Digital Computer による Gradient Method

戸島 濈

1. 序　　論

gradient method は **concave programming** の問題[1]の解をもとめる方法として組織的に確立されている algorithm のひとつであるが，これまでは，わずかな例外[2]を別にすれば，主として理論的側面にかぎって議論が展開されることが多かった。そして，その理論はそれ自身として，たとえば経済理論などにも密接な関係をもつものであることが示されている[3]。ところが，gradient method の本来の目的からいえば，これはいわば傍系的効用にすぎないのであって，この他に gradient method はもっと直接的な効用をもつものでなければならない。ここでいう直接的効用とはもちろん concave programming を実際に能率よくとくことができることをいみする。

linear programming の algorithm である simplex method の実用上の

（1） concave programming の問題はつぎの形式の問題である。

$$\max_{g(x)\geqq 0,\ x\geqq 0} f(x)$$ をあたえる x をもとめよ。

ここで，$x=<x_1, x_2, \cdots\cdots, x_n>$，$g(x)=<g_1(x), g_2(x), \cdots\cdots, g_m(x)>$（$<\cdots\cdots>$は列 vector をあらわす）で，$f(x)$, $g_j(x)(j=1, 2, \cdots\cdots, m)$ は $x\geqq 0$ なる x に対して定義された concave の実数値函数である。ただし，$x\geqq 0$, $g(x)\geqq 0$ は x と $g(x)$ のかく component が非負であることをいみする。また，concave function は以下のように定義される。R^n で n 次元 Euclidean space（ただし，$R=R^1$）をあらわし，$f: R^n\to R$ とする。$f(x)$ が x に関して concave であるとは，任意の R^n に属する x, x^o に対し

$$f(\theta x+(1-\theta)x^o)\geqq\theta f(x)+(1-\theta)f(x^o)\quad 0\leqq\theta\leqq 1$$

がなりたつことである。

（2） 〔5〕がそれである。

（3） この点に関して〔1〕，〔6〕はなどをみよ。ことに，〔1〕は網羅的で問題を概観するのに都合がよい。

価値については今日ではほとんど疑問の余地はないであろう。これに対して，gradient method に関してはその実用上（すなわち，具体的な計算上）の価値がとわれることは非常にすくなかったようにおもわれる。この場合，筆算の可能性ということは simplex method と gradient method の実用上の価値をきめるさいの決定的根拠とはならない。なぜならば，simplex method による linear programming の計算においても，問題の変数の数と拘束条件の数が多ければ実際には筆算は不可能である。また，不可能とはいわないまでも，筆算はきわめて能率のわるい結果しかもたらさないであろう。従って，このような場合にはどうしても計算機械を使用しなければならない。一方，gradient method はその性質からいって計算機械の使用を前提するが，必ずしも筆算が不可能であるわけではない。後述するような微分方程式である gradient equation を適当に定差方程式で近似しさえすれば，卓上計算機などを補助に使って比較的容易に目的に達することもできる。しかし，これとても程度の問題であって，計算量がきわめて多いときにはまったく能率が悪くなってしまうことは確実である。すなわち，筆算の可能性ということは simplex method にとっても gradient method にとってもそれほど大きな問題ではないと考えることができる。両者とも計算機械の使用を前提した algorithm であるとしなければならないのである。このことをはっきり認識したうえで，以下，本稿では gradient method を digital computer にのせることにともなっておこる諸問題を考察することにしたい。2. では問題を定式化し，3. では微分方程式の近似の問題にふれ，4. では計算機の programming 上で注意すべき点をのべ，5. では計算結果を呈示する。

2. concave programming の問題

まず，最初に実験に使用された問題を示しておこう。それはつぎのような簡単なものである。

$$\textit{maximize}\quad 2x_1+3x_2-\frac{1}{2}x_1^2-x_2^2,$$

$$\textit{subject to}\quad x_1+x_2\leqq 2,$$

$$x_1\geqq 0,\ x_2\geqq 0.$$

この問題の目的函数はあきらかに concave function であるから，これは concave programming の問題である。ところで，あらかじめこの問題の解をしっておくことは有益である。それは以下のようにしてもとめることができる。いま，たとえば

$$2x_1+3x_2-\frac{1}{2}x_1^2-x_2^2=4$$

とおけば，これは

$$\left(\frac{x_1-2}{1/\sqrt{2}}\right)^2+\left(\frac{x_2-3/2}{1/2}\right)^2=1$$

と変形することができるから，この式をみたす (x_1, x_2) の集合は (x_1, x_2)-平面における $x_1=2,\ x_2=\frac{3}{2}$ を中心とする長軸が $\sqrt{2}$，短軸が 1 の楕円である。目的函数の値を $17/4$ より連続的に小さくして行くと，この楕円と中心を共有してたがいに交わることのない楕円群（同心楕円）がえられる。目的函数の値は x_1, x_2 が実数であるかぎり $17/4$ より大にはならない。実際

$$f(x_1, x_2)=2x_1+3x_2-\frac{1}{2}x_1^2-x_2^2$$

を x_1 および x_2 について偏微分して 0 とおけば

$$fx_1=2-x_1=0,$$

$$fx_2=3-2x_1=0$$

であるから，$f(x_1, x_2)$ の最大値は $x_1=2,\ x_2=\frac{3}{2}$ で到達される。すなわち

$$f(2, 3/2)=17/4$$

である。つぎに

$$x_1+x_2\leqq 2,\ x_1\geqq 0,\ x_2\geqq 0$$

という不等式によって示される集合は (x_1, x_2)- 平面で $(2, 0)$ と $(0, 2)$ を

むすぶ直線と座標軸によって囲まれた三角形の辺上と内部である。以上によって，(x_1, x_2)- 平面上に拘束条件を示す三角形と目的函数に種々の値をあたえてえられる楕円群を同時に描いて，三角形に楕円が接する点の座標をもとめることができる。それは

$$x_1=1,\quad x_2=1$$

である。したがって

$$f(1, 1)=7/2$$

がわれわれの問題の最大値である。

さて，concave programming の問題を gradient method を用いてとくには，もとの問題をそれと同等な **saddle point**[4] をもとめる問題に変換しなければならない。この変換を可能にする条件がいくつかしられているが，つぎの **slater の条件**[5]はそのひとつである。

slater の条件

$g(x): R^n_+ \to R^m$ とし，表現 $g(x) \in R^m_+$ が programming の問題の拘束条件をあらわすものとする。このとき，$x' \in R^n_+$ なるある x^0 があつて

$$g(x^0) \in int\ R^m_+$$

となる。

われわれの concave programming の問題がこの条件をみたすことはあきらかである。たとえば

$$x_1=\frac{1}{2},\quad x_2=\frac{1}{2}$$

とすれば

$$g(x)=2-x_1-x_2=1$$

（4） R^{n+m} のうえに実数値函数 φ が定義せられているとき，E が R^{n+m} における φ の saddle point の集合であるとは，$\bar{x}\in R^n$, $\bar{u}\in R^m$ かつ $(\bar{x}, \bar{u})\in E$ ならば，すべての $x\in R^n$, $u\in R^m$ に対し

$$\varphi(x, \bar{u})\leqq\varphi(\bar{x}, \bar{u})\leqq\varphi(\bar{x}, u)$$

がなりたつことをいう。

（5） この条件をつかった変換については〔7〕をみよ。

となる。concave programming の問題を，それと同等な saddle point をもとめる問題に変換するさい，うえのような条件の吟味をわすれてはならない。現在までにしられている条件はすべて十分条件であるから，その条件をみたさなくても論理的には変換が不可能ということにはならないが，そのような場合，実際には gradient method の適用は不可能である。ただし，ひとつの十分条件をみたしさえすればよいのであるから，あるひとつの十分条件がなり立っていなければ，他の十分条件をつぎつぎに吟味してみればよい。ここにあげた slater の条件のほかに **Kuhn-Tucker の constraint qualification**[(6)]，ならびにその拡張である **Arrow-Hurwicz-Uzawa の constraint qualification**[(7)] などがあるが，ここでの問題に対してはいずれの条件もなり立っていることが容易に示されうる。一般に concave programming の問題の特殊な場合である linear programming の問題においては Kuhn-Tucker の constraint qualification はつねになり立っている。したがって，linear programming の問題を saddle point をもとめる問題に変換することはつねに可能である。なお，slater の条件は微分可能性を前提していないことに注意しておこう。

そこで，われわれの concave programming の問題に対応するそれと同等な saddle point をもとめる問題をつくることにしよう。いま

$$\varphi(x_1, x_2 : u_1, u_2, u_3) = 2x_1 + 3x_2 - \frac{1}{2}x_1^2 - x_2^2 + u_1(2 - x_1 - x_2) + u_2 x_1 + u_3 x_1,$$

$$(x_1, x_2) \in R^2, \quad (u_1, u_2, u_3) \in R^3_+$$

という函数を定義する[(8)]。φ は (x_1, x_2) に関して連続な concave function であり，また，任意の $(u_1, u_2, u_3) \in R^3_+$ に対して

（6）〔4〕をみよ。
（7）〔2〕をみよ。
（8）R^n_+ は R^n の非負象限をあらわす。

$$\chi(u_1, u_2, u_3)=\max_{(x_1, x_2)\in R^2}\varphi(x_1, x_2 : u_1, u_2, u_3)<+\infty$$

が存在する。そこで，われわれの concave programming の問題は $R^2\times R^3_+$ においてφの saddle point をもとめる問題と同等である。ここで注意すべきは，$x_1\geqq 0$, $x_2\geqq 0$ という条件が拘束条件の中にふくまれているため，φにおいて x_1, x_2 には符号の制限が陽表的には課されていないことである。ゆえに，あたえられた任意の $(u_1, u_2, u_3)\in R^3_+$ に対して，φに最大値 $\chi(u_1, u_2, u_3)$ をあたえる x_1, x_2 は

$$\varphi_{x_1}=2-x_1-u_1+u_2=0,$$

$$\varphi_{x_2}=3-2x_2-u_1+u_3=0$$

からもとめることができる。すなわち

$$x_1=2-u_1+u_2,$$

$$x_2=\frac{1}{2}(3-u_1+u_3)$$

である。したがって，この concave programming では 3 つの dual variable u_1, u_2, u_3 をもとめさえすれば，x_1, x_2 はただちにうえの式からもとまることになる。u_1, u_2, u_3 をもとめるには gradient equation をたててその収斂値を計算すればよい。いまの場合は 3 本の微分方程式がたてられることになる。gradient method の一般理論では gradient equation の数は primal variable の数と dual variable の数の合計にひとしいが，実際に計算を行うときには，しばしばその数をうえにのべたような技巧によって減少させて計算を容易にすることができる。一般にはφの concavity を利用することにより，任意の非負の dual variable に対して，φの有限な最大値を確保することができるならば，いつでもうえにのべたような方法で gradient equation の個数をへらすことが可能になる。さらに一般的にいえば，φの concavity がなくても，任意の非負の dual variable に対して，φの有限な最大値を確保しうるなら上述のことは可能である。しかし，実際の問題にさいして，φの concavity なしでは，有限最大値が存在しなかったり，仮に

存在してもその判定が容易でなかったりするであろう。また，最初から有限最大値の存在があきらかな問題も多くはないであろうから，結局，この方法がもっとも有効なのはやはり concave programming の問題にかぎられることになる。いまはとにかく concave programming の問題を対象としているのであるから，そのことはとくに問題ではない。

ところで，うえの問題はもうひとつ別様の gradient equation によって定式化することもできる。それは，x_1, x_2 の非負条件を拘束条件の中に加えないで

$$\varphi(x_1, x_2, : u) = 2x_1 + 3x_2 - \frac{1}{2}x_1^2 - x_2^2 + u(2 - x_1 - x_2),$$

$$(x_1, x_2) \in R^2_+, \quad u \in R_+$$

という函数を定義するのである。この場合も 3 本の gradient equation がえられて，前の場合と個数は一致する。これは一般にも一致する。なぜならば，primal variable が n 個，拘束条件が非負条件をふくめないで m 個あるとすれば，前の方法では primal variable の非負条件に対応する n 個の dual variable と m 個の拘束条件に対応する m 個の dual variable で合計 $n+m$ 個の gradient equation がえられる。後の方法では n 個の primal variable と m 個の dual variable でその合計は $n+m$ 個となり両者はたしかに一致する。したがって，両者とも equation の個数のうえからは同等な方法ではあるが，前者では gradient equation の収斂がただちに concave programming の問題の解ではないので，解をうるには変換をほどこさなければならないことになり，その点で後者の手続にいくらか劣る。しかし，前者は必ずしも concavity を前提としなくてもよいという点で論理的には後者よりも一般的であろう。concave programming の問題ではどちらの方法を採用しても equation の個数は同じであるから，どちらをとるかは上述のことを考慮にいれたうえで判断をくだせばよい。ここでは前者をとることにした。

3. gradient equation

よくしられている gradient equation はつぎの形をとる。[9]

$$\dot{y}_k=\delta_k\varphi_k(y) \quad (k=1, 2,\cdots,l),$$

$$\delta_k=\begin{cases}0 & \text{if } y_k=0 \text{ and } \varphi_k(y)<0,\\ 1 & \text{otherwise.}\end{cases}$$

この微分方程式でもっとも特徴的なことは，δ_k という不連続函数が右辺にふくまれていることである。δ_k は解が負領域に入るのを防止する **stopper ではなく**，解を非負領域のみで接続するいわば **connector であるが**，計算機の program をくむときにこれがひとつの困難を提供する。一般に微分方程式を数値計算法によってとくには微分方程式を適当な定差方程式にあらためなければならない。もっとも簡単には

$$\frac{dx}{dt}=\lim_{\Delta t\to 0}\frac{x(t+\Delta t)-x(t)}{\Delta t}$$

という関係を利用して，Δt を十分小さくとって，$\frac{dx}{dt}$ を $\frac{x(t+\Delta t)-x(t)}{\Delta t}$ でおきかえればよいが，これは Δt が適当でないと誤差が累積する危険の伴う方法である。そこで，その危険をすくなくするために通常は Runge-Kutta-Gill 法などが使用されるが，実際の計算にはやや複雑である。ここでは，誤差の多い危険を覚悟しながら，$\dot{x}$ をたんに $\frac{x(t+\Delta t)-x(t)}{\Delta t}$ でおきかえるだけの近似式を採用することにした。その理由は以下の行文から次第にあきらかになるであろう。

ところで，どのような近似式を用いようとわれわれが gradient equation をあつかうさいに直面する大きな問題は変数の non-negatvity の条件をい

---

(9) $y=\langle y_1, y_2,\cdots,y_l\rangle=\langle x, u\rangle$, $x\in R^n_+$, $u\in R^m_+$, $l=n+m$,

$$\varphi_k(y)=\frac{\partial\varphi(y)}{\partial y_k}\ (k\leqq n),\quad \varphi_k(y)=-\frac{\partial\varphi(y)}{\partial y_k}\ (k\geqq n+1),$$

$$\dot{y}_k=\frac{dy_k}{dt}.$$

かにして保つかということである。上掲の gradient equation はすでに注意したようにそれ自身の中に解の non-negativity を保つような装置をそなえていないので，その点をとくに考慮した計算機の program は非常に困難になる。仮にそのような program がうまくくめたとしても真の値との誤差が問題になるであろう。たとえば，解が saddle point に収斂するかわりに発散したり，永久に振動したりするかもしれない。事実そのような事態がおこることがみとめられているのである。[10] この困難をさけるには gradient equation はがあらかじめ解を non-negative の範囲にとじこめておく装置を built-in していればよい。この目的にかなう微分方程式としてさしあたって 2 つ考えることができるが，そのうちのひとつは，[11] 方程式の形そのものが複雑なうえ，ある種の計算ではその収斂のおそさが予想されているので，[12] その詳細な検討は他の機会にゆずることにして，ここではのこりのもうひとつの gradient equation の利用を考えた。それはつぎの形をとる。[13]

$$\dot{y}_k=\hat{\varphi}_k(y) \quad (k=1, 2,\cdots,l).$$

この微分方程式はその形だけからいえばはじめにあげた gradient equation と類似した簡潔な形をしておりあつかいやすい。これはもともと不連続函数 δ をふくまないことを目的として analogue computer 用に考案された方程式であるが，[14] 同様な使いやすさから digital computer 用としても役立ちうるということは注目してもいい事実であろう。しかし，これはそのまま使うわけにはいかない。なぜなら，そのままでは，方程式の形からあきらかなように，解が非負のものとしてもとまらないからである。そこで，すこしばかり改変しよう。まず，すでにのべた方針で近似すれば

---

(10) 〔6〕をみよ。

(11) 著者の北大に提出した博士論文の第 6 章で論じられた。なお，〔3〕の(Ⅲ)式はその linear case である。

(12) 〔8〕をみよ。

(13) $\hat{\varphi}_k(y)=\varphi_k(\hat{y})$, $\hat{y}=<\hat{y}_1, \hat{y}_2,\cdots, \hat{y}_l>$, $\hat{y}_k=max(0, y_k)$.

(14) 〔3〕をみよ。

$$y_k(t+\Delta t)=y_k(t)+\Delta t\hat{\varphi}_k(y) \quad (k=1,\ 2,\cdots,\ l)$$

がえられる。いっそうていねいにかくと，これは

$$y_k(t+\Delta t)=y_k(t)+\Delta t\varphi_k(\hat{y}) \quad (k=1,\ 2,\cdots,\ l),$$

$$\hat{y}=<\hat{y}_1,\ \hat{y}_2,\cdots,\ \hat{y}_l>,\ \hat{y}_k=max(0,\ y_k)$$

ということである。われわれはこの定差方程式の $t+\Delta t$ に対応する非負解を

$$max[0,\ y_k(t)+\Delta t\varphi_k(\hat{y})] \quad (k=1,\ 2,\cdots,\ l)$$

によって定義する。ここで，あらためて

$$z_k=\hat{y}_k \quad (k=1,\ 2,\cdots,\ l),$$

$$z=<z_1,\ z_2,\cdots,\ z_l>$$

とおけば，われわれは解の非負性を保存した定差方程式を

$$z_k(t+\Delta t)=max[0,\ z_k(t)+\Delta t\varphi_k(z(t))] \quad (k=1,\ 2,\cdots,\ l)$$

とかくことができる。これは定差方程式としても非常に簡単なものであるから，計算機の programming もかなり容易である。しかし，何といってもこの定差方程式の最大の強みは解の non-negativity の条件がきわめて簡単な形で built-in されていることである。gradient method によって実際に計算を行う場合，この定差方程式を利用することが便利である。

さて，われわれの concave programming の問題をうえの定差方程式にしたがって定式化してみよう。

$$\varphi_{u_1}=2-x_1-x_2,$$

$$\varphi_{u_2}=x_1,$$

$$\varphi_{u_3}=x_2$$

であるから，まとめて

$$\varphi_u=\begin{pmatrix}\varphi_{u_1}\\ \varphi_{u_2}\\ \varphi_{u_3}\end{pmatrix}=\begin{pmatrix}2\\ 0\\ 0\end{pmatrix}-\begin{pmatrix}1 & 1\\ -1 & 0\\ 0 & -1\end{pmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix}$$

とかくことにする。そこで

$$u(t+\Delta t)=\begin{pmatrix}u_1(t+\Delta t)\\ u_2(t+\Delta t)\\ u_3(t+\Delta t)\end{pmatrix}=max\left(\begin{pmatrix}0\\0\\0\end{pmatrix},\ \Delta t\left\{\begin{pmatrix}2\\0\\0\end{pmatrix}-\begin{pmatrix}1&1\\-1&0\\0&-1\end{pmatrix}\begin{pmatrix}x_1\\x_2\end{pmatrix}\right\}\right)$$

がえられる。ただし

$$max(<0,\ 0,\cdots,0>,<a,\ b,\cdots,c>)=(max(0,\ a),\ max(0,\ b),\cdots,\ max(0,\ c))$$

である。すでにえられている方程式

$$x(t)=\begin{pmatrix}x_1(t)\\x_2(t)\end{pmatrix}=\begin{pmatrix}1&0\\0&2\end{pmatrix}^{-1}\left(\begin{pmatrix}2\\3\end{pmatrix}-\begin{pmatrix}1&-1&0\\1&0&-1\end{pmatrix}u(t)\right)$$

を使って，これを変形すれば

$$u(t+\Delta t)=max\{0,(I-\Delta tBA^{-1}B')u(t)-\Delta t(b-BA^{-1}a)\},$$

$$A=\begin{pmatrix}1&0\\0&2\end{pmatrix},\quad B=\begin{pmatrix}1&1\\-1&0\\0&-1\end{pmatrix},\quad a=\begin{pmatrix}2\\3\end{pmatrix},\quad b=\begin{pmatrix}2\\0\\0\end{pmatrix}$$

となる。$detA\neq 0$ なるゆえ A^{-1} はあきらかに存在する。ここで，$\Delta t=10^{-5}$ とおいて，うえの定差方程式を具体的にかけばつぎの通りである。

$$u_1(t+\Delta t)=0.999985u_1(t)+0.00001u_2(t)+0.000005u_3(t)+0.000015,$$

$$u_2(t+\Delta t)=max(0,\ 0.00001u_1(t)+0.99999u_2(t)-0.00002),$$

$$u_3(t+\Delta t)=max(0,\ 0.000005u_1(t)+0.999995u_3(t)-0.000015).$$

この定差方程式では

$$u(0),\ u(\Delta t),\ u(2\Delta t),\cdots,\ u(t\Delta t),\cdots$$

なる系例がえられるが，これを簡単のため以下では

$$u(0),\ u(1),\ u(2),\cdots,\ u(t),\cdots$$

とかくことにする。以上によって，われわれの concave programming の問題をとくことは $\Delta t=10^{-5}$ の場合つぎの方程式によって，$x_1(t)$, $x_2(t)$ の極限値をもとめることと同等である。

$$u_1(t+1)=0.999985u_1(t)+0.00001u_2(t)+0.000005u_3(t)+0.000015,$$

$$u_2(t+1)=max(0,\ 0.00001u_1(t)+0.99999u_2(t)-0.00002),$$

$$u_3(t+1)=max(0,\ 0.000005u_1(t)+0.999995u_3(t)-0.000015),$$

$$x_1(t)=2-u_1(t)+u_2(t),$$

$$x_2(t)=\frac{3}{2}-\frac{1}{2}u_1(t)+\frac{1}{2}u_3(t),$$

$$u_1(0)=u_2(0)=u_3(0)=\frac{1}{2}.$$

これでとにかく計算機によってとかれるべき問題が確定したわけである。

4. 計算機の programming について

つぎに計算機の programming についてすこし考えてみよう。gradient method の実用化のためにはそれの library routine（または sub-routine）をつくっておくことがのぞましいが，ここではそのような一般的な問題はさておいて，前節で定式化した定差方程式の収斂値をもとめるにはどのような program をくめばよいかを考えてみることにしよう。

いうまでもないことであるが，gradient method は一種の近似法で，計算を次第にくりかえしておこなうならどこまでも真の値に接近していくものである。ところで，どのような計算機の桁数も必ず有限であるから，この近似を無限につづけるというわけにはいかない。もちろん，任意に指定した桁までの近似値をもとめることは適当な工夫によって可能であるが，問題によっては program が複雑化するであろうし，場合によっては実際には不可能かもしれない。したがって，通常は計算機の桁数が近似値の精度に対する制約となる。なお，固定小数点演算であろうと浮動小数点演算であろうとこの事情には何らかわりない。そこで，gradient equation を digital computer でとく場合，計算機の桁数の制約を考えて何桁までもとめるかということをきめなければならない。もちろん，それは一般的にきまっていることではなくて，問題に応じて個別的に決定しなければならないが，これをうまくきめれば計算時間その他で非常な利益をうることになるであろう。なお，これに

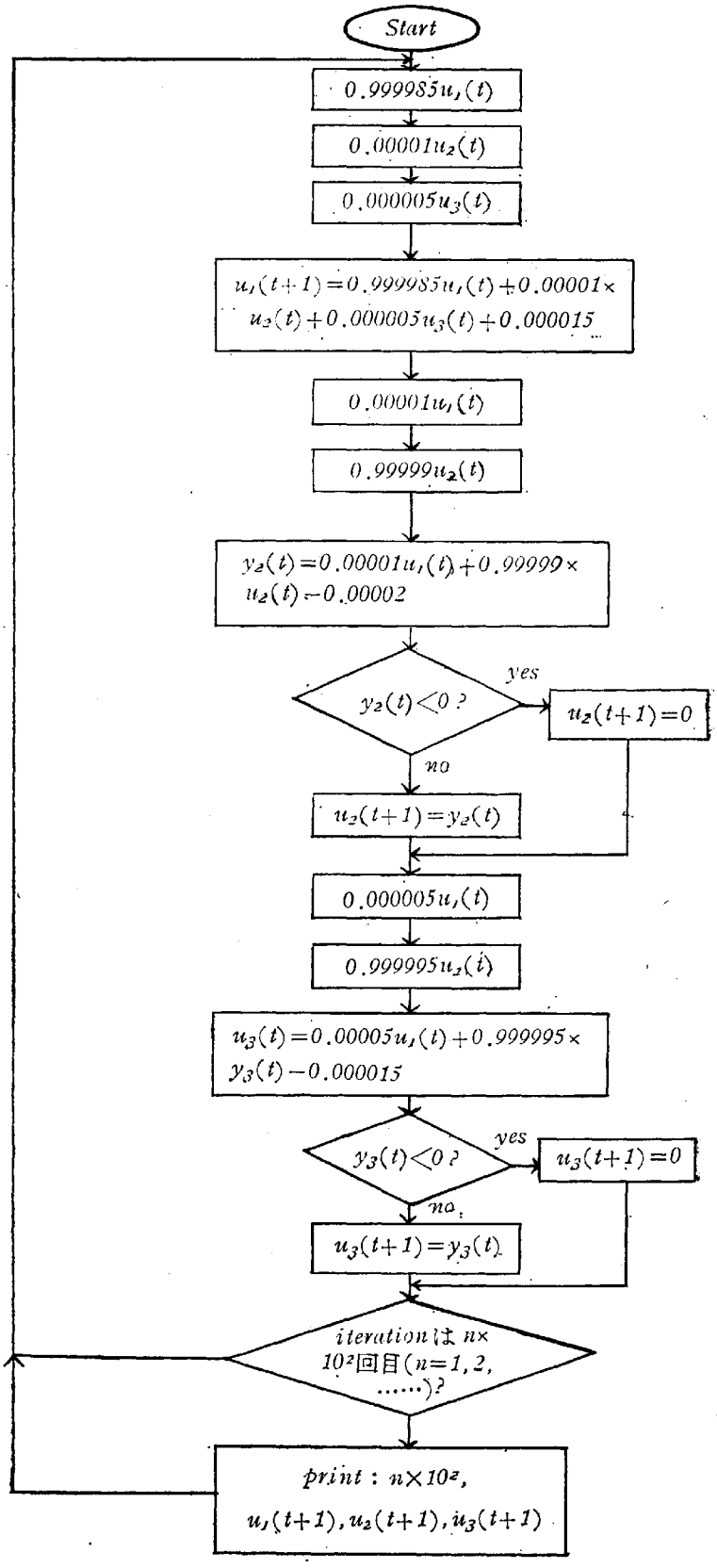
ついてはのちに有効桁数の問題としていま一度ふれる。さて，桁数がきまってもくりかえし計算をどこまでつづけたらよいかという問題がある。これに対しては，たとえば，桁数が固定小数点演算で小数点以下10桁であれば

$$|u(t)-u(t+n)|<10^{-9}$$

となったときに計算を stop させるという方法が考えられる。ここで，t, n は任意の非負の integer である（ただし $n\neq 0$）。しかし，この方法では計算を stop させたところがはたして問題の解であるか，いなかがあきらかではない。すなわち，その値が定差方程式の収斂値に9桁まで一致しているか，いなかに関しては判定をくだすことはできない。いくつかの step ののちにふたたび $u(t)$ の値が小数点以下2桁目あたりで動くかもしれないのである。gradient method の理論では必ずしも単調収斂を保証しているわけではないから，当然に定差方程式の近似においてもその解が単調収斂であると期待することはできない。したがって，単純にうえの方法によって計算を stop させることには危険がともなう。そこで，これにかわる方法として，実際に計算結果を適当な間隔をもって print out させ，それを監視しながら計算をすすめていけば，最初の方法がふくむ危険を防止することができるであろう。しかし，この方法では print out する数字は所望の結果ばかりでなく，中間結果もあるわけであるから，print する回数が場合によっては非常に多いことになる。どんな計算機組織でも計算時間より output の時間の方が長いので print out をどのようにおこなわせるかによって，全体の所要時間がかなり変ってくるであろう。print out が line printer によって可能である場合にも1回の iteration の結果を毎回 print out させることは非能率的である。したがって，われわれはある間隔をおいて print out される iteration の結果をみながら，どこで計算をうちきらせるかをきめなければならない。この判定は場合によってはくだすのがむずかしいこともあろうが，判定が不安ならば何回でも計算を続行させて，ある程度の確信がえられたときにはじめて計算を stop させればよい。それゆえ，計算機の program

は，stop の指令をいれることなく，途中の結果を適当に print out しつつどこまでもくりかえし計算が可能であるような形でくむことがのぞましい。われわれもここでの問題に対してそのような program を採用した。また，中間結果としては iteration 100回ごとに print out させることにした。以上の点を考慮した計算手続はつぎに示す flow chart の通りである。

Fow Chart of Gradient Method



5. 計 算 結 果[15]

$\Delta t=10^{-5}$ で iteration 100回ごとに中間結果を print out させ，初期条件が $u_1(0)=u_2(0)=u_3(0)=\frac{1}{2}$ である場合の計算結果はつぎの通りである。なお，われわれの concave programming の問題の解はすでにみたように，$x_1=x_2=1$ であるから，dual variable u_2, u_3 はすべて 0 でなければならない。[16] したがって，簡単な計算によって，$u_1=1$ であることがわかる。下に示す結果は4捨5入をしていない。

第1表　$\Delta t=10^{-5}$ の場合

iteration number	u_1	u_2	u_3
100	0.5014	0.4980	0.4980
200	0.5029	0.4960	0.4960
300	0.5044	0.4940	0.4940
400	0.5059	0.4920	0.4920
500	0.5074	0.4900	0.4900
1000	0.5147	0.4801	0.4801
2000	0.5289	0.4606	0.4606
3000	0.5426	0.4415	0.4415
4000	0.5559	0.4227	0.4227
5000	0.5687	0.4041	0.4042
10000	0.6258	0.3161	0.3161
20000	0.7105	0.1596	0.1596
30000	0.7632	0.0245	0.0245
40000	0.7965	0.0000	0.0000
50000	0.8248	0.0000	0.0000
100000	0.9172	0.0000	0.0000
200000	0.9815	0.0000	0.0000
300000	0.9958	0.0000	0.0000
400000	0.9960	0.0000	0.0000
500000	0.9997	0.0000	0.0000
600000	0.9999	0.0000	0.0000

(15)　計算に使用された機械は，OKITAC—5090A，NEAC—2203G である。
(16)　最適解の必要条件による。〔4〕をみよ。

この第1表の結果を検討すればただちに収斂が非常におそいことがわかる。ここに示した最終結果は iteration を60万回おこなったのであるが，これは gradient equation でいえば区間〔0,6〕を60万等分して逐次に解をもとめていったものにひとしく，そのことを考えればこのような収斂のおそさは当然ともいえる。そのかわり，これらの値はもとの微分方程式である gradient equation の近似解としてはかなり精度が高いのではないかとおもわれる。しかし，現在われわれが必要としているものは定差方程式の解そのものより収斂値であるから，収斂速度のおそいことは決定的な難点といわなくてはならない。そこで，収斂速度をもうすこし早めることにしよう。そのためには， difference の Δt を 10^{-5} よりもふやしてやればよい。ここでは order を1桁あげることにした。つぎに示す結果は $\Delta t = 10^{-4}$ の場合で初期条件などは全部前の場合と同じである。

第2表　$\Delta t = 10^{-4}$ の場合

iteration number	u_1	u_2	u_3
100	0.5147	0.4826	0.4850
200	0.5290	0.4654	0.4702
300	0.5429	0.4485	0.4556
400	0.5563	0.4318	0.4411
500	0.5692	0.4153	0.4268
1000	0.6278	0.3359	0.3569
2000	0.7172	0.1903	0.2262
3000	0.7756	0.0590	0.1054
4000	0.8103	0.0000	0.0000
5000	0.8369	0.0000	0.0000
6000	0.8594	0.0000	0.0000
7000	0.8790	0.0000	0.0000
8000	0.8959	0.0000	0.0000
9000	0.9104	0.0000	0.0000
10000	0.9228	0.0000	0.0000

われわれがすでにしっている収斂値 $u_1=1,\ u_2=u_3=0$ と比較して考えるなら，ここにえられた結果は一応は満足すべきもののようにみえる。最終結果は gradient equation でいえば，$t=1$ に対応する値である。$\Delta t=10^{-5}$ の場合のそれと異っているのは difference のきざみ方の相違から由来するものであるが，これを gradient equation の近似解という観点からみれば，その差は $\Delta t=10^{-4}$ とした近似の定差方程式の，$\Delta t=10^{-5}$ とした近似の定差方程式に対する相対的誤差をあらわすものと解することができる。したがって，前者は gradient equation の近似としては後者より精度がおちる。しかし，うえで注意したように，ここでの目的は gradient equation の近似解をもとめることにあるのではないから，この点に関して誤差の累積をそれほど警戒する必要はない。なお，第2表で u_2，u_3 が4000回以降0.0000になっているが，これは必ずしも計算機の内部で完全に0になっているのでないことを注意しておこう。このことを示すために小数点以下7桁までの結果の一部をかかげる。

第3表　下7桁までの場合

iteration number	u_1	u_2	u_3
3000	0.7756175	0.0590646	0.1054131
3400	0.7916035	0.0097346	0.0594436
4000	0.8103187	0.0000585	0.0000308
5000	0.8367416	0.0000585	0.0000308
10000	0.9228863	0.0000585	0.0000308

以上でみられるように，$\Delta t=10^{-4}$ の場合は u_2，u_3 に関してはかなり急速に適当な桁数内でもとめる値に収斂していることがみとめられるが，u_1 に関しては必ずしもそうとはいえない。そこで，さらに収斂をはやめることにしよう。以下では，$\Delta t=10^{-3}$ としてうえと同じ計算をおこなってみた。ただし，その場合，有効桁数を $\Delta t=10^{-4}$ の場合と同じに考えることにはいささか疑問がのこる。$\Delta t=10^{-4}$ では有効桁を小数点以下4桁までとったが，

$\Delta t=10^{-3}$ の場合はその4桁目は $\Delta t=10^{-4}$ の場合と同様な精度をもつことにはならないであろう。この観点にたって，ここでは小数点以下3桁までを4捨5入なしで考える。初期条件は以前と同じである。

第4表　$\Delta t=10^{-3}$ の場合

iteration number	u_1	u_2	u_3
100	0.627	0.316	0.356
200	0.714	0.159	0.226
300	0.770	0.024	0.104
400	0.804	0.000	0.000
500	0.831	0.000	0.000
1000	0.920	0.000	0.000
2000	0.982	0.000	0.000
3000	0.996	0.000	0.000
4000	0.999	0.000	0.000
5000	0.999	0.000	0.000

容易にみられるように，この場合は5000回のくりかえし計算で，ここで指定された桁の範囲内でほぼ完全に収斂している。いま，print out の時間を考慮しなければ，計算機がこの問題で5000回のくりかえし計算に要する時間はごくわずかなものであろう。われわれは実際には中間結果を100回ごとに print out したので，多少，時間がかかったが，それでも満足すべきはやさであった。

もとの concave programming の問題の解をもとめるには，うえでえられた u_1, u_2, u_3 の値を

$$x_1=2-u_1+u_2,$$

$$x_2=\frac{1}{2}(3-u_1+u_3)$$

に代入すればよい。こうして，$x_1=1.001$, $x_2=1.0005$ がえられる。

以上で，Δt が 10^{-5}, 10^{-4}, 10^{-3} の3つの場合について計算をおこなったが，それらを通じてあきらかになった注意すべき点をいま一度まとめてお

こう。まず第1に Δt の適切な評価という問題である。Δt の大きさいかんで収斂のはやさが非常に違ってくるので評価は慎重におこなわなければならない。収斂をはやめるために Δt を過度に大きくすると gradient equation の近似といういみがまったく失われてしまうことになりかねないので注意を要する。ここで gradient equation の近似解をもとめることが直接の目的でなくても，もともと理論的にその収斂が証明されている gradient equation を digital computer によってとくことが本来の目的なのであるから，定差方程式がある程度 gradient equation に近似していなければ gradient method の理論はつかえないことになるのである。そこで，Δt は定差方程式が gradient equation を近似しているといういみを失わない範囲内でなるべく大きくすることがのぞましい。実際に Δt の評価の組織的方法が確立されているわけではないので，この問題の根本的解決は将来にのこされることになる。つぎに，第2として有効桁数をいくらにするかという問題がある。われわれはうえでは一応 $\Delta t = 10^{-4}$ のときは小数点以下4桁，$\Delta t = 10^{-3}$ のときは小数点以下3桁という便宜的な基準にしたがった。しかし，一般にはこれがもっとも妥当であるという根拠はとぼしく，個々の場合に定差方程式の係数の大きさなどから具体的に考えなければならないであろう。これものこされた問題である。

なお，最後に $\Delta t = 10^{-2}$ の場合の結果を参考までにあげておこう。

第5表　$\Delta t = 10^{-2}$ の場合

iteration number	u_1	u_2	u_3
100	0.92	0.01	0.00
200	0.98	0.01	0.00
300	0.99	0.01	0.00
400	0.99	0.01	0.00
500	0.99	0.01	0.00

u_2 の値の収斂がよくないが，これで x_1,, x_2 の値を計算すると

$$x_1 = 1.02, \quad x_2 = 1.005$$

となる。$\Delta t = 10^{-2}$ のようにかなり粗い近似の場合は精度のよい解がえられると期待すべきではない。

引 用 文 献

〔1〕 Arrow, K. J. and L. Hurwicz. "Decentralization and Computation in Resource Allocation", in R. W. Pfouts (ed.), *Essays in Economics and Econometrics*, Chapel Hill : The University of North Carolina Press, 1960.

〔2〕 Arrow, K. J., L. Hurwicz and H.Uzawa. "Constraint Qualifications in Maximization Problems", *Naval Research Logistics Quarterly*, Vol. 8, No. 2.

〔3〕 Kose, T.. "Solutions of Saddle Value Problems by Differential Equations", *Econometrica*, Vol. 24, No. 1.

〔4〕 Kuhn, H. W. and A. W. Tucker. "Non-linear Programming", in J. Neyman (ed.), *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley and Los Angeles : University of California Press, 1955.

〔5〕 Marschak, T.. "An Example of a Modified Gradient Method for Linear Programming", in K. J. Arrow, L. Hurwicz and H. Uzawa (eds.), *Studies in Linear and Non-linear Progrmming*, Stanford, Calif. : Stanford University Press, 1958.

〔6〕 Marschak, T.. "Centralization and Decentralization in Economic Organization", *Econometrica*, Vol. 27, No. 3.

〔7〕 Uzawa, H.. "The Kuhn-Tucker Theorem in Concave Programming", in *Studies*.

〔8〕 古瀬大六.「計画法と分権的決定」商学討究，第4巻第1号.

(1964・4・9)