合本文夫、原正治、矢野正隆、井関健悟、田中茂、福本 晋、坂東敏博、各君らの協力により開発出来たものであります。

また、これらの図を画くについては、大阪大学大型計算機センターにお世話になりました。特にマルチプリンティングについては、色々と御めいわくをおかけしたのではないかと思います。

ここにあわせて、深甚の謝意を表したいと思います。

計算機関係の世界には、まだまだ歴史が浅いせいか、『不老不死の秘法である』という類の歌い文句が色々あるようです。ここでは、『ラインプリンターが図形処理に使える』という歌い文句がこの類のものでないことをのべてみたつもりでいるのですが、いかがお考えでしょうか。

