二変数関数の三次元表示に関する一考察

佐藤邦夫・上村鉄雄*

A Study on the Three-dimensional Representation of Functions of Two Variables

Kunio SATOU and Tetsuo KAMIMURA

1. 緒　　言

応用科学の分野において，マイクロコンピュータ及び周辺装置の普及は比較的多量のデータの取り扱いを可能とし，生データ及び処理結果の表示手段としても，XYプロッタやグラフィックディスプレイの使用を身近なものとした。

しかしコンピュータを用いても，多次元のデータを二次元平面に表示することは，かなり手間の掛かる作業であり，コンピュータグラフィクスの分野ではデータの構造，変換，表示方法などについて様様な研究が為されている[1]。特に最近では，数万色の多色表示が可能なグラフィックディスプレイを用いて三次元データを表示する，面画表示法（以降，面画法と略す）に関する研究が種々行われている。

三次元データの二次元化には，このほか比較的古くから研究されている線画表示法（以降，線画法と略す）があり，用途によって幾つかに分類される。一例として地勢の実体視や，一画面の画像データ処理に用いられるような二変数一価関数を扱ったもの[2]，また CAD (Computer Aided Design) や，CT (Computer Tomogram) で用いられるような，一般の三次元データを扱ったもの[3]などである。

しかし線画法による表示は面画法のものに比べ実体感が劣るので，今まで前者を用いていたもののうち，一部のものは後者の採用に移行しつつある。ただし一般の応用科学の分野で，パーソナルな使用をしているマイクロコンピュータにおいて，画質や処理速度で満足できる面画表示が可能となるまでには，若干の時間を必要とする情勢である。線画法は処理速度や出力装置の自由度において面画法に対して有利であり，更にデータの変数軸方向の変化を見るためには利用価値が大きいので，今後大型の計算機においても，線画法と面画法は併用されてゆくものと思われる。

筆者らは，二変数一価関数データを扱う線画表示プログラムを作成し[4]，関数の形や解の個数を調べることなどに利用している[5),6]。同じ目的を持つアルゴリズムのうち，明らかにされているものは幾つか存在するが，計算時間や所要メモリ容量，出力結果の質において満足できるものは意外に少なく，筆者らの採用したアルゴリズムは二三の点で有利であると思われる。

本報では，平行投影法を用いたものが 8 bit マイクロコンピュータの上で動作したのを機会に，仮に Zigzag 法と名付けたアルゴリズムについて，これがより所としている隠線処理の原理を中心に説明を試みる。

ただしここで扱うのは，格子状 xy 座標に対して与えられたデータ点を，網状に結ぶ線画に限るものとする。

2. 隠線の処理

三次元データの二次元化（以降，三次元表示と呼ぶ）では，視点から見えるはずのない点が表示されると，元の形状の把握が困難になることがある。三次元表示は人間の視覚に対する出力であることを考えると，これは重大な問題であり，これを避けるために隠線または隠面処理が必要となる。

線画法による三次元表示において，二変数一価関数を扱うものの隠線処理は，一般の三次元データを扱うものに比べ簡単である。しかし実際のプログラムでは，処理時間並びに所要メモリ容量を小さくするために，多種多様な方法が考えられており，これらを小さくするために

昭和58年10月15日　受理
＊名古屋大学プラズマ研究所

画質を犠牲にすることもある。

次に，明らかにされている代表的な隠線処理のアルゴリズムを挙げる。

2.1 データ点の可視，不可視を先に決定する方法[7),8),9),10)]

まず与えられた全てのデータ点について，視点とこのデータ点を結ぶ線分 l と，他の互いに隣り合う2つのデータ点 P_1, P_2 によってできる線分 P_1P_2 の上下関係を比較して可視，不可視を決める。次にその結果を基に，表1のような組合せに従ってデータ線分の投影 Q_1Q_2 の処理を行う。

表1　隠線処理一例

P_1, P_2の可視性	線分Q_1Q_2の処理
P_1, P_2は共に可視	無条件に陽線
P_1, P_2は共に不可視	無条件に隠線
P_1は可視，P_2は不可視	P_1P_2の中点を新たなデータ点として再び可視性を判断
P_1は不可視，P_2は可視	

この方法では原理的に不都合が生じることがあるが，データ点，または可視，不可視決定のための調査点を増すことで解決する必要があり，このとき処理時間も増す。また基本的には，全データ点について二次元変換後の配列を必要とする。

2.2 視点に近いデータ線から表示する方法[11),12)]

与えられるデータが，二変数一価の格子データという簡単な構造であることを利用し，視点に近いデータ線から表示を始める。作図途中において，最も外側の投影線を境界線として蓄積してゆき，新たな投影点が境界線の外側にあるとき可視，内側にあるとき不可視として処理を進め，更に境界線を蓄積する。

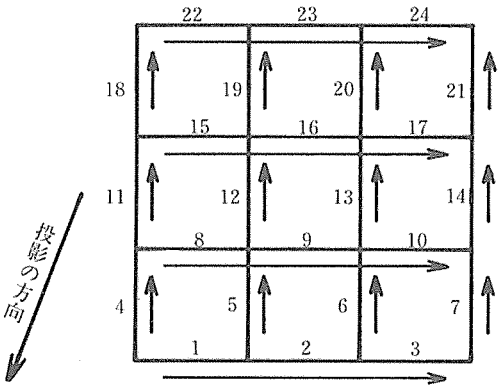

図1　処理順序例

この方法は1軸方向の表示には比較的容易に応用できるが，本報で扱うような網目状の表示に対しては境界線の取り扱いに注意する必要があり，2.1の方法に比べ，あまり用いられていないようである。

文献12)による処理・作画順序を4×4の格子点データを想定して，図1に番号で示す。

3. 平行投影による Zigzag 法

本報では三次元データから二次元画像への変換を，平行投影によって行う。

3.1 平行投影法

図2はデータ空間 π $(o\text{-}xyz)$ から投影平面 κ $(O\text{-}XY)$ への座標変換法を示す。

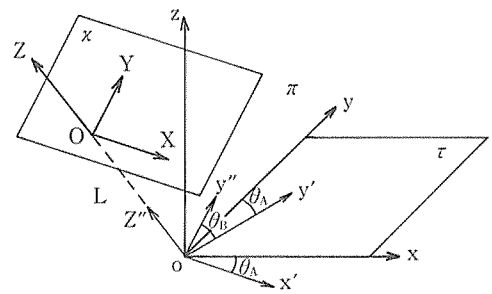

図2　座標変換

まず y 軸を z 軸の回りに x 軸方向へ θ_A 回転し，新座標系を $o\text{-}x'y'z$ とする。次に y' 軸を x' 軸の回りに z 軸の方向へ θ_B 回転し，新座標系を $o\text{-}x'y''z''$ とする。これを z'' の正の方向に L だけ平行移動し，新しくできた座標系を $O\text{-}XYZ$, 投影平面を $O\text{-}XY(\kappa)$ とする。ただし L は，全てのデータ点が Z 軸の負側にあるように十分大きくとり，また投影の方向は Z 軸に平行とする。

このとき，$o\text{-}xyz$ 座標系から $O\text{-}XY$ 座標系への変換式は次のように表わせる。

$$\left.\begin{aligned} X&=x\cos\theta_A-y\sin\theta_A \\ Y&=(x\sin\theta_A+y\cos\theta_A)\cos\theta_B+z\sin\theta_B \end{aligned}\right\} \quad \cdots(1)$$

データは格子点 (i, j) (ただし $i=1\sim i_M$, $j=1\sim j_M$) に対して1つずつ与えられる。このとき，格子点間隔 Δ_I, Δ_J が適当な実数値で設定されているものとして，図3のような格子点平面を考える。今，次のような制約

$$0<\theta_A<90° \quad \cdots(2)$$

$$0<\theta_B<180° \quad \cdots(3)$$

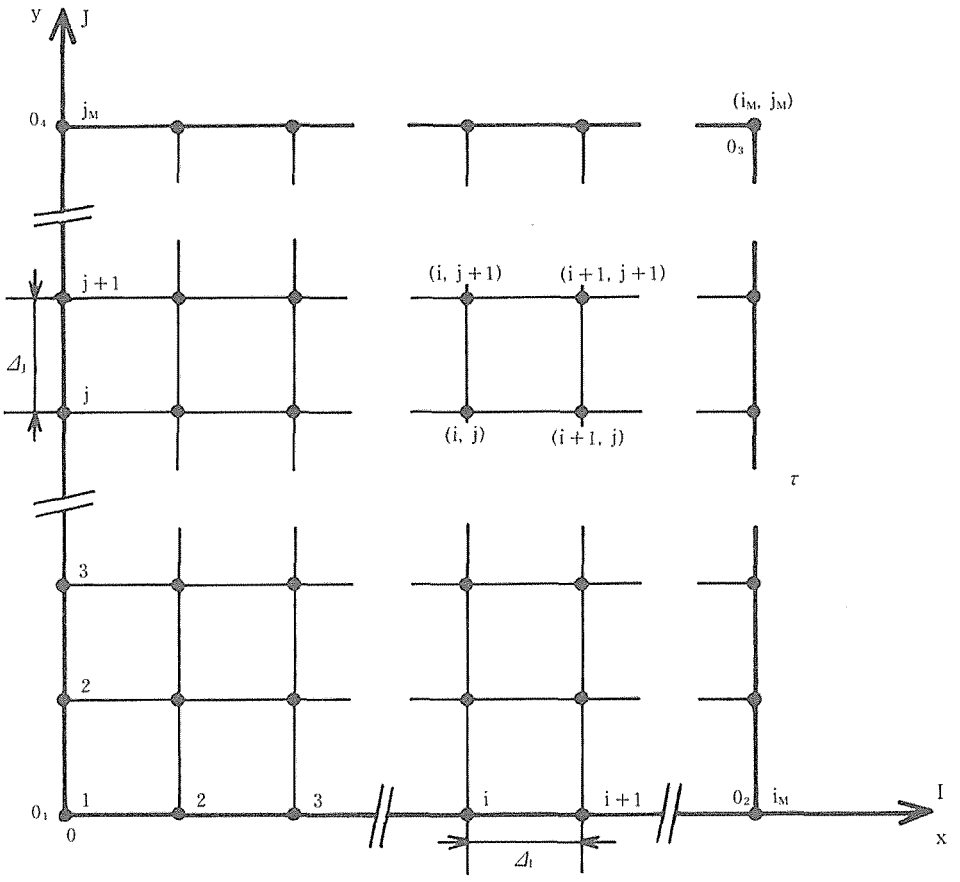

図3　格子点平面 τ

を仮定するが，作図方向の一般性を確保するためには，

1)　O_1O_2, O_1O_4

2)　O_2O_3, O_2O_1

3)　O_3O_4, O_3O_2

4)　O_4O_1, O_4O_3

をそれぞれ x 軸及び y 軸に一致させるように配置する4つの場合を考えればよい。以降簡単のため 1) の場合についてのみ考える。このときデータ点 $P_{I,J}$ の x, y, z 座標値は，$z_{I,J}$ がデータとして与えられるものとすると

$$x_I=(i-1)\Delta I,\quad y_J=(j-1)\Delta J,\quad z_{I,J} \qquad \cdots(4)$$

となり，式(1)により κ 上 $Q_{I,J}$ の X, Y 座標値に変換される。

以降，このようにして配置された格子点平面を τ と名付ける。

3.2　隠線処理と作画順序

Zigzag 法の隠線処理法は2.2で述べた方法の1つに分類される。すなわち視点の近くから先に書き始め，境界線を蓄積してゆき，隠線処理における計算手順の節約を図るのであるが，他の方法と区別されるのは処理，作画順序である（図4）。図において整数値 k と矢印はその

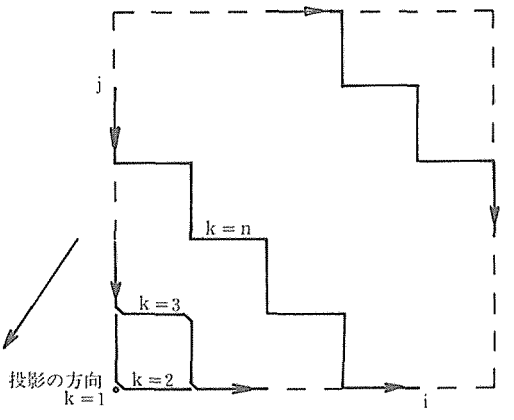

図4　Zigzag 法処理順序

順序を示すが，着目のデータ線の像が可視か不可視か判断しながらその結果を表示してゆく。

この方法では k の値の小さなデータ線群の像は，k の値の大きなデータ線群の像に干渉されることはなく，更に1つのデータ線群内に限定しても，1本のデータ線の像が他のデータ線の像に干渉されることはない，ということが前提となっている。このとき，データ線群 k の隠線処理は，1から k-1 までの処理の過程で形成された，上側及び下側境界線と着目データ線を比較しながら行う。

以下にこの原理について検討を行う。

3.3 可視と不可視（可視性）について

ここで扱うような三次元表示においては，π 上に与えられた隣接する4点はねじれ四辺形となる。一般にはこの4頂点を同時に含むような平面は存在せず，従ってそのような平面と視線の交わりを単純に論ずることもできない。議論を明確にするために，次のような定義を行う。

(i, j) ； 格子点座標
格子線 ； 隣接の2格子点によって形成される τ 上の線分
i_M ； i の最大値
j_M ； j の最大値
$P_{i,j}$ ； (i, j) に対して与えられた π 上の点，以降データ点と呼ぶ
データ線； 隣接の2データ点によって形成される，π 上の線分
P ； 任意データ線上の任意点
Q ； κ における P の像，P が決まれば式(1)により一意的に決まる。
P' ； P の τ 上への平行投影像
Q' ； Q の τ 上への平行投影像
T ； $i=1\sim i_M-1$, $j=1\sim j_M-1$ において，4データ線 $P_{i,j}P_{i+1,j}$, $P_{i+1,j}P_{i+1,j+1}$, $P_{i+1,j+1}P_{i,j+1}$, $P_{i,j+1}P_{i,j}$ によって形成される全てのねじれ四辺形を要素とする集合
t ； T の任意の要素 $t\in T$
T_p ； T の要素のうち，P を辺に含まない全てのねじれ四辺形を要素とする集合
$C(t)$ ； t の各頂点によって形成される四面体の表面及び内部の点の集合
$D(t)$ ； κ における t の像の周辺及び内部の点の集合
$E(t)$ ； t に対応する τ 上の格子点によって形成される長方形の周辺及び内部の点の集合
$q(t, P)$ ； 命題関数〔t は P を不可視とする〕
$r(P)$ ； 命題関数〔P は不可視である〕
$S_1(t, P)$ ； 命題関数〔$C(t)$ と線分 PQ は共有点を持つ〕
$S_2(t, P)$ ； 命題関数〔$D(t)$ に Q が含まれる〕
$S_3(t, P)$ ； 命題関数〔$E(t)$ と線分 $P'Q'$ は共有点を持つ〕

以上の定義の下に，更に

$$q(t, P)=(t\in T_p)\wedge S_1(t, P)\wedge S_2(t, P) \qquad \cdots(5)$$

と定義しても不都合は起こらない。このとき〔t は P を不可視としない〕ことは

$$\overline{q(t, P)}=(t\notin T_p)\vee\overline{S_1(t, P)}\vee\overline{S_2(t, P)} \qquad \cdots(6)$$

となるが，これも問題無い。従って本報では式(5)の右辺を，P と t の基本的関係における「不可視」の必要十分条件とする。またこのとき

$$r(P)=\exists t q(t, P) \qquad \cdots(7)$$

〔P は可視である〕ことは

$$\overline{r(P)}=\forall t\overline{q(t, P)} \qquad \cdots(8)$$

となる。

3.4 不可視の必要条件

一般には $q(t, P)$ のような，不可視の必要十分条件に照らして目的のデータ線の可視性を判断し，隠線処理を行えばよいが，条件の一部を他の手段で補い，計算手順を節約することができる。

ここでは $r(P)$ における，調べるべき t の数をなるべく小さく限定するため，不可視の必要条件について考察する。まず式(5)より

$$q(t, P)\longrightarrow(t\in T_p)\wedge S_1(t, P) \qquad \cdots(9)$$

次に，τ における格子点及び格子線はそれに対応するデータ点及びデータ線の平行投影像と考えることができ，

$$C(t)\xrightarrow{f}E(t) \qquad \cdots(10)$$

で示される対応 f は $C(t)$ から $E(t)$ の上への写像である。すなわち $E(t)$ のどの要素に対しても $C(t)$ の少なくとも1つの要素が対応している。

また(3)の範囲では，線分 PQ 上の点から線分 $P'Q'$ 上の点への対応は1対1の写像である。

以上により，線分 PQ が $C(t)$ と共有点を持てば，線分 $P'Q'$ は $E(t)$ と共有点を持つ。従って

$$S_1(t, P) \longrightarrow S_3(t, P) \qquad \cdots(11)$$

式(9)と合わせて

$$q(t, P) \longrightarrow (t \in T_p) \wedge S_3(t, P) \qquad \cdots(12)$$

図5は3.2で説明した手順で，格子線群 k 従ってデータ線群 k を処理している過程を示し，着目点 P はデータ線群 k 上の任意の位置にあるものとする。このとき(2)の制約の下で，線分 $P'Q'$ は範囲 (A) のどの $E(t)$ とも共有点を持たず，$S_3(t, P)$ は偽となる。

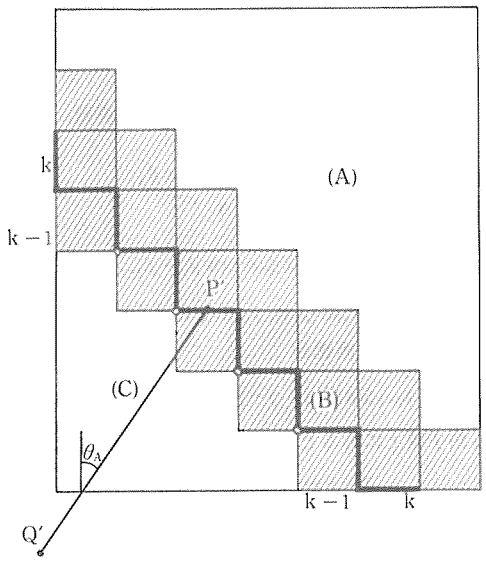

図5　格子点と視線の関係

次に範囲 (B) の $E(t)$ は点 P の位置により，点 P' を含む場合と含まない場合が考えられ，前者では $(t \in T_p)$ が，後者では $S_3(t, P)$ が偽となる。式(12)の対偶

$$(t \notin T_p) \vee \overline{S_3(t, P)} \longrightarrow \overline{q(t, P)} \qquad \cdots(13)$$

を考えると，図5の範囲 (A) 及び (B) に含まれるどの $E(t)$ に対応する t も，データ線群 k を不可視としない。

3.5 Zigzag 法における隠線処理

今までの議論により，図5においてデータ線群 k の可視性に関与するのは，番号 k-1 以下のデータ線群により形成される範囲 (C) の t であるとしてよい。更に

$$q' = (t \in T_p) \wedge S_2(t, P) \wedge S_3(t, P) \qquad \cdots(14)$$

とおくと

$$\overline{q(t, P)} \wedge q' = [(t \notin T_p) \vee \overline{S_1(t, P)} \vee \overline{S_2(t, P)}] \wedge q'$$
$$= (t \notin T_p) \wedge q' \vee \overline{S_1(t, P)} \wedge q' \vee \overline{S_2(t, P)} \wedge q' \qquad \cdots(15)$$

ここで

$$(t \notin T_p) \wedge q' = (t \notin T_p) \wedge (t \in T_p) \wedge S_2(t, P) \wedge S_3(t, P) = 0 \qquad \cdots(16)$$

q' が成り立てば直線 PQ は $C(t)$ と共有点を持つが，その τ 上への投影は $S_3(t, P)$ における共有点の集合に必ず含まれる。従って $q' \rightarrow S_1(t, P)$ が成り立ち，

$$\overline{S_1(t, P)} \wedge q' = 0 \qquad \cdots(17)$$

更に

$$\overline{S_2(t, P)} \wedge q' = (t \in T_p) \wedge S_2(t, P) \wedge \overline{S_2(t, P)} \wedge S_3(t, P) = 0 \qquad \cdots(18)$$

であるから，式(15)～(18)により

$$\overline{q(t, P)} \wedge q' = 0 \qquad \cdots(19)$$

また式(11)より

$$q(t, P) \rightarrow (t \in T_p) \wedge S_3(t, P) \wedge S_2(t, P)$$
$$= q' \qquad \cdots(20)$$

よって

$$q' \wedge q(t, P) = q(t, P) \qquad \cdots(21)$$

従って式(19)，(21)より

$$q' = q' \wedge I \qquad (I \text{ は恒真命題})$$
$$= q' \wedge [q(t, P) \vee \overline{q(t, P)}]$$
$$= q' \wedge q(t, P) \vee q' \wedge \overline{q(t, P)}$$
$$= q(t, P) \qquad \cdots(22)$$

これは，制約(2)の下では，可視性の判断において $S_1(t, P)$ を $S_3(t, P)$ に置き換えてもよいことを示し，式(7)も次のようになる。

$$r(P) = \exists t[(t \in T_p) \wedge S_2(t, P) \wedge S_3(t, P)] \qquad \cdots(23)$$

次に新たな命題関数

$S_1'(t, P)$;〔$C(t)$ と直線 PQ は共有点を持つ〕

$S_3'(t, P)$;〔$E(t)$ と直線 $P'Q'$ は共有点を持つ〕

を定義すると，明らかに

$$S_2(t, P) \longrightarrow S_1'(t, P) \qquad \cdots(24)$$

また式(11)と同様に

$$S_1'(t, P) \longrightarrow S_3'(t, P) \qquad \cdots(25)$$

従って

$$S_2(t, P) \longrightarrow S_3'(t, P) \qquad \cdots(26)$$

いま，Zigzag 法の手順でデータ線群 k ($k \geqq 4$) の処理を行っているものとし，このデータ線群上の1点 P について考える。既にデータ線群 1～k-1 については処理が終り，これまでに処理したデータ線の可視性，及び形成された t の集合 T_c は分かっているものとする。データ線群 k の処理では点 P が $t \in T_c$ なる t の頂点と一致する場合（図5の○印）が有限回発生するが，この点の可視性は既に分かっているから以下の議論はこのような点を除外して行う。

このとき，$T_c \subset T_p$ は明らかで，$t \in T_c$ なる t は ($t \in T_p$) を常に真とする。更に

$$S_2(t, P) = 1 \qquad \cdots(27)$$

を満たす t が $t \in T_c$ において1つ見つかったとすると，式(26)より

$$S_3'(t, P) = 1 \qquad \cdots(28)$$

ところが τ において，$E(t)$ と直線 $P'Q'$ の共有点は線分 $P'Q'$ に含まれるのは明らかであり，

$$S_3(t, P) = 1 \qquad \cdots(29)$$

また $S_2(t, P) = 0$ のとき明らかに

$$S_3'(t, P) = 0 \longrightarrow S_3(t, P) = 0 \qquad \cdots(30)$$

式(27)～(30)により，常に

$$S_2(t, P) \longrightarrow S_3(t, P) \qquad \cdots(31)$$

従って式(23)は $t \in T_c$ において

$$r(P) = \exists t S_2(t, P) \qquad \cdots(32)$$

すなわち Zigzag 法では，データ線群 k 上の点 P は，平面 κ 上の像 Q がデータ線群 k-1 までの処理により既に形成されている，少なくとも1つのねじれ四辺形 t の像の周辺または内部 $D(t)$ の中に含まれるとき不可視，含まれないとき可視としてよい。

3.6 平行投影法における境界線

Zigzag 法ではデータ線群 k の処理を行うとき，式(1)の x, y 座標は τ 上の格子線群 k の上に限られる。いま(2)の制約の下で，$k \leqq i_M$ かつ $k \leqq j_M$ が満たされており，x, y 座標が

$$x = x_i, \quad y_{j-1} \leqq y \leqq y_j \qquad (i = 1 \sim k-1) \qquad \cdots(33)$$

なる格子線上にあるとき（図6），

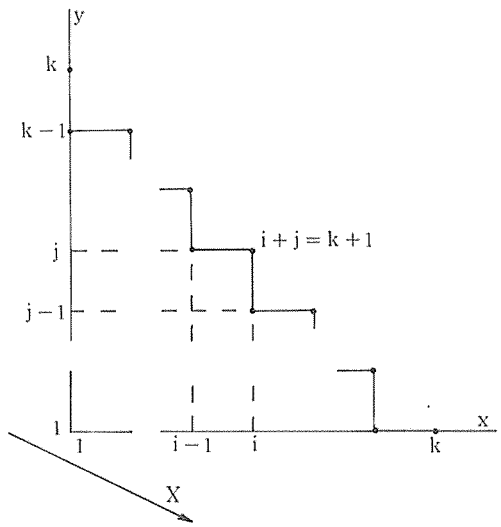

図6 平行投影法における X 座標値

$$X = x_i \cos\theta_A - y \sin\theta_A \qquad \cdots(34)$$

となり，その値は

$$x_i \cos\theta_A - y_j \sin\theta_A \leqq X \leqq x_i \cos\theta_A - y_{j-1} \sin\theta_A \qquad \cdots(35)$$

の範囲をとる。ただし x_i, y_j の値は式(4)に従うものとする。次に x, y 座標が

$$x_{i-1} \leqq x \leqq x_i, \quad y = y_j \qquad (j = 1 \sim k-1) \qquad \cdots(36)$$

なる格子線上にあるとき，

$$X = x \cos\theta_A - y_j \sin\theta_A \qquad \cdots(37)$$

となり，その値は

$$x_{i-1} \cos\theta_A - y_j \sin\theta_A \leqq X \leqq x_i \cos\theta_A - y_j \sin\theta_A \qquad \cdots(38)$$

の範囲をとり，当然ながら格子点 (x_i, y_j) において式(34)と(37)の X の値は一致する。

いま，Zigzag 法により格子点 $(1, k)$ から $(k, 1)$ に向けて処理を進めると，X の値は $x_1 \cos\theta_A - y_k \sin\theta_A$ から $x_k \cos\theta_A - y_1 \sin\theta_A$ まで単調に増加し，それぞれの X に対し Y が1つ決まる。この対応を F_k とすると $Y = F_k(X)$ は定義域

$$[x_1 \cos\theta_A - y_k \sin\theta_A,\ x_k \cos\theta_A - y_1 \sin\theta_A] \qquad \cdots(39)$$

の一価関数と見ることができる。

以上の議論は $k > j_M$ または $k > i_M$ のときも同様に行うことができ，前者では定義域の最小値が

$$x_{k-j_M} \cos\theta_A - y_{j_M} \sin\theta_A \qquad \cdots(40)$$

後者では定義域の最大値が

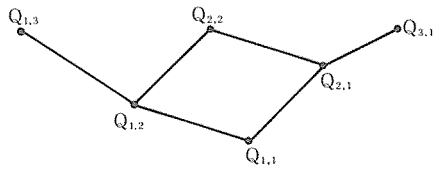

図7　データ線投影像(1)

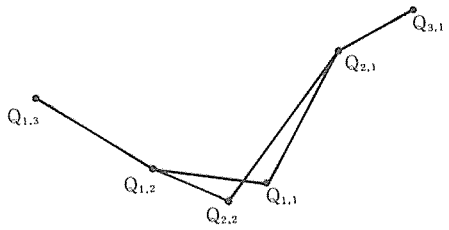

図8　データ線投影像(2)

$$x_{i_M}\cos\theta_A - y_{k-i_M}\cos\theta_A \qquad \cdots(41)$$

に変更されるが，$Y=F_k(X)$ はやはり一価関数である。

ここで $k=1\sim3$ の $Y=F_k(X)$ を全て表示すると図7～8のような形状になるが，いずれの場合もねじれ四辺形 t の像の周辺及び内部 $D(t)$ は $Q_{1,2}$ から出発して $Q_{2,1}$ で終る2つの線分群，すなわちこの範囲の全ての X に対して常に最大の Y の値を持つ線分群と常に最小の Y の値を持つ線分群によって囲まれており，この2つの線分群に挟まれている範囲の全ての点は，1つの $D(t)$ に含まれていると言える。ただし，2つの線分群に挟まれた範囲とは，線分群自身も含むとする。

Zigzag 法では，データ線群 $k=n$ を処理している過程において，それ自身 $Y=F_k(X)$ $(k=1\sim n-1)$ により組立てられ，同じ定義域の範囲で全ての X の値に対し，常に最大の Y の値を持つものを上側境界線 $Y=U_n(X)$，常に最小の Y の値を持つものを下側境界線 $Y=L_n(X)$ とする。なお定義域は，$k=1\sim n-1$ の処理において発生した全ての $D(t)$ の X 座標値の最小値から最大値までとする。このとき各境界線は定義域内で一価関数であることは明らかである。

いま $k=n$ $(n\geqq4)$ の処理を行うとき，$Y=U_n(X)$ 及び $Y=L_n(X)$ が決まっており，両境界線に挟まれた範囲の全ての点は少くとも1つの $D(t)$ $(t\in T_c)$ に含まれているとする。ここで更に $k=n+1$ の処理に進み，$Y=F_n(X)$ を考慮して両境界線を形成するとき，新たにできた全てのねじれ四辺形 t による $D(t)$ は，その頂点のうち3点までは $Y=F_{n-1}(X)$ の折点上に存在しており，これらは上側及び下側境界線に挟まれていると言える。

いま，新たにできた任意のねじれ四辺形 t の像の頂点について，$Y=F_{n-1}(X)$ の折点であるものを，X が小さい順に $Q_A(X_A, Y_A)$, $Q_B(X_B, Y_B)$, $Q_c(X_c, Y_c)$ とし，新たに $Y=F_n(X)$ によって与えられた頂点を $Q_D(X_D, Y_D)$ とすると，(2)の下では $X_A<X_D<X_c$ である。いずれの関数も一価関数であるから $X_A\leqq X\leqq X_c$ の任意の X について $Y=F_{n-1}(X)$ と $Y=F_n(X)$ はそれぞれ1つの値を持つ。

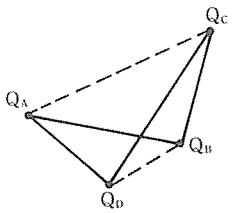

図9　データ線投影像(3)

ここで，図9の四辺形 $Q_AQ_DQ_BQ_c$ など，議論と無関係な四辺形と区別するため，ねじれ四辺形 t の像 $Q_AQ_BQ_cQ_D$ の周辺及び内部を詳細に定義すると

$$\{(X,Y)\mid(X_A\leqq X\leqq X_c)\wedge[(F_{n-1}(X)\leqq Y\leqq F_n(X))\vee(F_n(X)\leqq Y\leqq F_{n-1}(X))]\} \qquad \cdots(42)$$

のような集合で表わせる。

いま，一価関数 $Y=F_{n-1}(X)$, $Y=F_n(X)$, $Y=U_n(X)$ の全ての定義域に含まれるような $X_m\leqq X\leqq X_M$ において，

$$F_n(X_m)=U_n(X_m),\quad F_n(X_M)=U_n(X_M) \qquad \cdots(43)$$

特に $X_m<X<X_M$ において

$$F_n(X)>U_n(X) \qquad \cdots(44)$$

が成り立つとする。このとき $Y=U_n(X)$ の定義から

$$U_n(X)\geqq F_{n-1}(X) \qquad \cdots(45)$$

従って $X_m<X<X_M$ の範囲で

$$F_n(X)>U_n(X)\geqq F_{n-1}(X) \qquad \cdots(46)$$

ところが

$$F_n(X_A)=F_{n-1}(X_A)\leqq U_n(X_A) \qquad \cdots(47)$$

及び

$$F_n(X_c)=F_{n-1}(X_c)\leqq U_n(X_c) \qquad \cdots(48)$$

を考えると $X_A\leqq X_m<X_M\leqq X_c$ は明らかであり，$D(t)$ の定義より

$$F_n(X) \geqq Y \geqq F_{n-1}(X) \quad \cdots(49)$$

を満たす任意の点 (X, Y) は全て新しくできた $D(t)$ に含まれるから，式(46)より

$$F_n(X) \geqq Y \geqq U_n(X) \quad \cdots(50)$$

を満たす全ての点 (X, Y) も新しくできた $D(t)$ に含まれる。

またこのとき，$X_m < X < X_M$ における新たな上側境界線は，$Y = U_{n+1}(X) = F_n(X)$ となり，このようにして作られた上側境界線は，定義域においてやはり一価である。

また $Y = F_n(X)$ が $Y = L_n(X)$ を下側に越えるときも，同様な議論が成り立つ。

以上により，新たな2つの境界線 $Y = U_{n+1}(X)$ 及び $Y = L_{n+1}(X)$ に挟まれた範囲の全ての点は，少なくとも1つの $D(t)$ $(t \in T_c)$ により構成されている。

これを3.5の結論と共に用いると，平行投影によるZigzag法では，着目している点 P の κ 上の像 Q が，既に得られた図形の上側及び下側境界線の間にあるとき不可視，それ以外のとき可視とすることができる。

また両境界線がそれぞれ X に関して一価であることを考慮すると，Q の可視性は Q の X 座標における両境界線の Y 座標値と，Q の Y 座標値の大小比較だけで判断でき，隠線処理の計算手順を大幅に節約することができる。

4. 表示試験及び考察

4.1 表示プログラムの概要

ここでは8 bit マイクロコンピュータ用フォートランで動作させることを目標に，特に所要メモリ容量の小さなプログラムを作成した。従って境界線についても上側境界線のみを考え，表から見える部分だけ表示するようにした。これは前章で述べた，上側境界線（以降，境界線と呼ぶ）より下の点は全て不可視とするので処理速度も速くなり，小容量低速システムには向いていると言える。

次に，前章における検討の結果を実際のプログラムに応用する方法について述べる。

1) 境界線はデータ点，または境界線とデータ線の交点を X, Y 座標値で蓄積した配列であり，隣り合う2点で1本の境界線分を形成する。なお，この点を折点と呼ぶ。

2) データ線の可視性は，その像と境界線の上下関係，すなわち Y 座標の大小関係で判断する。

3) データ線群 k の処理において，$k \leqq i_M, k \leqq j_M$ のとき，最初と最後のデータ線 $P_{1,k}P_{1,k-1}$, $P_{k-1,1}P_{k,1}$ の像は必ず可視であり，$P_{1,k}$ 及び $P_{k,1}$ を次の境界線の最初及び最後の折点とする。これは前章の議論と異なるが，実際の処理には影響なく，逆に本報のように片側の境界線のみを考えるときにはこのようにしなければならない。

以下に処理，作画手順を示す。

1) 既に $k = n-1$ までの処理による境界線 $Y = U_n(X)$ は用意されており，現在 $k = n$ の処理を行っているとする。

2) 着目のデータ点を式(1)で投影変換しその像を求めたならば，境界線 $Y = U_n(X)$ から必要な線分を選択して交点の有無を調べる。

3) 交点が無いとき，その線分の可視性を判断して処理する。

4) 交点があるとき，まず交点までの部分の処理を行い，残りの部分については再び境界線から必要な線分を選び交点を調べる。

5) 処理の進行により，着目データ線群の不可視領域では既存の境界線折点を，可視領域では新たに発生した交点と可視データ点を順に作業用配列に蓄積する。

6) $k = n$ の処理が全て終了したとき，今までの作業用配列を次の処理のための境界線 $Y = U_{n+1}(X)$ とし，今までの境界線用配列は作業用配列とする。

7) $k = n+1$ の処理を開始する。

なお本プログラムでは $m \leqq n$ のとき，$m \times n$ の格子点データを処理するために

$$4m \times (1+\alpha) \quad \cdots(51)$$

の実数配列を必要とする。経験によると，α は 3.0 以下の数値で十分である。

4.2 表示例と考察

図10～16に表示例とそのデータ点数，投影平面 κ と格子点平面 τ の関係を示す。

図10は関数 $z = 1/r \cdot \sin r$, $r = \sqrt{x^2+y^2}$ を表示したものである。Zigzag 法では2つのデータ線群の折点を一致させて1つのデータ点を得るのであるが，本報のように市販 XY プロッタの高速描画モード（ペン速度 40 cm/s）を使用しても問題は生じなかった。

図11～13は比較的複雑で，なめらかな曲面の表示例である。図13は図12の1部を切り出して表示したものであ

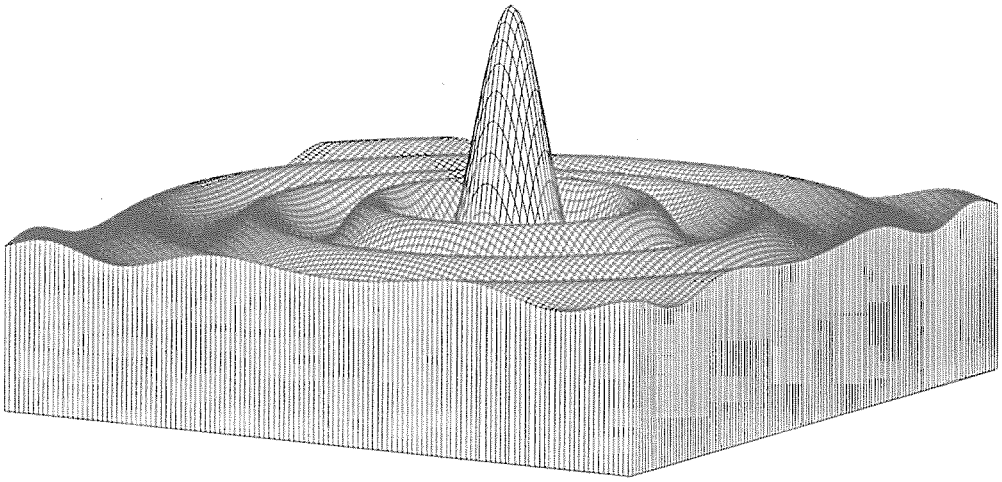

図10　表示例1
129×129,　$\theta_A=60°$,　$\theta_B=80°$

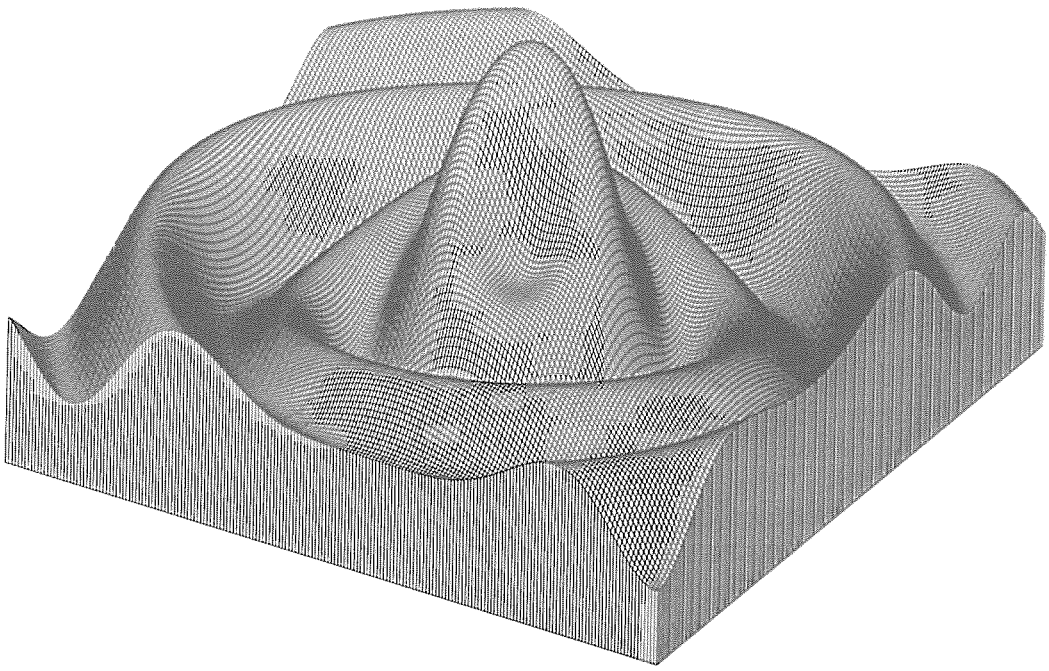

図11　表示例2
201×201,　$\theta_A=60°$,　$\theta_B=60°$

るが，このように必要な断面を見ることにより x 軸及び y 軸方向の関数の変化を知ることができる。また平行投影法では縮尺が一定であるから，フレア (flare) 部の長さを測ることによって，即座に z 軸方向の値の比較ができる。

次に図14～16は適当な間隔の孤立格子点に数値を入れて表示したもので，三次元表示プログラムに対しては苛酷な表示試験の例であるが，詳細に見ても欠落やヒゲなどの問題は発生していない。この3図は全く同一のデータを表示しているが，投影の方向によってデータの規則性の見え方に差が生じている。なおこれらの例では，正確な平行四辺形を形成している格子点平面が，目の錯覚

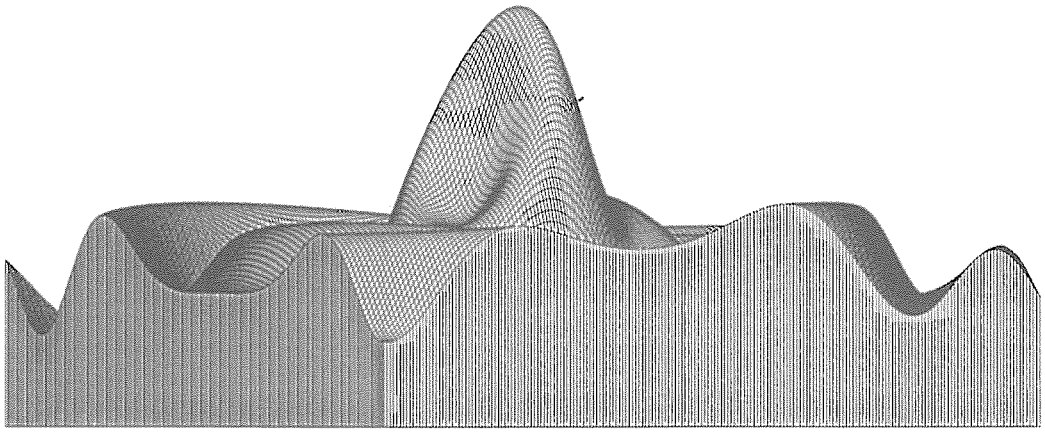

図12 表示例3
201×201, $\theta_A=30°$, $\theta_B=90°$

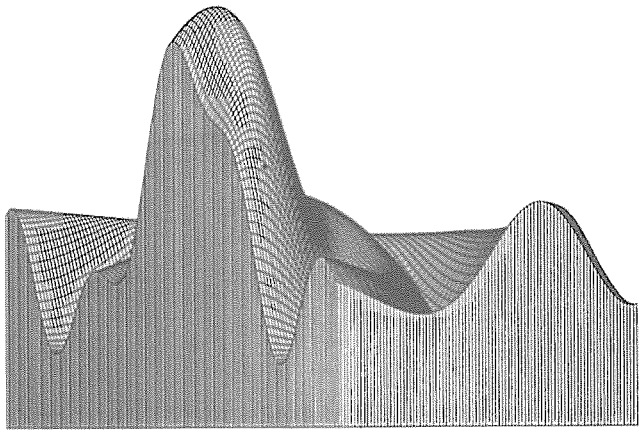

図13 表示例4
92×176, $\theta_A=30°$, $\theta_B=90°$

のため投影平面から遠い部分ほど大きく見える。これを避けるためには有心投影を用いた表示法が必要となる。

本報で作画に使用したマイクロコンピュータは，シャープ製 MZ-80C (CPU; Z-80A, 動作クロック; 4 MHz, メモリ容量; 48 KByte) であり，使用ソフトウェアはディジタル・リサーチ社製 CP/M 及びマイクロソフト社製 Fortran-80 である。このとき，隠線処理のみに要した時間はデータ点数が101×101で約15分，201×201で約60分であった。

以上のように，ここで作成したプログラムは小容量低速システムにおいても十分実用的な三次元表示が可能であることが分かった。本報のアルゴリズムでは格子点データを1度しか参照しないので，関数式や外部記憶装置からデータを入力するときには，プログラムの中に作業用配列以外の格子点データ記憶場所を設ける必要はない。データ点数 $m\times m$ の処理を基準として m が n 倍になると，概略値において Zigzag 法が n 倍の記憶容量と n^2 倍の処理時間を要するのに対し，2.1で述べた方法では n^2 倍の記憶容量と n^3 倍の処理時間を必要とする。

また2.2で述べた従来の方法に述べ，Zigzag 法は2軸方向のデータ線の取り扱いが全く同一で簡単となり，境界線の改訂回数も $m\times m$ のデータ点数に対して $2m$ 回で最も少なく，効率が良いと思われる。

このような点を考慮すると，本アルゴリズムはマイクロコンピュータ等の小容量低速システムにおける使用の

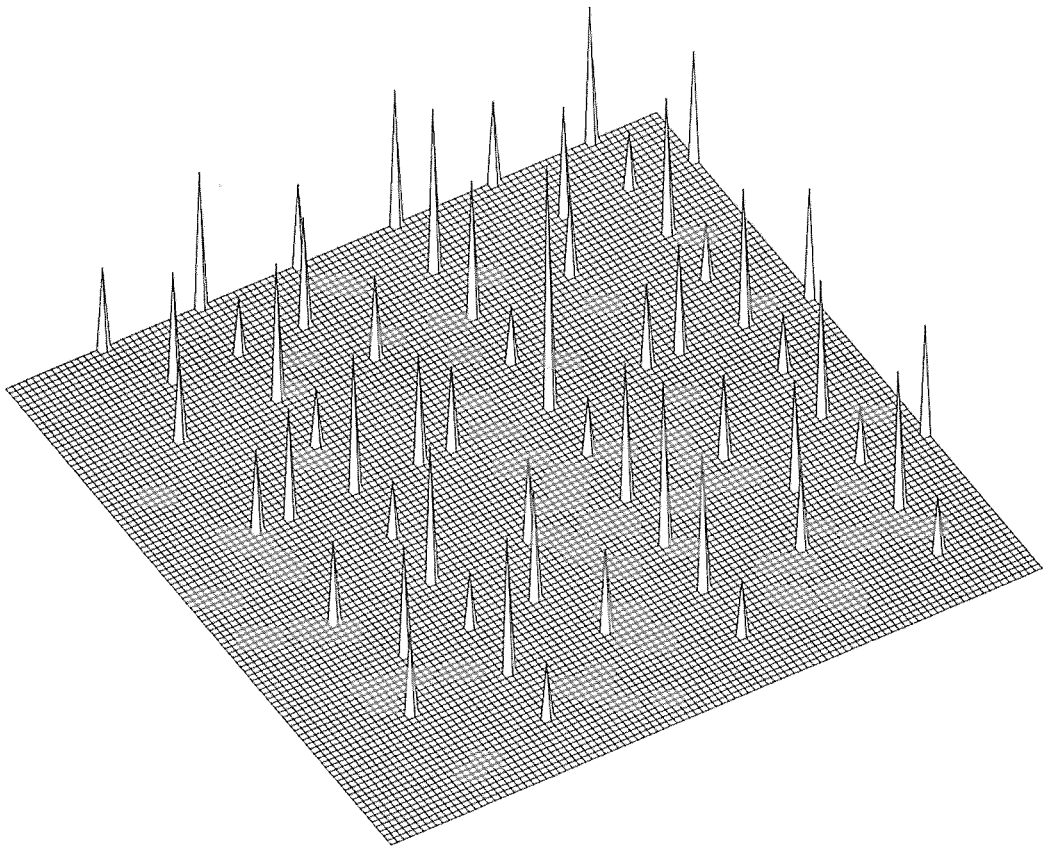

図14　表示例 5
101×101,　$\theta_A=30°$,　$\theta_B=45°$

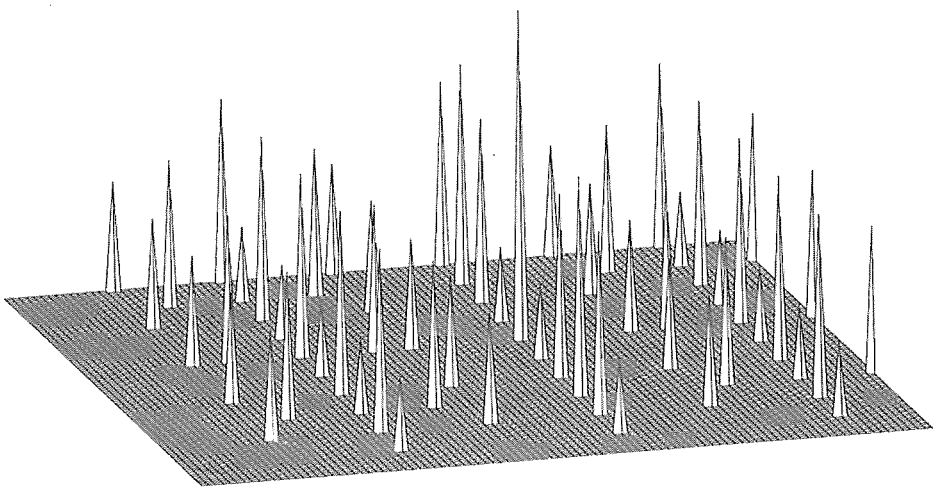

図15　表示例 6
101×101,　$\theta_A=15°$,　$\theta_B=75°$

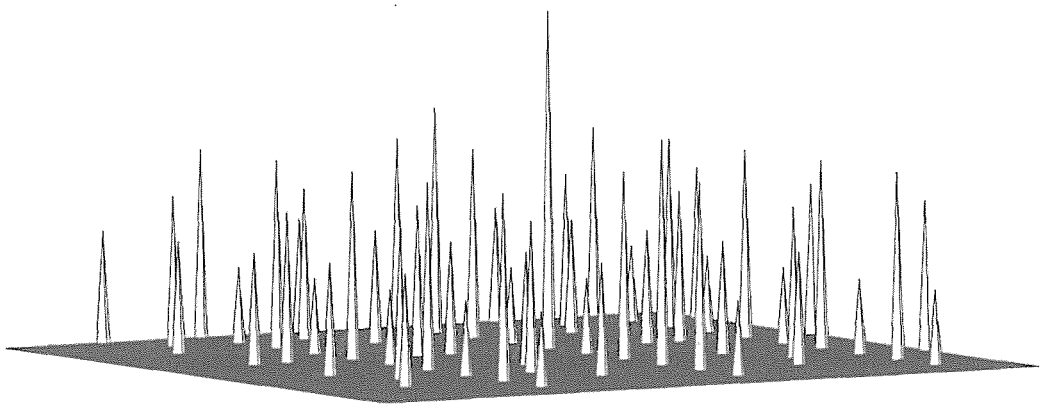

図16 表示例7
101×101, $\theta_A=30°$, $\theta_B=85°$

みならず，大型で高速な計算機システムにおいて，動的な表示をリアルタイムで行うような用途に対しても応用範囲が広がるものと思われる。

今後は有心投影を用いた表示法が必要となるが，これも Zigzag 法により実現可能であり，次の機会に考察する。

5. 摘　　要

二変数の一価関数を扱う三次元表示法について，基本的な隠線処理のアルゴリズムを構成した。この方法は与えられるデータが，二変数一価の格子点データという簡単な構造であることを利用し，視点に近いデータ線から表示する。

最も大きな特徴はその処理，表示順序にあるが，この方法の有効性を確かめるために，理論的考察を行った。その結果，本方法は計算処理においてメモリ効率，処理速度とも非常に優れたプログラムを記述しうることが分かった。

本報ではこのアルゴリズムを用いて，8ビットマイクロコンピュータ用フォートランプログラムを作成し，数例の作画を行った。この結果，処理速度，表示画質共に問題なく動作することが分かり，今まで本格的な三次元表示が行われていなかった，マイクロコンピュータなど小さなシステムにおいても，本アルゴリズムの利用が期待される。

更に大型コンピュータで本アルゴリズムの高速性を利用すれば，動的なディスプレイ表示をリアルタイムで行うような用途も生ずるものと思われる。

謝　　辞

本報で使用した XY プロッタは岩崎通信機 K.K. から貸与されたものである。ここに記して感謝の意を表します。

参 考 文 献

1) 山口富士夫：図形処理工学，東京，日刊工業新聞，1981など
2) 木平勇吉：森林の鳥瞰図，森林航測，Vol. 11 (8), 3-9, 1978など
3) 萬　淳一　他：円柱座標系データ構造による頭部CT像の三次元表示，情報処理学会論文誌，Vol. 23 (5), 516-521, 1982 など
4) 上村鉄雄：メッシュ法による三次元図形の表示，三次元図形に関する研究会報告（名大大型センター研究開発部），11-21，1973
5) 佐藤邦夫：トラクタ走行軌跡の写真計測，三重大農学報，No. 61, 319-332, 1980
6) 名大プラズマ研究所計算センターソフトパッケージ
7) B. KUBERT, J. Szabo, S. Giulieri: The Perspective Representation of Functions of Two Variables, JACM, Vol. 15 (2), 193-204, 1968
8) 秦野和郎：二変数関数の表示（その1），名大大型計算センター・ニュース，Vol. 4 (4), 288-308, 1973
9) 森　正武：曲線と曲面，東京，教育出版，1974
10) 東京大学大型計算センター：立体透視図作成用プロッタルーチン，センターニュース・ライブラリプログラム，Vol. 11 (2), 34-51, 1979
11) Hugh WILLIAMSON,: Hidden-Line Plotting Program, CACM, Vol. 15 (2), 100-103, 1972
12) 秦野和郎：二変数関数の表示（その2），名大大型計算センター・ニュース，Vol. 4 (5), 388-419, 1973

Summary

A fundamental hidden-line algorithm for the three-dimensional representation of single-valued functions of two variables was built up.

The simplicity of the given data, which consist of single-valued grid data of two variables, is utilized to represent the data-line from the near part of the view point. The most significant character of this algorithm is the sequential order of the process and display. First the availability of this method was checked by the logical study. As the result, it was ascertained that the computer programs, which are excellent in memory efficiency and processing speed, can be coded with this method.

In this paper, a FORTRAN program for the 8-bits micro-computer was developed with the above algorithm and several representations were plotted. Because there was no trouble in the processing speed and the quality of the representation, it is expected that this algorithm will be used in the small computer systems. Until now, in these small systems, as in micro-computer systems, serious three-dimensional representation was not made.