Good Lattice Points の計算法について†

佐賀井 重雄†† 伏見 正則††

Good Lattice Points (GLP) 法は，s 次元単位超立方体 $I_s(=[0,1]^s)$ 上で定義された実数値関数の，I_s 上での定積分を数値的に求めるための有用な方法の一つである．ある種の関数族に対して，効率の良い GLP 公式を求めるための試みは以前から多数行われてきた．本論では，杉原正顯氏によって研究された新たな関数族に対して，GLP 法の効率の良い公式を速く求める方法について述べる．そのため，まず効率を評価する尺度となる量を定義し，その尺度に照らして良い公式を求める問題が，ある max-min 問題になることを示す．そして，この問題に対する三つの算法について述べる．第1は，杉原氏によるものであり，第2はそれに改良を加えたものである．そして，第3は格子 (Lattice) 上の最短ベクトルを求めるための Dieter の算法を応用したものである．数値実験の結果，第2および第3の算法を使用すると，第1の方法よりも数倍程度速く効率の良い公式を決定できることが確認された．また，この実験の結果，いくつかの良い公式が得られたが，それも合わせて示す．

1. まえがき

s 次元単位超立方体 $I_s=[0,1]^s$ 上で定義された実数値関数 $f(\boldsymbol{x})$ の I_s 上における定積分 $J=\int f(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$ の値を次式で近似する算法は，Good Lattice Points (GLP) 法と呼ばれている．

$$S=\frac{1}{N}\sum_{k=1}^{N} f\left(\left\{\frac{k}{N}\boldsymbol{g}\right\}\right)$$
$$=\frac{1}{N}\sum_{k=1}^{N} f\left(\left\{\frac{k}{N}g_1\right\},\cdots,\left\{\frac{k}{N}g_s\right\}\right). \qquad (1.1)$$

ただし，$\boldsymbol{g}=(g_1,\cdots,g_s)$ は整数ベクトル，$\{x\}$ は x の小数部分を表す．

ある種の関数族に含まれる関数 f に対して打切誤差 $|J-S|$ に対する評価を与え，その評価尺度に照らして良い $\boldsymbol{g}$ を探すという問題に関しては，長年にわたって多数の研究が行われてきた（例えば文献 3), 5), 7)）．

一方，杉原[6]は，下記の関数族 $\mathcal{E}_s(\beta,C)$ に対して同様の問題を検討し，GLP 法の有用性を示した．

$f\in\mathcal{E}_s(\beta,C)$ $(\beta,C>0)$

$\Leftrightarrow$ f は，I_s 上の実数値関数であって次のような絶対収束する多重 Fourier 級数に展開可能である．

$$f(\boldsymbol{x})=\sum_{\boldsymbol{h}} C_{\boldsymbol{h}}\exp(2\pi\mathrm{i}\,\boldsymbol{h}\cdot\boldsymbol{x}) \qquad (*\cdot *\text{は内積}),$$
$$|C_{\boldsymbol{h}}|\le C\exp\{-\beta(|h_1|+\cdots+|h_s|)\} \qquad (1.2)$$

ただし，$\sum_{\boldsymbol{h}}$ は，あらゆる整数ベクトル $\boldsymbol{h}=(h_1,\cdots,h_s)$ について加え合わせることを意味する．

しかしながら，良い $\boldsymbol{g}$ を探すためには，いわゆる“しらみつぶし”計算に頼らざるをえないため，N あるいは s が大きくなるとともに計算時間が急激に増大してしまうという難点があった．そこで本論文では，良い $\boldsymbol{g}$ を探索するための時間を短縮するような算法を提案し，各算法の効率を実験によって比較し，あわせて実験の過程で得られた良い $\boldsymbol{g}$ の例を示すことにする．

2. 誤差の指標と GLP の探索問題

本章では，杉原の研究[6]のうちで本論文に直接関係ある部分を要約し，誤差 $|J-S|$ に対する評価尺度と，それに基づいて良い $\boldsymbol{g}$ を求める問題がどのように定式化されるかを述べる．

公式 (1.1) を関数族 $\mathcal{E}_s(\beta,C)$ に対して適用した時の誤差は，以下のようにして評価できる．

$\left(J=\int f(\boldsymbol{x})\mathrm{d}\boldsymbol{x}=C_0\ \text{であることに注意}\right)$

$$|J-S|=\left|C_0-\frac{1}{N}\sum_{k=1}^{N}\left(\sum_{\boldsymbol{h}} C_{\boldsymbol{h}}\exp\left(2\pi\mathrm{i}\frac{k}{N}\boldsymbol{h}\cdot\boldsymbol{g}\right)\right)\right|$$
$$=\left|C_0-\sum_{\boldsymbol{h}} C_{\boldsymbol{h}}\left(\frac{1}{N}\sum_{k=1}^{N}\exp\left(2\pi\mathrm{i}\frac{k}{N}\boldsymbol{h}\cdot\boldsymbol{g}\right)\right)\right|$$
$$=\left|\sum_{\boldsymbol{h}\in H(N;\,\boldsymbol{g})} C_{\boldsymbol{h}}\right|$$
$$\le\sum_{\boldsymbol{h}\in H(N;\,\boldsymbol{g})}|C_{\boldsymbol{h}}|$$
$$\le C\sum_{\boldsymbol{h}\in H(N;\,\boldsymbol{g})}\exp(-\beta(|h_1|+\cdots+|h_s|)) \qquad (2.1)$$

ただし

$$H(N;\,\boldsymbol{g})=\{\boldsymbol{h}\mid\boldsymbol{h}\neq\boldsymbol{0},\ \boldsymbol{h}\cdot\boldsymbol{g}\equiv 0(\mathrm{mod}\,N),\ \boldsymbol{h}\in\boldsymbol{Z}^s\} \qquad (2.2)$$

† On Fast Methods of Searching for Good Lattice Points Formulae by SHIGEO SAGAI and MASANORI FUSHIMI (Department of Mathematical Engineering and Instrumentation Physics, Faculty of Engineering, University of Tokyo).

†† 東京大学工学部計数工学科

である．

(2.1)式の最右辺の値が小さくなるようなNおよび$\boldsymbol{g}$を求めることができれば，誤差の小さい数値積分公式が得られるわけであるが，それを実行することは一般には困難である．そこで比較的扱いやすい誤差の指標となる量：

$$\rho_s(N;\boldsymbol{g})\triangleq\min_{\boldsymbol{h}\in H(N;\boldsymbol{g})}(|h_1|+\cdots+|h_s|) \quad (2.3)$$

を導入する．この時，おおよそ

$$|\text{誤差}|\sim O(e^{-\beta\rho_s(N;\boldsymbol{g})}) \quad (2.4)$$

が成立する．(2.4)式は，正確な評価式ではないが，より精密な誤差評価式が杉原[6]で得られている．いずれの評価式を用いるとしても，結局は，$\rho_s(N;\boldsymbol{g})$が大きい公式が好ましいことには変わりがないので，以下そのようなNおよび$\boldsymbol{g}$を求めることを考える．

GLP法による数値積分公式としては，

(i) できるだけ高い精度で結果が求まり，

(ii) できるだけ少ない分点数でその高い精度が実現できる

ものが望ましい．

(i)の点に関して考えると，次元数sおよび分点数Nが与えられた場合，

$$\max_{\boldsymbol{g}\in G(N)}\rho_s(N;\boldsymbol{g})=\max_{\boldsymbol{g}\in G(N)}\min_{\boldsymbol{h}\in H(N;\boldsymbol{g})}(|h_1|+\cdots+|h_s|) \quad (2.5)$$

$$G(N)=\{\boldsymbol{g}\,|\,\boldsymbol{g}=(g_1,\cdots,g_s),\ 1\leq g_k\leq N/2\ \ (k=1,\cdots,s)\} \quad (2.6)$$

を実現するような$\boldsymbol{g}$を決定すればよい．これは誤差の評価式(2.1)における誤差のオーダの改善をすることに等しい．

一方(ii)については，$\rho_s^*(N)=\max_{\boldsymbol{g}\in G(N)}\rho_s(N;\boldsymbol{g})$の値が等しい$N$が一般に複数個あるので，そのうちで最小のものを選ぶのがよい．$\rho_s^*(N)=j$を実現する最小のNをN_jと書くことにするとき，もしも$N_j<N'$，$\rho_s^*(N_j)\geq\rho_s^*(N')$となったとすると，分点数として$N'$を用いる公式は，$N_j$を用いる公式より劣るので，採用しないのがよい．したがって，さまざまな精度について好ましいNの値を求めるためには，結局，Nの値を小さい方から1ずつ増加させて$\rho_s^*(N)$の値を求めていき，初めて過去の$\rho_s^*(N)$の最大値を上まわったNをつぎつぎに採用すればよいことになる．

3. GLP 探索上の問題点と対策

第2章の最後に述べたことを忠実に実行して，厳密な意味で最適な$\boldsymbol{g}$およびNを探索しようとすると，Nを動かしながら，各Nについて$\boldsymbol{g}$と$\boldsymbol{h}$とを動かして総当たり（しらみつぶし）計算を行わなければならず，探索時間が次元数sの増加とともに爆発的に増大することになり，事実上，探索は困難である．そこで本論文では，いくつかの点で探索時間を減少させることを試みる．そのうちで最も重点を置いたものは，Nと$\boldsymbol{g}$を固定して

$$\rho_s(N;\boldsymbol{g})=\min_{\boldsymbol{h}\in H(N;\boldsymbol{g})}(|h_1|+\cdots+|h_s|)$$

を求める際に，最適性をそこなうことなく，$\boldsymbol{h}$の探索範囲を狭める試みである．これについては，第4章で詳しく述べる．

次に，$\boldsymbol{g}$の探索範囲を(2.6)の$G(N)$から，下記の$G_0(N)$に制限することによって探索時間を減少させる．

$$G_0(N)=\{\boldsymbol{g}\,|\,\boldsymbol{g}=(1,g,g^2,\cdots,g^{s-1}),\ g\text{は}N/2\text{以下の奇数，}g^k\text{は}\bmod N\text{で考える}\} \quad (2.7)$$

これによって，Nが与えられた場合の$\boldsymbol{g}$に関する探索の手間は$O(N^s)$から$O(N)$に減少する．このようにしたとき，一般に最適性はそこなわれるが，それがそれほど大きくはないことが，後出の数値実験の結果によって示唆される．また，このようなタイプの$\boldsymbol{g}$は従来の研究でもよく用いられてきたものであり，Korobov型のGLPと呼ばれている．

なお，Nに関する探索については，次章で述べることにする．

4. 算　　法

4.1 算法A1（杉原の算法）

ミンコフスキーの不等式（文献1）などを参照のこと）に基づく，次の定理を用いて，$\boldsymbol{h}$の探索範囲を限定する．

定理　任意の$N\in\boldsymbol{Z}$, $\boldsymbol{g}\in\boldsymbol{Z}^s$に対して，

$$\rho_s(N;\boldsymbol{g})=\min_{\boldsymbol{h}\in H(N;\boldsymbol{g})}\left(\sum_{j=1}^{s}|h_j|\right)\leq\sqrt[s]{s!N}$$

これを用いると，$\rho_s(N;\boldsymbol{g})$は，次式によって計算できる．

$$\rho_s(N;\boldsymbol{g})=\min(\|g_2h_2+\cdots+g_sh_s\|_N+|h_2|+\cdots+|h_s|) \quad (4.1)$$

ただし，minは，

$$h_2\geq 0,\ (h_2,\cdots,h_s)\neq\boldsymbol{0}$$
$$|h_2|+\cdots+|h_s|\leq[\sqrt[s]{s!N}] \quad (4.2)$$
$$h_2,\cdots,h_s\in\boldsymbol{Z}$$

の範囲で探せばよい．

ここに，$\|x\|_N=\min\{\bmod(x,N),N-\bmod(x,N)\}$，

$\mathrm{mod}(x, N)$ は x を N で割ったときの余り，$[y]$ は y の整数部である．

この式を用いたときには，格子点 $(h_1,\cdots,h_s)$ の個数は $O(N^{(s-1)/s})$ となるので，$\rho_s(N; \boldsymbol{g})$ は，$O(N^{(s-1)/s})$ の手間で求められる．したがって $\max_{\boldsymbol{g}\in G_0(N)} \rho_s(N; \boldsymbol{g})$ を求める手間は，$O(N\cdot N^{(s-1)/s})=O(N^{(2s-1)/s})$ となる．

4.2 算法 A2（A1 に分枝限定的な考え方を取り入れた算法）

これは，$\rho_s(N; \boldsymbol{g})$ を求めるためには公式(4.1)を使用するが，$\max_{\boldsymbol{g}} \rho_s(N; \boldsymbol{g})$ を求める際に，問題全体としてみたときには max-min 問題 (2.5) となっていることを利用し，$\boldsymbol{h}$ に関する探索の段階で，次のようにして不要な探索を行わないようにするものである：

ある $\boldsymbol{g}=\boldsymbol{g}^*$ に対して $\rho_s(N; \boldsymbol{g})$ を求める際に，それ以前に調べたすべての $\boldsymbol{g}$ の中での $\rho_s(N; \boldsymbol{g})$ の最大値 $\rho_{\max}$ を保存しておき，$\boldsymbol{g}^*$ に関する $\boldsymbol{h}$ の探索の途中で，

$$\rho_s(N; \boldsymbol{g})\leq\rho_{\max}$$

となることが判明したら，それ以後の $\boldsymbol{h}$ に関する探索を打ち切る．

4.3 算法 A3 (Dieter の算法)

以下に紹介する Dieter[2] の算法は，s 次元ユークリッド空間における格子 (Lattice) $\boldsymbol{L}$ において最短のノルムを求めるための算法である．この算法は，本来は合同法乱数発生のためのよい乗数を求めるために考案されたものであるが，ここでは，積分の効率化への適用可能性を検討してみる．

$\boldsymbol{L}$ は s 本の線形独立な基底ベクトル $\boldsymbol{e}_i(1\leq i\leq s)$ によって張られるものとする．すなわち，

$$\boldsymbol{L}=\left\{\boldsymbol{x}=\sum_{k=1}^{s} z_k\boldsymbol{e}_k \mid z_i\in\boldsymbol{Z},\ 1\leq i\leq s\right\}$$

である．また，$\boldsymbol{L}$ に対して，共役な格子 $\boldsymbol{L}^*$ は，

$$\boldsymbol{L}^*=\left\{\boldsymbol{x}^*=\sum_{k=1}^{s} z_k^*\boldsymbol{e}_k^* \mid z_k^*\in\boldsymbol{Z},\ 1\leq k\leq s\right\}$$

によって定義される．ここに

$$\boldsymbol{e}_i\cdot\boldsymbol{e}_k^*=\delta_{ik}\ (\text{クロネッカーのデルタ}) \qquad (4.3)$$

である．

さらに $\boldsymbol{L}$ および $\boldsymbol{L}^*$ には，ノルム $\|\cdot\|$ および $\|\cdot\|^*$ が導入されているものとする．たとえば，$\boldsymbol{x}=(x_1,\cdots,x_s)$ として，$\boldsymbol{L}$ における $\boldsymbol{x}$ のノルムを

$$\|\boldsymbol{x}\|=\sum_{j=1}^{s}|x_j|$$

ととれば，$\boldsymbol{L}^*$ における $\boldsymbol{x}^*=(x_1^*,\cdots,x_s^*)$ のノルム（共役ノルム）は

$$\|\boldsymbol{x}^*\|^*=\max_{1\leq k\leq s}|x_k^*|$$

となる．

Dieter の算法において基本となるのは，次式である．

$$|\boldsymbol{x}^*\cdot\boldsymbol{x}|\leq\|\boldsymbol{x}^*\|^*\|\boldsymbol{x}\| \qquad (4.4)$$

ここで，$\boldsymbol{x}$ として，$\boldsymbol{L}$ における 0 でないノルム最小のベクトル $\boldsymbol{w}$，

$$\boldsymbol{w}=z_1\boldsymbol{e}_1+\cdots+z_s\boldsymbol{e}_s \qquad (4.5)$$

をとれば，(4.3)～(4.5)式により

$$\begin{aligned}|z_i|&=|\boldsymbol{e}_i^*\cdot(z_1\boldsymbol{e}_1+\cdots+z_s\boldsymbol{e}_s)|\\&=|\boldsymbol{e}_i^*\cdot\boldsymbol{w}|\leq\|\boldsymbol{e}_i^*\|^*\|\boldsymbol{w}\|\end{aligned} \qquad (4.6)$$

が成立する．$\boldsymbol{w}$ は計算の途中では不明であるが，$\|\boldsymbol{w}\|\leq\min_{1\leq j\leq s}\|\boldsymbol{e}_j\|$ であるから，(4.6)から

$$|z_i|\leq\|\boldsymbol{e}_i^*\|^*\min_{1\leq j\leq s}\|\boldsymbol{e}_j\| \qquad (4.7)$$

が導かれる．すなわち，基底ベクトルから，$\boldsymbol{w}$ を構成するための係数 $z_i\in\boldsymbol{Z}(1\leq i\leq s)$ は，(4.7)式を満たす範囲内にしか存在しないことがわかる．したがって $\boldsymbol{w}$ を求めるためには，

$$C_i=[\|\boldsymbol{e}_i^*\|^*\min_{1\leq j\leq s}\|\boldsymbol{e}_j\|],\quad 1\leq i\leq s \qquad (4.8)$$

とおけば，$\boldsymbol{e}_i$ の各係数 z_i に対して正負合わせて $(2C_i+1)$ 通り，全体では，

$$P=\prod_{i=1}^{s}(2C_i+1) \qquad (4.9)$$

通りの組合せについて探索を行えばよい．ただし，P が大きければ探索回数が大きくなってしまうので，その場合には下記の基底変換の組 (unimodular 変換の一つ) を施して，基底系の最小ノルムが小さくなるようにする：

基底変換

$$T_i:\left.\begin{matrix}\boldsymbol{e}_i\leftarrow\boldsymbol{e}_i\\ \boldsymbol{e}_i^*\leftarrow\boldsymbol{e}_i^*+\sum_{k\neq i}m_k\boldsymbol{e}_k^*\end{matrix}\right\},\quad\left.\begin{matrix}\boldsymbol{e}_k\leftarrow\boldsymbol{e}_k-m_k\boldsymbol{e}_i\\ \boldsymbol{e}_k^*\leftarrow\boldsymbol{e}_k^*\end{matrix}\right\}k\neq i \qquad (4.10)$$

$$T_i^*:\left.\begin{matrix}\boldsymbol{e}_i^*\leftarrow\boldsymbol{e}_i^*\\ \boldsymbol{e}_i\leftarrow\boldsymbol{e}_i+\sum_{k\neq i}m_k^*\boldsymbol{e}_k\end{matrix}\right\},\quad\left.\begin{matrix}\boldsymbol{e}_k^*\leftarrow\boldsymbol{e}_k^*-m_k^*\boldsymbol{e}_i^*\\ \boldsymbol{e}_k\leftarrow\boldsymbol{e}_k\end{matrix}\right\}k\neq i \qquad (4.11)$$

ここに，

$$m_k=[0.5+\boldsymbol{e}_i\cdot\boldsymbol{e}_k/\boldsymbol{e}_i\cdot\boldsymbol{e}_i],$$

$$m_k^*=[0.5+\boldsymbol{e}_i^*\cdot\boldsymbol{e}_k^*/\boldsymbol{e}_i^*\cdot\boldsymbol{e}_i^*]$$

である．

変換 $T_i(T_i^*)$ を施す際，いずれかの $m_k(m_k^*)(k\neq i)$ が 0 ではないならば，$T_i(T_i^*)$ は有効であるといい，そうでなければ（すべての $m_k(m_k^*)$ が 0）$T_i(T_i^*)$ は

無効であると呼ぶことにする．ある段階ですべての $T_i(T_i^*)$, $1\leq i\leq s$, が無効であっても，いったん $T_j(T_j^*)$ を施した後では，$T_i(T_i^*)$ が有効となることがある．

以上をまとめると，次のような算法が得られる．

M を十分大きな定数とする．

Dieter の算法

1° $C:=M$; flag $:=-1$; $i:=s+1$;
格子の基底系 $\{e_i\}$ を与え，$\{e_i^*\}$ を求める．

2° C_k $(k=1\sim s)$ および P をそれぞれ(4.8), (4.9)式で計算．
if $P<C$ **then**
$t^*:=s$; $t:=s$; $C:=P$; 3° へ飛ぶ．
else
if flag$=1$ **then** $t^*:=t^*-1$
else $t:=t-1$
if $t^*=0$ **then** 4° へ飛ぶ．

3° **if** $t^*+t=0$ **then** 5° へ飛ぶ．
$i:=i-1$; **if** $i=0$ **then** $i:=s$.
$k=1\sim s$, $k\neq i$ なる k に関して，m_k^* を計算．
変換 T_i が有効でない場合：
$t^*:=t^*-1$; **if** $t^*=0$ **then** 4° へ飛ぶ．
else 3° を再実行．
変換 T_i が有効である場合：
変換 T_i を(4.10)式に従って $\{e_i\}$, $\{e_i^*\}$ に施す．
flag $:=-1$; 2° へ飛ぶ．

4° **if** $t^*+t=0$ **then** 5° へ飛ぶ．
$i:=i-1$; **if** $i=0$ **then** $i:=s$.
$k=1\sim s$, $k\neq i$ なる k に関して，m_k を計算．
変換 T_i^* が有効でない場合：
$t:=t-1$; **if** $t=0$ **then** 3° へ飛ぶ．
else 4° を再実行．
変換 T_i^* が有効である場合：
変換 T_i^* を(4.11)式に従って $\{e_i^*\}$, $\{e_i\}$ に施す．
flag $:=1$; 2° へ飛ぶ．

5° $-C_i\leq z_i\leq C_i$, $1\leq i\leq s$ を満たすすべての整数の組合せを調べ，ノルム最小のベクトル $\boldsymbol{w}$ (4.5)を選ぶ．

この算法を，当面のわれわれの目的に対して適用すると，次のようになる．

$$\boldsymbol{h}\cdot\boldsymbol{g}=h_1+gh_2+\cdots+g^{t-1}h_t\equiv 0(\mathrm{mod}\ N),\quad h_1,h_2,\cdots,h_t\in\boldsymbol{Z} \tag{4.12}$$

を $\boldsymbol{h}$ に関する方程式とみなすと，そのすべての解は，

$$\boldsymbol{h}=\begin{bmatrix}h_1\\h_2\\\vdots\\h_t\end{bmatrix}=k\begin{bmatrix}N\\0\\\vdots\\0\end{bmatrix}+h_2\begin{bmatrix}-g\\1\\0\\\vdots\\0\end{bmatrix}+\cdots+h_k\begin{bmatrix}-g^{k-1}\\0\\\vdots\\0\\1\\0\\\vdots\\0\end{bmatrix}<k$$
$$+\cdots+h_t\begin{bmatrix}-g^{t-1}\\0\\\vdots\\0\\1\end{bmatrix} \tag{4.13}$$

という形に書き表せる．ここに，$k, h_2, \cdots, h_t$ は任意の整数である．したがって，(4.12)の解の集合は格子となるので，これに Dieter の算法を適用できる．

なお，上述のアルゴリズムをそのままプログラム化した場合 P の値が大きくて，各 z_i に関する全数探索が効率的に行えない場合が多い．そこで実際のプログラムにおいては，4° のステップで 2° へ飛ぶ直前を次のように変更した：M^* を5程度の定数とし，プログラムの初めで $k^*:=0$ とおいておくものとする．

if $t+t^*=0$ **then**
if $k^*\leq M^*$ **then**
$t:=s$; $t^*:=s$; $k^*:=k^*+1$; 3° へ飛ぶ．

上述の方法を取り入れることで，変換の回数は増加するが，P の値が小さくなり，5° の全数探索の範囲が小さくてすむようになった．

この算法では，必要な計算時間の主たる部分は，基底を変換する部分であるため，平均的には基底変換がある回数で抑えられるものとすれば，与えられた N, $\boldsymbol{g}$ に対して $\rho_s(N;\boldsymbol{g})$ を求める時間は，ほぼ $O(s^2)$ となり，N とは無関係になる．したがって $\max_{\boldsymbol{g}\in G_s(N)}\rho_s(N;\boldsymbol{g})$ を求める手間は平均的に $O(Ns^2)$ である．

5. 数値実験

5.1 計算時間の比較

前章までで述べた三つの算法に関して，どの算法によるのが一番速く効率の良い GLP 公式を決定できるかを調べるために，それぞれの算法で同じ N に対して $\max_{\boldsymbol{g}\in G_s(N)}\rho_s(N;\boldsymbol{g})$ を計算する時間の比較を，次元数 $s=4, 5, 6$ について行った．その結果を**図1～図3**に示

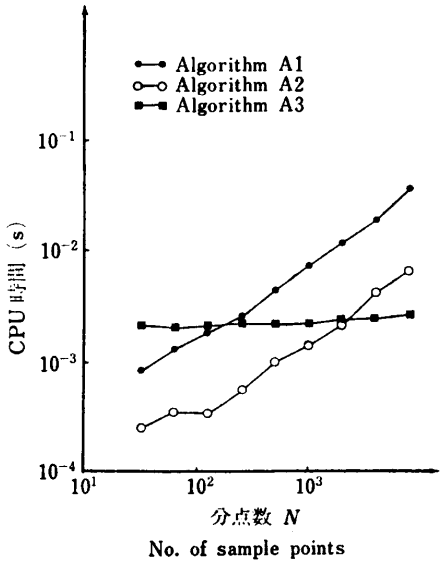


図 1 $\rho_s(N; \boldsymbol{g})$ を求めるための平均 CPU 時間
($|G_s(N)|$ 個の $\boldsymbol{g}$ に対する平均値)

Fig. 1 Average CPU time to calculate $\rho_s(N; \boldsymbol{g})$ (Average is based on the total CPU time corresponding to all $\boldsymbol{g}$'s in $G_s(N)$).

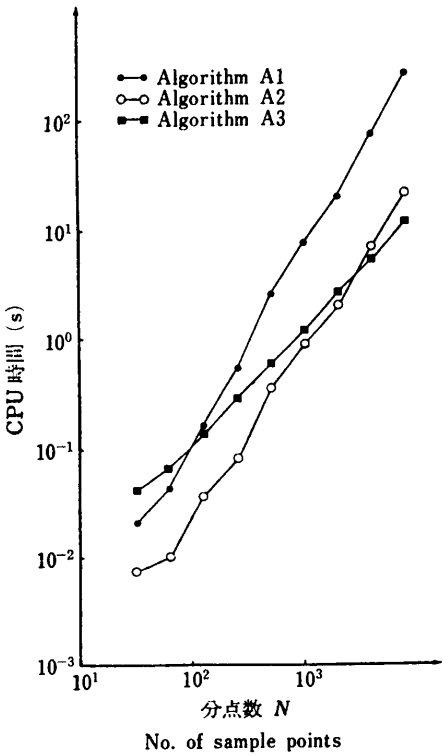


図 2 $\max_{\boldsymbol{g}} \rho_s(N; \boldsymbol{g})$ を求めるための CPU 時間

Fig. 2 CPU time to calculate $\max_{\boldsymbol{g}} \rho_s(N; \boldsymbol{g})$.

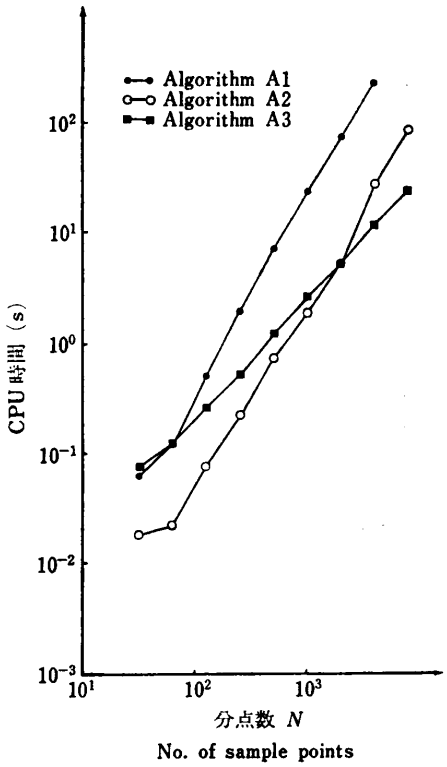


図 3 $\max_{\boldsymbol{g}} \rho_s(N; \boldsymbol{g})$ を求めるための CPU 時間

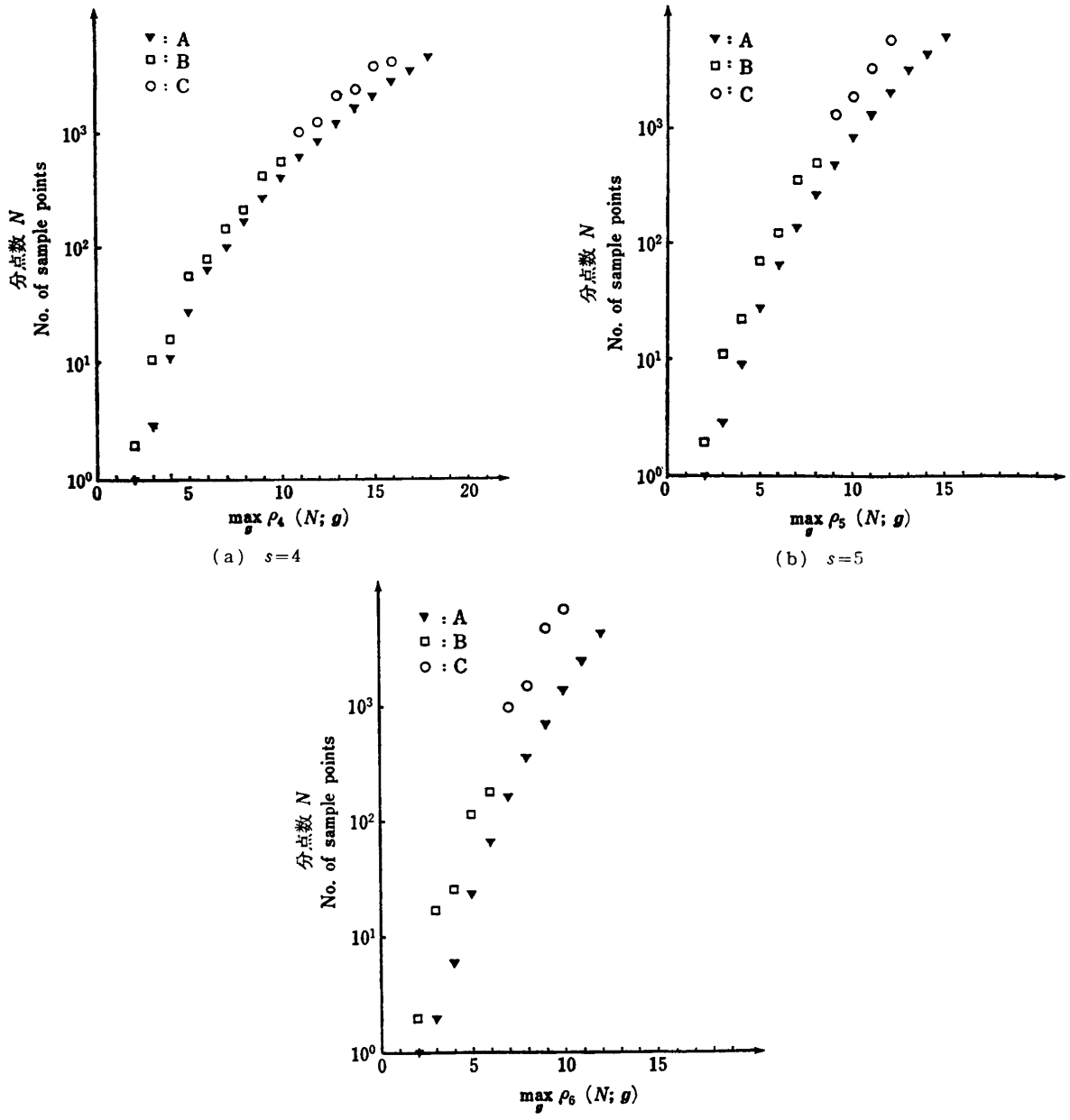
Fig. 3 CPU time to calculate $\max_{\boldsymbol{g}} \rho_s(N; \boldsymbol{g})$.

す.

なお，使用計算機は，東京大学大型計算機センター HITAC M-280 H，最適化 FORTRAN 77 (OPT=3) である.

図 2, 3 は，$\max_{\boldsymbol{g} \in G_s(N)} \rho_s(N; \boldsymbol{g})$ を $s=5$, 6 の場合に計算するためにかかった CPU 時間をいろいろな N の値に対して測定した結果を示したものである．また，$s=4$ の場合について同様の計算をする際の 1 個の $\boldsymbol{g}$ 当たりの平均 CPU 時間を示したのが図 1 である．図 1 を見ると算法 A3 については，平均 CPU 時間がほぼ一定であり，(p. 658 で述べた) "平均的には N によらない" という仮定の正当性を裏付けている.

図 1～図 3 のどの図においても，杉原の算法 A1 と比較すると，分枝限定的な考えを取り入れた算法 A2 の効果は明白である．また，算法 A2 と Dieter の算法 A3 とを比較すると，分点数 N が比較的小さいうちは，算法の単純な A2 の方が速いが，N が大きくなってくると，平均的な手間のオーダが勝っている A3 の方が有利となってくる．実際に，図 2 および 3 を見ると，A1, A2 は N に関してほぼ 2 乗のオーダで CPU 時間が増大しているが，A3 は N に関しては線形の



(a) $s=4$ (b) $s=5$ (c) $s=6$

図 4 $\max_g \rho_s(N; g)$ と，それを実現する最小の N との関係

A：定理から得られる N の下界
B：実験で得られた N（最小値であることが確認されたもの）
C：実験で得られた N（最小値であるかどうかは未確認）

Fig. 4 Relationship between $\max_g \rho_s(N; g)$ and the number of samples N.

A : The lower bound for N given by the Theorem.
B : Minimum N which attains $\max_g \rho_s(N; g)$.
C : Smallest N which attains $\max_g \rho_s(N; g)$ found by the numerical computation.

表 1 $\max_{g} \rho_s(N; g)$ と N, g

Table 1 $\max_{g} \rho_s(N; g)$ and N, g.

(a) $s=4$

$\max_g \rho_4(N; g)$	A	B	g				C
2	1	2	(1,	1,	1,	1)	2
3	3	11	(1,	5,	3,	4)	4
4	11	16	(1,	5,	9,	13)	4
5	27	57	(1,	11,	7,	20)	6
6	54	80	(1,	37,	9,	13)	6
7	100	191	(1,	59,	43,	54)	8
8	171	226	(1,	95,	211,	157)	8
9	274	435	(1,	191,	376,	41)	10
10	417	562	(1,	221,	509,	89)	10
11	610	1009	(1,	247,	469,	817)	12
12	864	1248	(1,	151,	337,	967)	13
13	1191	2061	(1,	137,	220,	1268)	14
14	1601	2320	(1,	389,	521,	829)	15
15	2110	3677	(1,	673,	658,	1594)	17
16	2731	3950	(1,	377,	3879,	883)	18

(b) $s=5$

$\max_g \rho_5(N; g)$	A	B	g					C
2	1	2	(1,	1,	1,	1,	1)	2
3	3	11	(1,	5,	3,	4,	9)	4
4	9	22	(1,	9,	15,	3,	5)	4
5	27	71	(1,	25,	57,	5,	54)	6
6	65	124	(1,	33,	97,	101,	109)	6
7	141	363	(1,	161,	148,	233,	124)	8
8	274	502	(1,	113,	219,	149,	271)	9
9	493	1333	(1,	163,	1242,	1163,	283)	10
10	834	1899	(1,	55,	1126,	1162,	1243)	11
11	1343	3301	(1,	197,	2498,	257,	1114)	13
12	2073	5959	(1,	13,	169,	2197,	4725)	14

(c) $s=6$

$\max_g \rho_6(N; g)$	A	B	g						C
2	1	2	(1,	1,	1,	1,	1,	1)	3
3	2	17	(1,	3,	9,	10,	13,	5)	4
4	6	26	(1,	7,	23,	5,	9,	11)	5
5	23	117	(1,	29,	22,	53,	16,	113)	6
6	65	182	(1,	23,	165,	155,	107,	95)	7
7	164	991	(1,	173,	199,	733,	952,	190)	9
8	365	1517	(1,	319,	122,	933,	1231,	1303)	10
9	739	3991	(1,	165,	3279,	2250,	87,	2382)	11
10	1389	6903	(1,	683,	5905,	3005,	5375,	5644)	13

A：定理から得られる N の下界

B：実験で得られた $\max_g \rho_s(N; g)$ を実現する N

C：N=B 欄の分点数とおいた時の $\max_g \rho_s(N; g)$ の上限

A: The lower bound for N given by the Theorem.

B: Smallest N which attains $\max_g \rho_s(N; g)$ found by the numerical computation.

C: The least upper bound for $\max_g \rho_s(N; g)$, where N=(sample number N in column B in the same row of the table).

オーダになっているのがわかる.

5.2 効率の良い N および g

算法 A2 および A3 を使って効率の良い N および g を求めることを試みた. 厳密な意味で最適な N を求めるためには, 2章の最後に述べているように, N を1ずつ増して $\max_{g \in G_s(N)} \rho_s(N; g)$ を計算しなければならないが, それでは計算時間が爆発的に増大するので, N がある程度以上大きくなった場合には, ほぼ次のような簡略法を用いた.

それまでに効率の良い GLP が求められている N の最大値を N_0 とする. N_0 より (例えば2倍程度) 大きな $N=N^*$ で,

$$\max_{g \in G_s(N)} \rho_s(N; g) > \max_{g \in G_s(N_0)} \rho_s(N_0; g) \qquad (5.1)$$

を満たすものを探す. 以下, $N_0 < N < N^*$ の範囲を探索することによって, なるべく小さな N で(5.1)を満たすものを探す.

図4は, $s=4, 5, 6$ について, このようにして求められた N と $\max_g \rho_s(N; g)$ との関係を示したものである. 横軸は $\max_g \rho_s(N; g)$ の値を示し, 縦軸は対応する N の値を示している. □印は, その N が $\max_g \rho_s(N; g)$ を実現する最小値であることが確認されているものであるが, ○印については最小値かどうかは未確認である. しかし, 定理によって与えられる N の下界 (▼印) と比べて, それほど大きくはない. なお, 各実験点に対する g の値を表1に示しておく.

図4から, また表1の C 欄の数値と対応する $\max_g \rho_s(N; g)$ 欄の数値を比較することによって, g を(2.6)の $G_0(N)$ に属するもの (Korobov 型) に限っても, 本論文で考えている指標の意味での効率はそれほど悪くならないことがわかる.

6. むすび

GLP 法の効率の良い公式をできるだけ短い計算時間で求める方法を提案し, 実際に数値実験を行い, その有用性を確認した. また, その算法を使用して次元数 $s=5, 6$ で分点数がある程度大きな公式を実際に求めた. また, それによると, Korobov 型の GLP でも, それほど効率が低下しないことがわかった.

なお，計算時間を短縮するという観点から見ると，ベクトル計算機を使用することが考えられるが，上述の算法を実際にプログラム化したとき，そのループ長としては，$[\sqrt[s]{s!N}]$ (算法 A1, A2), s (算法 A3) 程度にしかならないため，飛躍的な計算時間の短縮はできなかった．ベクトル計算機向けの算法ないしはプログラムの開発は，今後の検討課題である．

参考文献

1) Cassels, J. W. S.: *An Introduction to the Geometry of Numbers*, 343 pp., Springer-Verlag, Berlin (1959).
2) Dieter, U.: How to Calculate Shortest Vectors in a Lattice, *Math. Comp.*, Vol. 29, No. 131, pp. 827-833 (1975).
3) Kedem, G. and Zaremba, S. K.: A Table of Good Lattice Points in Three Dimensions, *Numer. Math.*, Vol. 23, pp. 175-180 (1974).
4) Knuth, D. E.: *The Art of Computer Programming*, Vol. 2, *Seminumerical Algorithms*, 2nd ed., Addison-Wesley, Reading, Mass. (1981).
5) Niederreiter, H.: Quasi-Monte Carlo Method and Pseudo-Random Numbers, *Bull. Amer. Math. Soc.*, Vol. 84, No. 6, pp. 957-1041 (1978).
6) 杉原正顯：Good Lattice Points を用いた多重数値積分，京都大学数理解析研究所講究録，No. 483, pp. 249-283 (1983).
7) Zaremba, S. K.: Good Lattice Points, Discrepancy and Numerical Integration, *Ann. Mat. Pura Appl.* (iv), Vol. 73, pp. 293-318 (1966).

(昭和 60 年 7 月 3 日受付)
(昭和 61 年 4 月 17 日採録)

佐賀井重雄（正会員）

昭和 36 年生．昭和 59 年東京大学工学部計数工学科卒業．昭和 61 年同大学院工学系研究科修士課程修了．同年（財）電力中央研究所に入所．現在，経営情報システムの開発などに従事している．

伏見　正則（正会員）

昭和 14 年生．昭和 38 年東京大学工学部応用物理学科（数理工学専攻）卒業．昭和 43 年同大学院博士課程修了．工学博士．埼玉大学講師，助教授を経て，昭和 49 年東京大学助教授（工学部計数工学科）となり現在に至る．応用統計学，オペレーションズ・リサーチ等の研究に従事．日本オペレーションズ・リサーチ学会，応用統計学会，日本数学会，American Statistical Association 等各会員．