3次元ベクトル演算の並列実行に関する考察†

成 瀬 正††

3次元ベクトル，3×3マトリクスから構成される式の並列実行について集合論的観点から論ずる．また，それを実現する演算機構について論ずる．まず，3次元定数ベクトル，3×3定数マトリクスから構成される集合を考え，その上に基本演算を定義する．さらに，3次元ベクトル変数を導入し集合を拡張する．(それをGとする．)そして，集合Gがもついくつかの基本的な性質を示す．また，G上に微分演算子を導入し，微分した結果の式がGの基本演算で記述できることを示す．ついで，G上の基本演算を並列に実行する演算機構を示す．3個の演算器を用いた場合が演算器の利用率をもっとも高められるので，3演算器構成の並列実行機構を具体的に示す．Gの式をこの並列実行機構と逐次実行機構でそれぞれ演算した場合の演算回数の比として演算並列度を定義すると，G上の基本演算は演算並列度3で実行できることを示す．次に，線形代数で用いられる主な演算について，それらが基本演算を用いて記述できることを示す．これらの中にはベクトル要素の総和をとるような本質的に逐次な演算が含まれる．本質的に逐次な演算だけが演算並列度を低下させる要因であり，その部分を除けばGの式は演算並列度3で実行できる．

1. はじめに

自然界が3次元空間であることから，3次元ベクトル，3×3 マトリクス演算をベースとして記述できる問題はきわめて多い．たとえば，コンピュータグラフィックス (CG) における光線追跡法[1]，ロボットのアーム制御[2]，プラズマシミュレーション[3]，などがある．

光線追跡法は，幾何光学に基づいて光線の振舞いを忠実に計算する手法であり，きわめて写実的な画像が生成できる．そのため，アニメーションのみならず，CAD/CAM システム[5]や建築物の景観のシミュレーション[6]などへの応用がなされている．また，光線追跡法は，音場の解析[7]や，電波障害解析でも利用されている．しかしながら，これらの問題では膨大な計算量を要求することが多く，その高速演算が望まれている．たとえば，光線追跡法による CG では1枚の画像を生成するのにミニコンピュータを使って数時間〜数十時間を要する．会話型の画像生成環境を考えると，実時間で画像を生成する超高速計算機が必要となる．

ロボットのアーム制御では，数ミリ秒ごとに外部環境をセンスしアーム位置を制御する必要があり，実時間処理の要求がきびしい．粒子モデルによるプラズマシミュレーションでは，20万粒子のシミュレーション1回に大型計算機で30時間程度を要している[4]．より多数の粒子でより短時間でシミュレーションをしたいという要求がある．

光線追跡法やプラズマ問題の高速処理にはマルチプロセッサによる並列処理が有力であり，実際に，いくつかのシステムが開発されている（たとえば参考文献8), 10)). これらのシステムは演算レベルでは逐次処理を行っている．しかしながら，まだ速度的に不十分であり，より高速な処理を狙うには演算自身の並列処理を行う必要が生じてくる．そこで，これらの問題に特有な3次元ベクトル，3×3 マトリクス演算を並列処理することが考えられ，その機構を内蔵したシステムが著者らにより開発されている[11]．

このような背景から，次のような興味ある問題が生まれる．すなわち，「3次元ベクトル，3×3 マトリクスから構成される演算はどの程度並列に実行できるのか，またそれを効率よく実行する機構はどのようなものか．」という問題である．

本論文では集合論的観点からこの問題を考案する．まず，3次元定数ベクトル，3×3 定数マトリクスの集合を考え，その上に基本演算を定義する．さらに3次元ベクトル変数を導入してこの集合を拡張する．(拡張した集合をGとする．) 次いで，集合Gがどのような性質をもつかを明らかにする．そして，集合Gに属する式の並列実行を検討し，その並列実行に必要な機能を明らかにする．最後に，並列実行した場合と逐次実行した場合との演算ステップ数の比で演算並列度を定義し，ベクトル演算の並列度を論じる．

2. 定義および基本的性質

この章では，ベクトル演算を議論するために必要な

† Parallel Execution of 3 Dimensional Vector Operations by TADASHI NARUSE (NTT Human Interface Laboratories).

†† NTT ヒューマンインタフェース研究所

基本事項の定義を行い，また，そこから導かれる基本的性質について述べる．

3次元実ベクトル，3×3 実マトリクスの全体からなる集合を F とする．ここで，ベクトルは横ベクトルとする．以下では，F のベクトル要素を a, b などと書き，マトリクス要素を M, N などと書く．また，マトリクスは次のような skewed 配置になっているものと考える．ただし，m_{ij} はマトリクス要素であり，m_i はベクトルで (m_{i1}, m_{i2}, m_{i3}) を意味する．また，$\ominus$, $\oplus$ は後述するシフト演算である．

$$\begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix}$$

あるいは

$$\begin{pmatrix} m_1 \\ m_2 \\ m_3 \end{pmatrix}$$

通常配置

$$\begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{23} & m_{21} & m_{22} \\ m_{32} & m_{33} & m_{31} \end{pmatrix}$$

あるいは

$$\begin{pmatrix} m_1 \\ \ominus m_2 \\ \ominus m_3 \end{pmatrix}$$

skewed 配置

まず，F 上に基本演算を定義する．

ベクトル-ベクトル演算

① ベクトル要素間の二項演算 $\odot$

$a \odot b = (a_1 \odot b_1, a_2 \odot b_2, a_3 \odot b_3)$

$\odot$ は +，−，×，／のいずれかである．ただし，割算では，b は strictly-nonzero $(b_1, b_2, b_3 \neq 0)$ でなければならない．

② ベクトル要素の単項演算

$-a = (-a_1, -a_2, -a_3)$

③ シフト演算 $\ominus$, $\oplus$

$\ominus a = (a_3, a_1, a_2)$

$\oplus a = (a_2, a_3, a_1)$

以下の議論で分かるように，シフト演算はベクトル演算において中心的役割を果たす．

ベクトル-マトリクス演算

積 aM を考える．

マトリクス-ベクトル演算

積 Ma を考える．（厳密には，$^t(M^t a)$ と書くべきであるが，煩雑になるので本論文では Ma のように記述する．）

マトリクス-マトリクス演算

① マトリクス要素間の二項演算 $\odot$

$\odot$ は +，−，×，／のいずれかである．割算では，除数は strictly-nonzero でなければならない．

② マトリクス要素の単項演算 −

ベクトル演算と同様．

③ シフト演算 $\ominus$, $\oplus$, ①, ①

$$\ominus M = \begin{pmatrix} \ominus m_1 \\ \ominus(\ominus m_2) \\ \ominus(\oplus m_3) \end{pmatrix} \qquad ① M = \begin{pmatrix} \oplus m_2 \\ \ominus m_3 \\ m_1 \end{pmatrix}$$

である．$\oplus$, ①も同様．

④ 対角要素の抽出 diag, diag^r, diag^l

skewed 配置の場合

$\mathrm{diag}(M) = (m_{11}, m_{21}, m_{31})$

$\mathrm{diag}^r(M) = (m_{32}, m_{12}, m_{22})$

$\mathrm{diag}^l(M) = (m_{23}, m_{33}, m_{13})$

ここで，m_{ij} はマトリクス要素

なお，後の議論で必要となる通常配置のマトリクスに対する対角要素の抽出演算は次のようになる．

$\mathrm{diag}(M) = (m_{11}, m_{22}, m_{33})$

$\mathrm{diag}^r(M) = (m_{31}, m_{12}, m_{23})$

$\mathrm{diag}^l(M) = (m_{21}, m_{32}, m_{13})$

⑤ マトリクス-マトリクス積

通常のマトリクス積を考える．

このように基本演算を定義すると，集合 F に関して次の性質が成り立つ．

［性質 1］

F は基本演算に関して閉じている．

次に，ベクトル変数 x を導入する．$(x = (x_1, x_2, x_3)$ あるいは $x = (x, y, z)$ と記す．）集合 $G_0 (= F)$ の要素と x に上記の基本演算を行って作られる式を G_0 に付加した集合を $G_1[x]$ と書く☆．すなわち，

$G_1[x] = G_0 \cup \{e \odot x : e \in G_0,\ \odot$ は基本演算$\}$

次いで，G_n を

$G_n[x] = G_{n-1}[x] \cup \{e_1 \odot e_2 : e_1, e_2 \in G_{n-1}[x],$

$\odot$ は上記演算$\}$

とし，その推移的閉包 $G[x]$ を次のように定義する．

$$G[x] = \bigcup_n G_n[x]$$

$G[x]$ はベクトル x を変数とし，定数ベクトル，定数マトリクスを係数として作られるすべての多項式およびマトリクスの集合である．たとえば，$(x^2, y^2, z^2) \in G[x]$ であるが，$(x^3, y^2, z) \notin G[x]$ である．

作り方から明らかなように，

［性質 2］

$G[x]$ は基本演算に関して閉じている．

次に $G[x]$ 上に微分演算子を導入する．すなわち，$E(x) = (E_x, E_y, E_z) \in G[x]$ に対して，微分演算子を，

$$\frac{dE(x)}{dx} = \left(\frac{\partial E_x}{\partial x}, \frac{\partial E_y}{\partial y}, \frac{\partial E_z}{\partial z}\right)$$

☆ 意味のある演算だけを考える．たとえば，$a + x$ は意味があるが，$M + x$ は意味をもたない

のように定義する．（これを $E'(x)$ とも記す．）微分演算が $G[x]$ で閉じていること，すなわち，$G[x]$ の式を微分した結果の式はまた $G[x]$ の式であることを以下に示す．

最初に，ベクトルのシフト演算に関していくつかの性質を述べておく．

［性質3］ $\ominus$に関する二，三の公式（$\oplus$についても同様の関係が成立する．）

$f(x)$, $g(x)\in G[x]$, $M\in G[x]$ のとき，

1. $\ominus(f(x)\odot g(x))=(\ominus f(x))\odot(\ominus g(x))$
   ；$\odot$は二項演算
2. $\ominus(\odot f(x))=\odot(\ominus f)x))$　；$\odot$は単項演算
3. $\ominus\ominus f(x)=\ominus f(x)$
4. $\ominus(Mf(x))=(\textcircled{1}M)(\ominus f(x))$
   $\ominus(f(x)M)=(\ominus f(x))(\textcircled{1}M)$

［証明］

いずれも定義にもどって考えれば明らかである．たとえば4.の上の式は，$M=(m_{ij})$, $f(x)=(f_i)$ とすると

$$Mf(x)=(\Sigma m_{1i}f_i, \Sigma m_{2i}f_i, \Sigma m_{3i}f_i)$$
$$(\textcircled{1}M)(\ominus f(x))=(\Sigma m_{2i}f_i, \Sigma m_{3i}f_i, \Sigma m_{1i}f_i)$$

より成立することが分かる．（マトリクスは skewed 配置であることに注意）

［性質4］（シフト還元）

$\ominus f(x)$ なる形の式は中間のレベルにシフト演算を含まない式に還元できる．すなわち，還元した式ではシフト演算は $\ominus x$，$\ominus x$ なる形でのみ現れる．

［証明］

付録1に示す．

次に，微分演算に関して二，三の公式を述べる．

［性質5］

$$(f(x)\pm g(x))'=f'(x)\pm g'(x)$$
$$(f(x)\times g(x))'=f'(x)\times g(x)+f(x)\times g'(x)$$
$$(\ominus x)'=(\ominus x)'=0$$

［証明］ 定義にもどって考えれば明らか．

性質4，性質5を使うと，次の性質が証明できる．

［性質6］ $G[x]$ の式を微分した結果の式は基本演算の組合せで記述できる．すなわち，$G[x]$ は，上記微分演算に関して閉じている．

［証明］

付録2に示す．

この性質により，微分した結果の式は他の基本演算と同様並列に計算できる．

光線追跡法では，曲面とその法線ベクトルが重要な役割を果たすが，それに関して性質6から次の系が導かれる．

［系］

曲面の式が $G[x]$ に属すなら，その法線ベクトルを表す式は $G[x]$ に属す．

3. 基本演算の並列実行

前章で定義した基本演算の並列実行を考える．

演算の実行という観点からは，マトリクス-ベクトル積，マトリクス-マトリクス積を他の基本演算を用いて記述しておくのが都合よい．

マトリクス-ベクトル積　Ma

$$Z=\begin{pmatrix}z_1\\z_2\\z_3\end{pmatrix}=\begin{pmatrix}m_1\times a\\\ominus(\ominus m_2)\times a\\\ominus(\ominus m_3)\times a\end{pmatrix}$$

とおく．（Z は通常の配置になっている．）

$$Ma=\mathrm{diag}(Z)+\ominus\mathrm{diag}^r(Z)+\ominus\mathrm{diag}^l(Z)$$

となる．

ベクトル-マトリクス積も同様に記述できる．

マトリクス-マトリクス積　$L=MN$

$$Z^i=\begin{pmatrix}m_i\times\mathrm{diag}(N)\\m_i\times\ominus\mathrm{diag}^r(N)\\m_i\times\ominus\mathrm{diag}^l(N)\end{pmatrix}\quad i=1,2,3$$

と置くと，

$$l_1=\mathrm{diag}(Z^1)+\ominus\mathrm{diag}^r(Z^1)+\ominus\mathrm{diag}^l(Z^1)$$
$$\ominus l_2=\ominus\mathrm{diag}(Z^2)+\mathrm{diag}^r(Z^2)+\ominus\mathrm{diag}^l(Z^2)$$
$$\ominus l_3=\ominus\mathrm{diag}(Z^3)+\ominus\mathrm{diag}^r(Z^3)+\mathrm{diag}^l(Z^3)$$

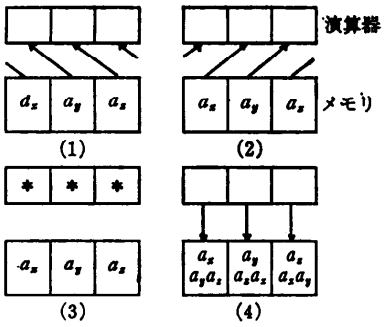
となる．

したがって，以下ではこれらを除いた基本演算に対して並列実行機構を議論する．

まず，（2個以上の演算器を使うと仮定して）演算器を幾つ使うかが問題となるが，評価関数として，演算器の利用率を用いると3個使うときがもっとも良くなることは明らかである．したがって，以下では演算器個数を3とする．

次に，これらの基本演算を行うためのメカニズムについて述べる．具体的には，3個の演算器（x-, y-, z-演算器と呼ぶ）と3個のメモリ（x-, y-, z- メモリと呼ぶ）を仮定し，その上にどのような機能があれば基本演算が逐次計算機で演算した場合の 1/3 の演算回数でできるかを検討する．なお，ベクトルデータは3個のメモリの同一アドレスに格納され，また，マトリクスデータは連続するアドレスに格納され，各行ベクトルは同一アドレスに格納されるものとする．

考えるべき演算は，二項演算，単項演算，シフト



$\ominus a \times \ominus a = (a_y \times a_z, a_z \times a_x, a_x \times a_y)$ の計算
(1) データ (a_x, a_y, a_z) を左上の演算器に転送する.
(2) データ (a_x, a_y, a_z) を右上の演算器に転送する.
(3) 乗算を行う.
(4) 結果を真下のメモリに転送する.

図-1 ベクトルの並列演算
Fig. 1 A parallel computation of vectors.

演算，対角要素の抽出演算だけである．二項演算と単項演算は対応する演算器とメモリの間にデータの転送ラインがあれば実行できる．シフト演算は x-, y-, z-メモリから y-, z-, x-演算器への転送ラインと x-, y-, z-メモリから z-, x-, $-y$ 演算器への転送ラインがあれば実行できる．対角要素の抽出演算は，3通りのアドレスオフセット機能(具体的には，$(+0, +1, +2)$, $(+2, +0, +1)$, $(+1, +2, +0)$ の3通り）があれば実行できる．したがって，これらの機能を具備した演算装置をつくればよい．一例として，$\ominus a \times \ominus a (= a_y \times a_z, a_z \times a_x, a_x \times a_y)$ の実行を図-1に示す．

ここで注意すべきことは，「データがメモリから演算器に転送され，演算を行って，結果がメモリに格納される.」というシーケンスで1回の演算が完了することを考えると，四則演算がともなうときシフト操作と対角要素抽出操作は四則演算に付随した操作となることである．たとえば，$\ominus a \times \ominus a$ 計算は1回の演算で実行できる．これは，演算の回数を考えるとき重要となる．

$G[x]$ の式 E に対し，E を逐次計算機で計算した場合の演算回数を $N_s(E)$ とし，上記の演算装置で計算した場合の演算回数を $N_p(E)$ とする．また，これらの比，

$$r = N_p(E)/N_s(E)$$

を演算並列度とよぶ.
このように定義すると r は，

$$1 \leqq r \leqq 3$$

の範囲の値をとる．

上記の各基本演算に対して，それを逐次的に計算した場合の演算回数を数えて比較すれば，

［性質7］
上記の各基本演算に対して r の値は3である.

という結果を得る(表-1).

表-1 演算並列度
Table 1 Parallelism of the vector operations.

演算			N_p	N_s	r
基本演算	ベクトル	二項演算	1	3	3
		単項演算	1	3	3
		シフト演算	1	3	3
	ベクトル-マトリクス積		5	15	3
	マトリクス-ベクトル積		5	15	3
	マトリクス	二項演算	3	9	3
		単項演算	3	9	3
		シフト演算	3	9	3
		対角要素抽出	1	3	3
	マトリクス-マトリクス積		15	45	3
拡張演算	総和		2	2	1
	連乗		2	2	1
	内積		3	5	1.67
	外積		3	9	3
	転置行列		3	9	3
	行列式		7	17	2.43
	逆行列		19	53	2.79
	二次形式		9	23	2.56

4. 拡張演算の並列実行

本章では，ベクトル演算やマトリクス演算で使用される内積，外積，逆行列などの種々の演算が2.で定義した基本演算の組合せで記述できることを示しその演算並列度を検討する．(線形代数[12]で使用される主要な演算について検討する.)

まず，以下の演算に有用な二つの演算を定義する.

ベクトル要素の総和：

$$\Sigma x = x + \ominus x + \ominus x = (\Sigma x_i, \Sigma x_i, \Sigma x_i)$$

ベクトル要素の連乗：

$$\Pi x = x \times \ominus x \times \ominus x = (\Pi x_i, \Pi x_i, \Pi x_i)$$

これらの演算は，本質的に逐次演算であり，並列演算はできない．これらの演算の演算回数は2である．また，r の値は1である．

ベクトル演算

① 内積：ここでは内積を次のように定義する．

$$(\boldsymbol{a},\boldsymbol{b})=\hat{\Sigma}(\boldsymbol{a}\times\boldsymbol{b})$$

内積演算の結果はスカラであるが，ここでは，ベクトルとして扱う．そうすることによる計算上の不都合は何もない．

② 外積：$[\boldsymbol{a},\boldsymbol{b}]=\ominus\boldsymbol{a}\times\ominus\boldsymbol{b}-\ominus\boldsymbol{a}\times\ominus\boldsymbol{b}$

③ rot：回転は次のように記述できる．

$$\mathrm{rot}(\boldsymbol{E})=\ominus\frac{d(\ominus\boldsymbol{E})}{d\boldsymbol{x}}-\ominus\frac{\boldsymbol{d}(\ominus\boldsymbol{E})}{d\boldsymbol{x}}$$

したがって，$\boldsymbol{E}\in\boldsymbol{G}[\boldsymbol{x}]$ なら性質6により $\mathrm{rot}(\boldsymbol{E})\in\boldsymbol{G}[\boldsymbol{x}]$ である．

④ div：発散は次のように定義する．

$$\mathrm{div}(\boldsymbol{E})=\hat{\Sigma}\frac{d\boldsymbol{E}}{d\boldsymbol{x}}$$

演算結果は，ベクトルと考える．回転と同様 $\boldsymbol{E}\in\boldsymbol{G}[\boldsymbol{x}]$ なら $\mathrm{div}(\boldsymbol{E})\in\boldsymbol{G}[\boldsymbol{x}]$ である．

マトリクス演算 (skewed 配置行列)

① 転置行列：転置行列は次のように記述できる．

$${}^{t}M=\begin{pmatrix}\mathrm{diag}(M)\\ \mathrm{diag}^{r}(M)\\ \mathrm{diag}^{l}(M)\end{pmatrix}$$

② 行列式：行列式の値は次のように記述できる．(ただし，演算結果はベクトル値とする．)

$$\det(M)=\hat{\Sigma}(\boldsymbol{m}_1\times\ominus(\ominus\boldsymbol{m}_2)\times\ominus(\ominus\boldsymbol{m}_3)-\boldsymbol{m}_1\times\ominus(\ominus\boldsymbol{m}_2)\times\ominus(\ominus\boldsymbol{m}_3))$$

③ 逆行列：逆行列は，以下のように定義される．

$$M^{-1}=\frac{1}{\Delta}\begin{pmatrix}\Delta_{11}&\Delta_{21}&\Delta_{31}\\ \Delta_{12}&\Delta_{22}&\Delta_{32}\\ \Delta_{13}&\Delta_{23}&\Delta_{33}\end{pmatrix}$$

ここで，$\Delta=\det(M)$，Δ_{ij} は (i,j) 余因子[12]である．ここでは，skewed 配置を前提としているから，上式は次のようになる．

$$M^{-1}=\frac{1}{\Delta}\begin{pmatrix}\Delta_{11}&\Delta_{21}&\Delta_{31}\\ \Delta_{32}&\Delta_{12}&\Delta_{22}\\ \Delta_{23}&\Delta_{33}&\Delta_{13}\end{pmatrix}$$

そうすると，余因子行列の各行は，次のように記述できる．

$$(\Delta_{11}\ \Delta_{21}\ \Delta_{31})=\ominus\mathrm{diag}^{r}(M)\times\ominus\mathrm{diag}^{l}(M)-\mathrm{diag}^{r}(M)\times\mathrm{diag}^{l}(M)$$

$$(\Delta_{32}\ \Delta_{12}\ \Delta_{22})=\ominus\mathrm{diag}(M)\times\ominus\mathrm{diag}^{l}(M)-\mathrm{diag}(M)\times\mathrm{diag}^{l}(M)$$

$$(\Delta_{23}\ \Delta_{33}\ \Delta_{13})=\ominus\mathrm{diag}(M)\times\ominus\mathrm{diag}^{r}(M)-\mathrm{diag}(M)\times\mathrm{diag}^{r}(M)$$

なお，3×3 行列の場合，直接計算のほうが掃き出し法で計算するより演算回数は少ない．

二次形式

二次形式は，$(\boldsymbol{x},M\boldsymbol{x})$ で与えられる．ここで，$\boldsymbol{x}$ はベクトル，Mはマトリクスである．したがって，以下のように記述できる．

$$(\boldsymbol{x},M\boldsymbol{x})=\hat{\Sigma}(\boldsymbol{x}\times M\boldsymbol{x})$$

以上の演算の演算並列度を表-1に示す．前章および本章の議論から分かるように，ここで検討しているベクトル演算において並列性を阻害する要因は，「演算の本質的逐次性」だけである．したがって，次の性質が成り立つ．

[性質8] 与えられた式が上記の要因を含まないならば，すなわち，$\hat{\Sigma}$，$\hat{\Pi}$ を含まなければ，与式は演算並列度3で計算できる．

5. おわりに

グラフィックス，ロボティックス，など種々の分野で基本演算となる3次元ベクトル，3×3 マトリクス演算の並列実行について考察した．まず，3次元ベクトル，3×3 マトリクスから構成される式の集合を考えその性質を調べた．そして，線形代数に現れる基本的なベクトル演算がこの集合に含まれることを示した．したがって，この集合は3次元ベクトル演算に現れるほとんどすべての式を含む．

ついで，この集合に属す式の並列演算機構を考え，その上で計算したときの演算並列度について検討した．残念ながらこの集合には本質的に逐次的な演算を必要とする式を含む．そのような式に対しては，並列演算器の演算並列度は低下する．しかし，それを含まない式に対しては，並列演算器は演算並列度3で式の計算を行える．この並列演算器は，著者らが提案している光線追跡法の並列処理を狙った計算機 SIGHT に，TARAI 演算器という名前で組み込まれ，光線追跡法の並列処理効率を著しく向上している[13),14)]．(TARAI 演算器の名前は，図-1からも分かるように，データがたらい回しに送られて演算が進むところに由来する．)

今後の課題として，種々の応用分野において，プログラムのどれくらいの部分がベクトル演算で記述できるか明らかにする必要がある．また，n次元への拡張も検討する必要がある．

謝辞 知能ロボット部高野陸男部長は本研究の機会を与えられた．滝川 啓主幹員，金子 博主幹員に

は，日ごろの討論に加え，論文の改良のために種々の有益なコメントをいただいた．橋本秋彦研究主任，斎藤隆文研究主任には，日ごろの討論で幾つかの有益な助言をいただいた．深く感謝する．

参 考 文 献

1) Whitted, T.: An Improved Illumination Model for Shaded Display, Comm. ACM, Vol. 23, No. 6, pp. 343-349 (June 1980).
2) Fu, K. S. 他著，本多他訳：ロボティクス，日刊工業新聞社（平成元年）．
3) 上村：プラズマの輸送現象と計算機シミュレーション，第17回プラズマ若手夏の学校テキスト，pp. 163-210（昭和53年8月）．
4) 阿部他：プラズマ科学における情報処理，情報処理，Vol. 22, No. 1, pp. 10-18（昭和56年1月）．
5) Magnenat-Thalmann, N. and Thalmann, D.: Image Synthesis Theory and Practise, p. 172, Springer-Verlag (1987).
6) 今月の表紙，日経 CG, No. 8, p. 45, 日経マグロウヒル．
7) 日高他：室内音場の合成シミュレーション(1)，(2)，日本音響学会講演論文集 2-7-11, 2-7-12, pp. 567-570（昭和63年3月）．
8) Nishimura, H. et al.: LINKS-1: A Parallel Pipelined Multimicrocomputer System for Image Creation, Proc. 10th Annu. Int'l Symp. on Comput. Arch., pp. 387-394 (1984).
9) Sato, H. et al.: Fast Image Generation of Constructive Solid Geometry Using a Cellular Array Processor, Computer Graphics, Vol 19, No. 3, pp. 95-102 (1985).
10) Hoshino, T.: An Invitation to the World of PAX, COMPUTER, Vol. 19, No. 5, pp. 68-79 (May 1986).
11) Naruse, T., Yoshida, M. and Takahashi, T.: SIGHT—A Dedicated Computer Graphics Machine, Computer Graphics Forum, Vol. 6, No. 4, pp. 327-334 (1987).
12) 佐竹：行列と行列式，裳華房（昭和46年）．
13) Takahashi, T., Yoshida, M. and Naruse, T.: Architecture and Performance Evaluation of the Dedicated Graphics Computer: SIGHT, Proc. Compint '87 (MONTECH '87), pp. 153-160 (1987).
14) 成瀬，吉田，高橋：SIGHT上の3次元ベクトル演算に関する考察，情報処理学会計算機アーキテクチャ研究会資料，CA 77-7 (1989).

（平成元年8月30日受付）

付録1 性質4の証明

$\ominus$について示す．($\oplus$についても同様)

$\ominus \boldsymbol{k}(\boldsymbol{x}) \in \boldsymbol{G}[\boldsymbol{x}]$なる式を考えたとき，$\boldsymbol{k}(\boldsymbol{x})$は次のいずれかの形をしていなければならない．(ここで$\boldsymbol{k}(\boldsymbol{x})$はベクトルであることに注意．)

① $\boldsymbol{f}(\boldsymbol{x}) \odot \boldsymbol{g}(\boldsymbol{x})$; $\odot$は二項演算
; $\boldsymbol{f}(\boldsymbol{x})$, $\boldsymbol{g}(\boldsymbol{x}) \in \boldsymbol{G}[\boldsymbol{x}]$

② $\odot \boldsymbol{f}(\boldsymbol{x})$; $\odot$は単項演算
; $\boldsymbol{f}(\boldsymbol{x}) \in \boldsymbol{G}[\boldsymbol{x}]$

③ $\ominus \boldsymbol{f}(\boldsymbol{x})$, $\oplus \boldsymbol{f}(\boldsymbol{x})$; $\boldsymbol{f}(\boldsymbol{x}) \in \boldsymbol{G}[\boldsymbol{x}]$

④ $M\boldsymbol{f}(\boldsymbol{x})$; Mは定数マトリクス
$\boldsymbol{f}(\boldsymbol{x})M$; $\boldsymbol{f}(\boldsymbol{x}) \in \boldsymbol{G}[\boldsymbol{x}]$

⑤ $\boldsymbol{a}$, $\boldsymbol{x}$; 定数／変数ベクトル

上記のうち①～④にシフト演算を適用すると，性質3により部分式のシフト演算に還元することができる．また，⑤の場合はシフト還元された形になっている．この還元手続きを繰り返し行うことにより$\ominus \boldsymbol{k}(\boldsymbol{x})$は中間レベルにシフト演算を含まない式に還元できることをいう．そのためには，有限回の繰り返しで還元手続きが終了することを言えばよい．

いま，式に含まれる基本演算の数を$N_{op}(\cdot)$で表す．

$$N_{op}(\ominus(\boldsymbol{f}(\boldsymbol{x}) \odot \boldsymbol{g}(\boldsymbol{x}))) > N_{op}(\ominus \boldsymbol{f}(\boldsymbol{x}))$$
$$N_{op}(\ominus(\boldsymbol{f}(\boldsymbol{x}) \odot \boldsymbol{g}(\boldsymbol{x}))) > N_{op}(\ominus \boldsymbol{g}(\boldsymbol{x}))$$
$$N_{op}(\ominus(\odot \boldsymbol{f}(\boldsymbol{x}))) > N_{op}(\ominus \boldsymbol{f}(\boldsymbol{x}))$$
$$N_{op}(\ominus(\ominus \boldsymbol{f}(\boldsymbol{x}))) > N_{op}(\ominus \boldsymbol{f}(\boldsymbol{x}))$$
$$N_{op}(\ominus(\oplus \boldsymbol{f}(\boldsymbol{x}))) > N_{op}(\boldsymbol{f}(\boldsymbol{x}))$$
$$N_{op}(\ominus(M\boldsymbol{f}(\boldsymbol{x}))) =$$
$$N_{op}(\textcircled{1} M(\ominus \boldsymbol{f}(\boldsymbol{x}))) > N_{op}(\ominus \boldsymbol{f}(\boldsymbol{x}))$$
$$N_{op}(\ominus(\boldsymbol{f}(\boldsymbol{x})M)) =$$
$$N_{op}((\ominus \boldsymbol{f}(\boldsymbol{x}))\textcircled{1} M) > N_{op}(\ominus \boldsymbol{f}(\boldsymbol{x}))$$

であるから，部分式の基本演算の数は元の式の基本演算の数より必ず小さい．$N_{op}(\ominus \boldsymbol{k}(\boldsymbol{x}))$は有限であるから，シフト演算の入った部分式はいつかは，$\ominus \boldsymbol{x}$, $\ominus \boldsymbol{a}$などの形になる．[証明終]

付録2 性質6の証明

$\boldsymbol{k}(\boldsymbol{x}) \in \boldsymbol{G}[\boldsymbol{x}]$に対して，$\boldsymbol{k}(\boldsymbol{x})$をシフト還元する．それをあらためて$\boldsymbol{k}(\boldsymbol{x})$とする．そうすると，$\boldsymbol{G}[\boldsymbol{x}]$の作り方から$\boldsymbol{k}(\boldsymbol{x})$は次のいずれかの形に書ける．

① $\boldsymbol{f}(\boldsymbol{x}) \odot \boldsymbol{g}(\boldsymbol{x})$; $\odot$は二項演算
; $\boldsymbol{f}(\boldsymbol{x})$, $\boldsymbol{g}(\boldsymbol{x}) \in \boldsymbol{G}[\boldsymbol{x}]$

② $\odot \boldsymbol{f}(\boldsymbol{x})$; $\odot$は単項演算
; $\boldsymbol{f}(\boldsymbol{x}) \in \boldsymbol{G}[\boldsymbol{x}]$

③ $\ominus \boldsymbol{x}$, $\oplus \boldsymbol{x}$

④ $Mf(x)$　；Mは定数マトリクス
　$f(x)M$　；$f(x)\in G[x]$
⑤ a, x　；定数／変数ベクトル

各場合につきその微分が$G[x]$に入るかをチェックする.

① $f(x)\odot g(x)$：$\odot$は+, −, ×, ／のいずれかである. $f'(x)$, $g'(x)\in G[x]$ならば性質5により$(f(x)\odot g(x))'\in G[x]$

② $\odot f(x)$：$f'(x)\in G[x]$ならば$(\odot f(x))'\in G[x]$

③, ⑤は明らか.

④ $(Mf(x))'$

$$=\left(\frac{\partial}{\partial x}(m_{11}f_1+m_{12}f_2+m_{13}f_3),\right.$$
$$\frac{\partial}{\partial y}(m_{21}f_1+m_{22}f_2+m_{23}f_3),$$
$$\left.\frac{\partial}{\partial z}(m_{31}f_1+m_{32}f_2+m_{33}f_3)\right),$$

ここで,

$$V=\begin{pmatrix}(f(x))'\\(\ominus f(x))'\\(\ominus f(x))'\end{pmatrix}$$

$$W=\begin{pmatrix}\mathrm{diag}(V)\times\mathrm{diag}(M)\\\ominus\mathrm{diag}^r(V)\times\mathrm{diag}^r(M)\\\ominus\mathrm{diag}^l(V)\times\mathrm{diag}^l(M)\end{pmatrix}$$

と置くと上式は,

$$\mathrm{diag}(W)+\ominus\mathrm{diag}^r(W)+\ominus\mathrm{diag}^l(W)$$

と書ける.

したがって, $(f(x))'$, $(\ominus f(x))'$, $(\ominus f(x))'\in G[x]$なら $(Mf(x))'\in G[x]$ である.
$(f(x)M)'$ も同様である.

以上から $f'(x)$, $g'(x)$, $(\ominus f(x))'$, $(\ominus f(x))'$ について, すなわち, 部分式について上記の①〜⑤のチェックを行うことにより, 元の $k'(x)$ が $G[x]$ にはいるかどうかが分かる. (ただし, $\ominus f(x)$, $\ominus f(x)$ についてはシフト還元を行った式をチェックする.) よって, 残るは有限回で部分式のチェックが終了することを言えばよい. ①, ②, ④について, シフト演算を除く基本演算の数を $N_{op}(\cdot)$ とすると,

$N_{op}(k(x))>N_{op}(f(x))$
$N_{op}(k(x))>N_{op}(g(x))$
$N_{op}(k(x))>N_{op}(\ominus f(x))$
$N_{op}(k(x))>N_{op}(\ominus f(x))$

である. なぜなら, シフト還元により, シフト演算以外の基本演算の数は変わらないからである. また, シフト還元を行った式では, シフト演算は $\ominus x$, $\ominus x$ の形で現れる. この式はこれ以上分解の必要がないので, 部分式の分解においてシフト演算の数をカウントする必要はない.

よって, 部分式は有限回で③, ⑤の形に帰着する. ゆえに, $k'(x)\in G[k]$ が言える.　[証明終]