JAERI 1263

JAERI 1263

最適化手法の評価と最適化コード・システム SCOOPの開発

1979年11月

日本原子力研究所

Japan Atomic Energy Research Institute

JAERI レポート

この報告書は，日本原子力研究所で行われた研究および技術の成果を研究成果編集委員会の審査を経て，不定期に刊行しているものです．

入手（資料交換による），複製などの問合わせは，日本原子力研究所技術情報部（〒319-11 茨城県那珂郡東海村）あて，お申しこみください．なお，このほかに財団法人原子力弘済会資料センター（茨城県那珂郡東海村日本原子力研究所内）で複写による実費頒布をおこなっております．

JAERI Report

Inquiries about the availability of reports and their reproduction should be addressed to the Division of Technical Information, Japan Atomic Energy Research Institute, Tokai-mura, Naka-gun. Ibaraki-ken, Japan.

編集兼発行　日本原子力研究所
印　　刷　三美印刷株式会社

正　誤　表

ページ	列　　行	誤	正
1	右　下から 17	プログラム等号制約	プログラム，等号制約
7	左　下から 7	$\tilde{\pi}_i$ はベクトル π	$\tilde{\pi}_i$ はベクトル $\tilde{\pi}$
14	右　上から 2	mng)	ming)
17	Fig. 6	$g_3=0$	$g_2=0$
		$g_2=0$	$g_3=0$
		（2と3の番号を付けかえる）	
21	右　上から 11	方法を提し，	方法を提案し，
	下から 21	$\partial g_m/\partial x_m$	$\partial g_m/\partial x_n$
22	右　上から 10	λ 関して	λ に関して
26	右　IFN の欄 上から 1	212	213
30	右　F-final の欄 上から 9	0.30749−2	0.30749−3
39	左　下から 17	ROTA,	ROTAX,

最適化手法の評価と最適化コード・システム SCOOP の開発

日本原子力研究所東海研究所原子炉工学部

鈴 木 忠 和

1979年5月22日受理

線形問題，非線形問題および制約条件式の有無などの最適化問題の形に応じた合計 32 個の最適化プログラムが開発・整備され，各プログラムの持つ手法の安定性や収束効率などに対する評価が行なわれた．また，その評価に基づいて，開発・整備されたプログラム群を統合化した最適化コードシステム SCOOP の第一版が完成された．

SCOOP は，効率性，信頼性，有用性，汎用性というシステムにとって4つの重要な基本的性能が考慮されたコードシステムであり，ユーザの持つあらゆる形の最適化問題に対して，それに最も適した手法を用いて大域的な最適点が探索される．

Comparative Evaluation of Various Optimization Methods and the Development of An Optimization Code System SCOOP

Tadakazu Suzuki

Division of Reactor Engineering

Tokai Research Establishment

Japan Atomic Energy Research Institute

(Received May 22, 1979)

Thirty two programs for linear and nonlinear optimization problems with or without constraints have been developed or incorporated, and their stability, convergence and efficiency have been examined. On the basis of these evaluations, the first version of the optimization code system SCOOP-I has been completed.

The SCOOP-I is designed to be an efficient, reliable, useful and also flexible system for general applications. The system enables one to find global optimization point for a wide class of problems by selecting the most appropriate optimization method built in it.

Key words;

linear optimization, nonlinear optimization, constrained linear programming, unconstrained nonlinear programming, constrained nonlinear programming, quadratic programming, goal programming, integer programming, decomposition, direct search method, descent search method, sensitivity analysis, optimization code system SCOOP

目　　次

Contents

1. 序　　論

「最適化」という言葉と我々の社会的，経済的活動とは深いかかわりあいを持っている．我々は日常生活の中で，自分に最も適したものを求めるという行動の最適化を計っている．その場合，「適している」という言葉にはどのような意味で適しているのかという基準が与えられているであろう．また，行動の対象となる領域には一般にいろいろな制約が課せられているにちがいない．このような，定められた最適性の基準にしたがって，与えられた制約領域の中から自分に最も適したものを探すという我々の一般的な行動様式は，まさに最適化問題そのものである。行動の対象となる分野の構造が単純である場合には，解は我々の過去の経験とその構造に関する情報を分析することにより得られるであろうが，その構造が複雑で構成要素の数が多くなると，もはや我々の経験だけで解を求めることは難かしくなる．そのような複雑なシステムの最適化の問題としては，我々が最も関心を持っている原子力の分野では，たとえば，最適な核エネルギー開発を行なうための開発計画の長期戦略[1),2),3)]や炉型戦略の問題，最適炉心計算，最適遮蔽，燃料交換最適化や最適炉停止などの原子炉システムの最適化問題[4),5),6)]が挙げられる．そのほかに，将来における原子力，石油，石炭，天然ガスなどのエネルギー供給源から自動車，航空機，船舶などの輸送部門，冷暖房，厨房などの民生部門，鉄鋼などの産業部門へいたる最適なエネルギーネットワークフローの解析や，エネルギー政策の決定などのエネルギーシステム解析の問題[7)]，さらに商業経済の分野では商品の生産計画，仕入れ管理あるいは輸送計画の問題などがある．

上述のように複雑なシステムの最適化問題を解く1つの方法は，そのシステムに類似したモデルを作り，システムと同一の状態になるように構成要素（パラメータ）を調整し，その挙動を類推するシミュレーションによる方法がある．他の方法は，システムの挙動を数式化した数学モデルを作成し，数理計画法の問題として扱う方法である．我々は，最適化問題を数理計画法の問題として扱った場合についての最適化の手法について考えることにする．

最適化の手法は微分法や変分法などの古典的な方法と反復法によるものがあるが，最近の電子計算機の超大型化，超高速化に伴い後者の発展が著しく，線形，非線形，制約条件式の有無などの対象とするシステムの特性に応じた多くの手法が開発されている．しかしながら，それらの手法に対する総合的なベンチマーク・テストはほとんど行なわれておらず，したがって最適化プログラムは個々の形かせいぜい数個がパッケージ化された形でユーザーの使用に供されているのが現状である．このようなことは，線形から非線形へなど，問題の形が変わるごとに別の最適化プログラムの必要性を生じさせる．また最適化問題を解くうえで大切なことは「最適化問題のための最適な手法の選択」であると言われているが，反面このことは，最適化の手法に対する相当の知識をユーザに求めることになり，問題を解くことが手段であり，目的でないユーザーにとってこれは煩わしいことであろう．あらゆる問題に適した万能型の単独の手法が存在するなら，このような問題は生じないのだが，残念ながら最適化問題の分野にはそのような統一的手法は存在しない．このようなことから，あゆる問題に対処できるように多くの手法を統合化した最適化コードシステムの開発が望まれており，著者は数年前から原子炉工学部内に発足した「システム解析手法研究ワーキンググループ」における活動の一環として，その要望に応えるため各種の最適化プログラムの開発，整備を行ない，整備されたプログラムのベンチマーク・テストを実施してきた．これまでに合計32個の最適化プログラムの開発・整備を終え，それらの中の26個のプログラムのベンチマーク・テストを完了し，各プログラムの安定性，計算効率に対する評価を行なった．これらの中には，著者が提案した球面探索法[8)]によるプログラム，Allran と Johnsen によって提案された方法を修正した修正 Allran-Johnsen 法によるプログラム等号制約を扱えるように近似計画法を修正したプログラムおよび制御ランダム法の収束を加速した方法によるプログラムが含まれている．また，著者が作成したプログラムは前記の開発，修正したものを含めて計14個である．また，ベンチマーク・テストの結果を踏まえて整備されたプログラムを統合化した最適化コードシステム SCOOP (System of Codes for Optimization Problems の略) のバージョンIが完成した．これによって，これまで単独のプログラムでは解くことができなかったユーザーの持つ多くの最適化問題を解くことができる．

本報では，第2章において最適化問題の分類を行ない，その分類にしたがって，第3章では線形計画法を中心に，目標計画法，分解原理法，切除平面法などの線形システムで定義される問題に対する手法の説明と開発した各プログラムの紹介，第4章では非線形最適化手法を制約条件式を持つ問題と持たない問題のそれぞれに対す

る場合に分けて紹介する．制約条件を持つ問題に対する手法では，フレキシブル・トレランス-シンプレックス法，コンプレックス法，加速制御ランダム法などの発見的方法によるもの，近似計画法，切除平面法などの線形近似によるもの，および修正 Allran-Johnsen 法や Powell などの変換法によるものの紹介とそれらの手法に基づくプログラムの紹介がなされ，ベンチマーク・テストの結果が示されている．制約条件を持たない問題に対する手法では，シンプレックス法，座標回転法，パターン探索法，球面探索法，加速制御ランダム法などの直接探索法と呼ばれる手法と，最急降下法，共役方向法，Newton 法，可変計量法，PARTAN 法などの降下法と呼ばれる手法の紹介と，それらの手法に基づく計 16 個のプログラムのベンチマーク・テストの結果が解説されている．最後の第5章では，我々が開発した最適化コードシステム SCOOP-バージョンIの基本的な流れの説明が行なわれている．

2. 最適化問題の分類

最適化問題は，数学的には数理計画法の問題として扱われる．この場合必要なことは

（ⅰ） 解析の対象となっているシステムの挙動を数学的にモデル化すること．

（ⅱ） 最適性の基準を設定すること．

（ⅲ） 最適化の手段を選択すること．

である[9]．（ⅰ）の数学的モデルの作成のためには，例えばシステムを支配している物理的，あるいは経済的法則からシステムの変数間の関係式を導きだすことが必要である．（ⅱ）は目的関数あるいは評価関数 (objective function) と呼ばれるもので，（ⅰ）の数学的モデルを構成する関数群と同じ変数により定義されるものである．（ⅲ）は最適化を実行するために必要なアルゴリズムである．最適化問題は数理計算法の問題として，$\boldsymbol{x}$ を n 次元ユークリッド空間 E^n 内のベクトルとすると，

$$\left.\begin{array}{l}\underset{\boldsymbol{x}\in E^n}{\text{Minimize}}\ F(\boldsymbol{x})\\ \text{Subject to}\\ \boldsymbol{x}\in R\end{array}\right\} \qquad (2\text{-}1)$$

のように定式化される．この中で F は設定された最適性の基準であり，R はシステムの挙動を示す関数群の集まりで，制約条件式と呼ばれている．この制約条件は一般には

$$g_i(\boldsymbol{x})\geq 0,\quad i=1,\cdots,m$$

という形をとる．但し m は制約条件式の個数である．

問題 (2-1) においては最小化問題を考えているが，最大化問題に対しては目的関数の符号を変えることにより最小化問題に帰着できるので，以後一般性を失なうことなく最適化問題を最小化問題と考えることにする．

(2-1) で定式化される最適化問題は，目的関数と制約条件式の形により，次のように分類することができる．

Optimization Problem

- Linear Problem
- Nonlinear Problem
  - Constrained
  - Unconstrained

さらに上のような問題の形に応じていろいろな最適化の手法が開発されている．第3章以降では我々が開発・整備した最適化プログラムを上述の分類にしたがって紹介する．

3. 線形問題に対する最適化手法

目的関数および制約条件式が独立変数 $\boldsymbol{x}=(x_1, \cdots, x_n)^t$ の一次式で表されるとき，すなわち問題(2-1)が

$$\left.\begin{aligned} &\underset{\boldsymbol{x}\in E^n}{\text{Minimize}}\ F(\boldsymbol{x})=\boldsymbol{c}^t\cdot\boldsymbol{x} \\ &\text{subject to} \\ &\quad A\boldsymbol{x}=\boldsymbol{b} \\ &\quad \boldsymbol{x}\geq 0 \end{aligned}\right\} \qquad (3\text{-}1)$$

という形をとるときには線形計画法 (Linear Programming) の問題となり，これは数理計画法の問題の中で最も基礎的な問題である．但し(3-1)において A は $m\times n$ の係数行列，$\boldsymbol{b}$ は m 次元の右辺ベクトル，$\boldsymbol{c}$ は n 次元のコスト係数ベクトルで肩の t は転置を意味している．この問題に対する解法も Dantzig によって開発された単体法[10),11)] (Simplex Method) と呼ばれる確固とした手法があり，もし (3-1) の実行可能領域が有界であるならば，必ず有限回の探索で求める最適点を得ることができる保証が与えられている．

(3-1) で与えられる線形計画法の問題では，制約領域は一般には m 個の超平面

$$H_i=\sum_{j=1}^{n} a_{ij}x_j,\quad i=1, \cdots, m$$

で仕切られる凸多角面体となる．また目的関数も n 次元空間内の超平面となることから，求める最小値はその凸多面体上の端点 (extreme point) のいずれかで与えられることになる．Dantzig によって開発された単体法は，1つの端点 $\boldsymbol{x}^0$ から $F(\boldsymbol{x}^1)<F(\boldsymbol{x}^0)$ になるような他の端点 $\boldsymbol{x}^1$ を探索するアルゴリズムである．なお (3-1) では等号制約を考えているが，不等号制約

$$\sum_{j=1}^{n} a_{ij}x_j\geq b_i$$

に対しては正のスラック変数 s_i を導入して

$$\sum_{j=1}^{n} a_{ij}x_j-s_i=b_i$$

として等号制約に置換できるので，線形計画法の問題の一般形としては (3-1) の形を考えることが多い．但し，上式において a_{ij} は行列 A の (i, j) 成分，x_j, b_i は，それぞれベクトル $\boldsymbol{x}, \boldsymbol{b}$ の第 j，第 i 成分である．

単体法の反復計算で重要な部分は，最適性の判定と，基底変数と非基底変数の入れ換え，すなわち端点の移動である．これを説明するために以下のような量を定義する．

$\boldsymbol{a}_j, j=1, \cdots, n$: 係数行列 A の第 j 列の要素から成る m 次元ベクトル

K : $\boldsymbol{a}_j$ の中で一次独立なベクトルの添字の集合

TABLE 1 Programs for linear problems

Program	Method
LPM	two phase revised simplex
DUAL	dual simplex
GOALP	goal programming for infeasible and fuzzy constraints
DECOMP	decomposition principle for large scale problems
DUOPLX	duoplex method for many constraints problems
CUTPLN	cutting plane method for convex and integer program
SENSE	sensitivity analysis routine

K' : $j\notin K$ なる添字の集合

B : $\boldsymbol{a}_j$, $j\in K$ なるベクトルから成る行列で基底行列と呼ばれる

N : $\boldsymbol{a}_j$, $j\in K'$ なるベクトルから成る行列で，非基底行列と呼ばれる．

$\boldsymbol{x}_B$: $\boldsymbol{x}$ の要素の中で，$j\in K$ なる x_j から成るベクトルで，基底ベクトルあるいは単体法の反復過程では基底解と呼ばれる．

$\boldsymbol{x}_N$: $\boldsymbol{x}$ の要素の中で，$j\in K'$ なる x_j から成るベクトルで，非基底解と呼ばれる．単体法の反復過程においては $x_j=0$ for $j\in K'$ である．

$\boldsymbol{c}_B$: コスト係数 $\boldsymbol{c}$ の要素の中で $j\in K$ なる c_j から成るベクトル．

$\boldsymbol{c}_N$: コスト係数 $\boldsymbol{c}$ の要素の中で $j\in K'$ なる c_j から成るベクトル．

このような定義の下で問題(3-1)を書き直すと，

$$\left.\begin{aligned} &\text{Minimize}\ F(\boldsymbol{x})=\boldsymbol{c}_B^t\cdot\boldsymbol{x}_B+\boldsymbol{c}_N^t\cdot\boldsymbol{x}_N \\ &\text{subject to} \\ &\quad B\boldsymbol{x}_B+N\boldsymbol{x}_N=\boldsymbol{b} \\ &\quad \boldsymbol{x}_B, \boldsymbol{x}_N\geq 0 \end{aligned}\right\} \qquad (3\text{-}2)$$

となる．(3-2)の制約条件式の基底変数 $\boldsymbol{x}_B$ を非基底変数で表すと，

$$\boldsymbol{x}_B=B^{-1}\boldsymbol{b}-B^{-1}\cdot N\cdot\boldsymbol{x}_N \qquad (3\text{-}3)$$

となり，これを(3-2)の目的関数に代入すると，

$$\begin{aligned} F(\boldsymbol{x}) &= \boldsymbol{c}_B^t[B^{-1}\boldsymbol{b}-B^{-1}N\cdot\boldsymbol{x}_N]+\boldsymbol{c}_N^t\cdot\boldsymbol{x}_N \\ &= \boldsymbol{c}_B^t\cdot B^{-1}\boldsymbol{b}-[\boldsymbol{c}_B^tB^{-1}N-\boldsymbol{c}_N^t]\boldsymbol{x}_N \\ &= \boldsymbol{\pi}\cdot\boldsymbol{b}-\sum_{j\in K'}[\boldsymbol{\pi}\boldsymbol{a}_j-c_j]x_j \end{aligned} \qquad (3\text{-}4)$$

を得る．但し $\boldsymbol{\pi}=\boldsymbol{c}_B^t\cdot B^{-1}$ である．単体法の反復過程では，端点 $\boldsymbol{x}=(\boldsymbol{x}_B\geq 0, \boldsymbol{x}_N=0)$ が(3-1)の最適解になっているか否かは，0になっている非基底変数のいずれかの

要素を正に増加させることにより，目的関数 F の値を更に減少させることができるか否かによる．すなわち(3-4)式から

$$\boldsymbol{\pi a}_j - c_j \leqq 0, \text{ for all } j \in K' \qquad (3\text{-}5\,a)$$

であれば目的関数の改良はできず，そのときの点

$$\boldsymbol{x} = (\boldsymbol{x}_B, \boldsymbol{x}_N)$$

が最適解になり，関数値は(3-4)式で $x_j=0$, for $j\in K'$ とおいた $F(\boldsymbol{x})=\boldsymbol{\pi b}$ により与えられる．逆に，

$$\boldsymbol{\pi a}_j - c_j > 0 \text{ for some } j \in K' \qquad (3\text{-}5\,b)$$

なら，対応する x_j の値を正に増加させることにより目的関数の値をさらに減少できることになる．もし複数個の $j\in K'$ に対して(3-5 b)が成り立っているなら，

$$\max_{j\in K'} \{z_j = \boldsymbol{\pi a}_j - c_j,\ z_j > 0\} \qquad (3\text{-}6)$$

を与える j^* に対応する x_j^* を選ぶことにより目的関数の値を最も改良できるのである．(3-6) 式によって新しく基底の中に入れるべき変数 x_j^* が決定されたが，次に基底の中から追い出す変数を探すことが必要になる．これは，新しく決定される点は実行可能であること，即ち(3-3)式より

$$\begin{aligned} x_{\nu_i} &= \bar{b}_i - \sum_{j\in K'} \overline{a_{ij}} \cdot x_j \\ &= \bar{b}_i - \overline{a_{ij^*}} \cdot x_{j^*} \geq 0, \text{ for } i = 1, \cdots, m \end{aligned} \qquad (3\text{-}7)$$

でなければならない．但し，$x_{\nu_i}, \bar{b}_i, \overline{a_{ij}}$ は，それぞれ $\boldsymbol{x}_B$, $\bar{\boldsymbol{b}}=B^{-1}\cdot\boldsymbol{b}$, $\bar{\boldsymbol{a}}_j=B^{-1}\boldsymbol{a}_j$ の第 i 成分である．したがって，新しく基底に取り入れられる x_{j^*} の増加範囲は

$$\overline{a_{ij^*}} \cdot x_{j^*} \leq \bar{b}_i, \text{ for } i = 1, \cdots, m$$

なる制約を受けるから，

$$\theta = \min_{i=1,\cdots,m} \{\bar{b}_i / \overline{a_{ij^*}},\ \overline{a_{ij^*}} > 0\}$$

を求めて $x_{j^*}=\theta$ とすればよいことになる．

以上が，単体法の反復計算における最適性の判定と端点の移動の概略である．

線形計画法の問題では，問題の規模や制約条件式の性質などによって，単体法をそのまま用いて解くことができない場合がある．そのような問題に対してはあらかじめ問題を操作する必要があるが，TABLE 1 には問題の種類に応じて必要とされる問題操作のプログラムが載せられている．

この中で LPM は線形計画法の基本的な役割りを果たす単体法のプログラムで，2段階法 (two phase method)[10] を採用している．この2段階法というのは，(3-1) の初期基底実行可能解を求めるために，まず各制約条件式に人為変数 (artificial variable) と呼ばれる新しい変数 $y_1, \cdots, y_m$ を導入し，問題を

$$\left.\begin{aligned} &\underset{\boldsymbol{x},\boldsymbol{y}}{\text{Minimize}}\ (y_1 + \cdots + y_m) \\ &\text{subject to} \\ &\quad \sum_{j=1}^{n} a_{ij}x_j + y_i = b_i,\ i = 1, \cdots, m \\ &\quad x_j, y_i \geq 0 \text{ for all } i \text{ and } j \end{aligned}\right\} \qquad (3\text{-}8)$$

に変換し，$\boldsymbol{y}=(y_1, \cdots, y_m)$ を初期基底解として(3-8)を解く第1段階と，その結果得られた最適解を (3-1) の初期基底解として (3-1) を解く第2段階に分けて探索を行なう方法である．そして問題 (3-1) が有界な実行可能領域を持つなら (3-8) の解は (3-1) の初期実行可能解となることが簡単に証明できる．

DUALP は双対単体法 (dual simplex method)[10] を用いたプログラムで，(3-1) の代わりに次のような双対問題

$$\left.\begin{aligned} &\text{Maximize } \boldsymbol{b}^t \cdot \boldsymbol{y} \\ &\text{subject to} \\ &\quad A^t\boldsymbol{y} \leq \boldsymbol{c} \end{aligned}\right\} \qquad (3\text{-}9)$$

を解くプログラムである．このような問題操作は (3-1) の制約条件式の個数が多いときに行なわれるもので，線形計画法の計算時間は制約条件式の数に依存することから，(3-1) の代りに (3-9) を解くことが有効になる場合がある．そして双対定理によって (3-1) が最適解を持つならその双対問題 (3-9) も最適解を持ち，(3-1) の最小値と (3-9) の最大値は一致することが保証されている．但し (3-9) において $\boldsymbol{y}^t=(y_1, \cdots, y_m)$ は双対変数と呼ばれる量である．

GOALP は目標計画法 (goal programming)[12] を用いたプログラムであり，この手法は，制約条件式の中にあいまい性を持つものがあったり，相矛盾する制約式が存在することにより実行可能領域が存在しないような LP 問題を扱うために開発された手法である．LP では制約条件式を全て満たす点の中で所与の目標を最小にする点を探索するのに対し，目標計画法では制約条件式を目標と考え，その目標からのずれを最小にするような点を探索する．また目約関数そのものをも目標と設定することにより多重目標 (multi-criteria) の問題への応用も可能である．以下に目標計画法のアルゴリズムを説明する．

いま問題 (3-1) の代わりに次のような問題を考える．

$$\left.\begin{aligned} &\text{Minimize } z = \sum_{j=1}^{n} c_j x_j \\ &\text{subject to} \\ &\quad \sum_{j=1}^{n} a_{ij}x_j \gtrless b_i,\ \text{for } i \in M_1 \\ &\quad \sum_{j=1}^{n} d_{ij}x_j \gtrless e_i,\ \text{for } i \in M_2 \\ &\quad x_j \geqq 0,\ j = 1, \cdots, n \end{aligned}\right\} \qquad (3\text{-}10)$$

(3-10) の中で $i\in M_1$ に対する制約条件式は上位目標とする．即ち探索の過程でこの制約は常に満たされていなければならないような条件式の集まりである．$i\in M_2$ に対する制約は下位目標とし，これらの制約式に対してはもし満たされなければ，その過不足を最小にしてほしいものとする．このとき (3-10) に対する目標計画法の問題は次のように定式化される．

$$\left.\begin{aligned} &\text{Minimize } z' = y_0^+ - y_0^- + \sum_{i\in M_2} (s_i^+ \cdot y_i^+ + s_i^- \cdot y_j^-) \\ &\text{subject to} \end{aligned}\right.$$

$$\left.\begin{aligned}&\sum_{j=1}^{n} a_{ij}x_j \gtreqless b_i, \quad \text{for } i\in M_1\\&\sum_{j=1}^{n} d_{ij}x_j - e_i = y_i^{+} - y_i^{-}, \quad \text{for } i\in M_2\\&\sum_{j=1}^{n} c_j x_j = y_0^{+} - y_0^{-}\\&\text{and } x_j, y_i^{+}, y_i^{-} \geqq 0\end{aligned}\right\} \tag{3-11}$$

(3-11)において新しく導入された補助変数 y_i^{+} は目標を超過したときの超過の大きさを示し，y_i^{-} は目標に達しないときの不足の大きさを示す変数である．また目的関数の第3項の変数 s_i^{+}, s_i^{-} は次のように定義される．問題(3-11)における下位の制約式（すなわち $i\in M_2$ に対する制約式）の形が

$$\sum_{j=1}^{n} d_{ij}x_j$$

$$\begin{cases}\geqq e_i \text{ なら}, & s_i^{+}=0,\ s_i^{-}=1 \text{ すなわち}\\ & s_i^{+}y_i^{+}+s_i^{-}y_i^{-}=y_i^{-} \qquad (3\text{-}12)\\ = e_i \text{ なら}, & s_i^{+}=1,\ s_i^{-}=1 \text{ すなわち}\\ & s_i^{+}y_i^{+}+s_i^{-}y_i^{-}=y_i^{+}+y_i^{-} \qquad (3\text{-}13)\\ \leqq e_i \text{ なら}, & s_i^{+}=1,\ s_i^{-}=0 \text{ すなわち}\\ & s_i^{+}y_i^{+}+s_i^{-}y_i^{-}=y_i^{+} \qquad (3\text{-}14)\end{cases}$$

と定める．上式からわかるように(3-12)は目標値 e_i に不足することをできるだけ避けたい場合，(3-13)は目標値 e_i にできるだけ近づけたい場合，(3-14)は目標値 e_i の超過をできるだけ避けたい場合である．(3-11)を解いて得られた最適解における y_i^{+}, y_i^{-} の値を参照することにより，設定した目標からどのくらいずれたかを知ることができる．もし $y_i^{+}=0$, $y_i^{-}=0$ であったなら，その制約条件式は満たされたことを意味している．

DECOMP は問題(3-1)の変数の数 n や制約条件式の数 m が多い大規模システムの最適化問題を解くために Dantzig と Wolfe が提唱した分解原理法 (Decomposition Principle)[13] を用いたプログラムで，解析の対象となっているシステムを r 個の部分システムに分割した次のような問題を考える．

$$\left.\begin{aligned}&\text{Minimize } z=\sum_{i=1}^{r} \boldsymbol{c}_i^{t}\cdot\boldsymbol{x}_i\\&\text{subject to}\\&\sum_{i=1}^{r} A_i\boldsymbol{x}_i=\boldsymbol{b} \qquad (\text{R1})\\&B_i\boldsymbol{x}_i=\boldsymbol{b}_i, \quad i=1,2,\cdots,r \qquad (\text{R2})\\&\boldsymbol{x}_i\geqq 0\end{aligned}\right\} \tag{3-15}$$

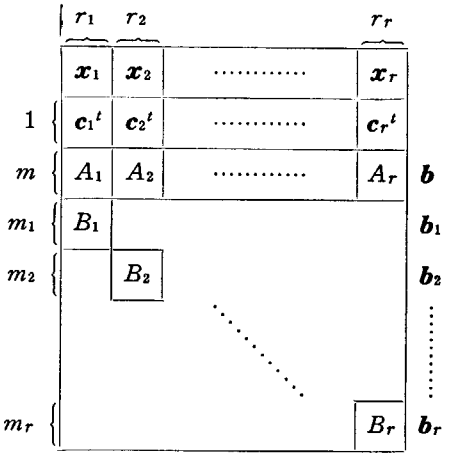
但し

A_i : $(m\times r_i)$ 行列，B_i : $(m_i\times r_i)$ 行列

$\boldsymbol{c}_i$: r_i-ベクトル，$\boldsymbol{x}_i$: r_i-ベクトル

$\boldsymbol{b}$: m-ベクトル，$\boldsymbol{b}_i$: m_i-ベクトル

である．(3-15)をタブロー形式で図示すると，次のようになる．



問題(3-15)の制約条件式において（R1）は分割された部分システム間の相互干渉を表すもので，（R2）は各部分システム内の挙動を表すものである．以下に分解原理の概略を述べる．

$$S_i=\{\boldsymbol{x}_i \mid \boldsymbol{x}_i\geqq 0,\ B_i\boldsymbol{x}_i=\boldsymbol{b}_i\} \tag{3-16}$$

によって凸多面体 S_i, $i=1,\cdots,r$ を定義する．いま各 S_i の全ての端点の集合を

$$W_i=\{\boldsymbol{x}_{i1}, \boldsymbol{x}_{i2}, \cdots, \boldsymbol{x}_{iK_i}\}$$

で表わすと，S_i 内の任意の点はその端点の凸結合で表すことができる．すなわち凸多面体に関する定理により

$$\left.\begin{aligned}&\forall \boldsymbol{x}_i\in S_i;\\&\boldsymbol{x}_i=\sum_{j=1}^{K_i}\alpha_{ij}\boldsymbol{x}_{ij}\\&\sum_{j=1}^{K_j}\alpha_{ij}=1, \quad \alpha_{ij}\geqq 0\end{aligned}\right\} \tag{3-17}$$

である．(3-17)式を用いて問題(3-15)を書き直すと

$$\left.\begin{aligned}&\text{Minimize } z=\sum_{i=1}^{r}\sum_{j=1}^{K_i} c_{ij}\alpha_{ij}\\&\text{subject to}\\&\sum_{i=1}^{r}\sum_{j=1}^{K_i}\boldsymbol{P}_{ij}\alpha_{ij}=\boldsymbol{b}\\&\sum_{j=1}^{K_i}\alpha_{ij}=1 \quad \text{for } i=1,\cdots,r\\&\alpha_{ij}\geqq 0 \quad \text{for all } i \text{ and } j\end{aligned}\right\} \tag{3-18}$$

となる．但し

$c_{ij}=\boldsymbol{c}_i^{t}\boldsymbol{x}_{ij}$ （スカラー）

$\boldsymbol{P}_{ij}=A_i\boldsymbol{x}_{ij}$ （m-ベクトル）

である．(3-18)の制約条件式をタブロー表示すると

$$\begin{matrix}m\\r\end{matrix}\left\{\begin{matrix}(\boldsymbol{P}_{11}\cdots\boldsymbol{P}_{1K_1})(\boldsymbol{P}_{21}\cdots\boldsymbol{P}_{2K_2})\cdots(\boldsymbol{P}_{r1}\cdots\boldsymbol{P}_{rK_r})\\(1\cdots\cdots 1)\\ \qquad (1\cdots\cdots 1)\\ \qquad\qquad \cdots\\ \qquad\qquad\qquad (1\cdots\cdots 1)\end{matrix}\right.\begin{bmatrix}\alpha_{11}\\ \vdots\\ \alpha_{1K_1}\\ \alpha_{21}\\ \vdots\\ \alpha_{2K_2}\\ \vdots\\ \alpha_{r1}\\ \vdots\\ \alpha_{rK_r}\end{bmatrix}$$

$$=\begin{bmatrix} b_1 \\ \vdots \\ b_m \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

となる．また問題(3-15)と(3-18)の間には

「$\alpha_{ij},\ i=1,\cdots,r,\ j=1,\cdots,K_j$ を(3-18)の解とすると

$$\boldsymbol{x}_i=\sum_{j=1}^{K_i}\alpha_{ij}\boldsymbol{x}_{ij},\quad i=1,\cdots,r \tag{3-19}$$

は問題(3-15)の解である」

という関係がある．それ故(3-15)を解く代りに(3-18)を解いて得られた解から(3-19)により原問題の解を求めるとよいことになる．しかしながら現実には各 S_i の全ての端点 $\boldsymbol{x}_{ij}$ $(i=1,\cdots,r,\ j=1,\cdots,K_i)$ が既知であることはないので，次のような方法が用いられる．

問題(3-18)の係数行列を構成する列ベクトル

$$\boldsymbol{P}^i=\begin{bmatrix}\boldsymbol{P}_{ij}\\ \boldsymbol{e}_i\end{bmatrix},\quad i=1,\cdots,\sum_{j=1}^{K_j}$$

は $(m+r)$ 次元のベクトルである．但し上式において $\boldsymbol{e}_i$ は第 i 成分を1とする r 次元単位ベクトルである．$m+r$ 次元ベクトル空間において一次独立なベクトルの個数はたかだか $(m+r)$ 個である．いま $\boldsymbol{P}^i$ の中で一次独立なベクトルから成る行列（基底行列）を $\boldsymbol{P}_B$ とし，そのときの列ベクトルの添字の集合を B とする．また残りの列ベクトルから成る行列（非基底行列）を $\boldsymbol{P}_N$ とし，添字の集合を N とする．このとき問題(3-17)の最適性の条件は，

$$\boldsymbol{c}_B{}^t\boldsymbol{P}_B{}^{-1}\cdot\boldsymbol{P}_N-\boldsymbol{c}_N\leqq 0 \tag{3-20}$$

となる．但し $\boldsymbol{c}_B$ はコスト係数 c_{ij} の中で $i,j\in B$ から成るベクトル，$\boldsymbol{c}_N$ は $i,j\in N$ から成るベクトルである．(3-20)式において評価ベクトル(pricing vector) $\boldsymbol{c}_B{}^t\cdot\boldsymbol{P}_B{}^{-1}$ を

$$(\boldsymbol{\pi},\tilde{\boldsymbol{\pi}})=\boldsymbol{c}_B{}^t\cdot\boldsymbol{P}_B{}^{-1}$$

で表す．但し $\boldsymbol{\pi}$ は m 次元，$\tilde{\boldsymbol{\pi}}$ は r 次元ベクトル，$\boldsymbol{\pi}$ と $\tilde{\boldsymbol{\pi}}$ を成分とするベクトルを $(\boldsymbol{\pi},\tilde{\boldsymbol{\pi}})$ とする．この時(3-20)は

$$(\boldsymbol{\pi},\tilde{\boldsymbol{\pi}})\boldsymbol{P}_N-\boldsymbol{c}_N\leqq 0 \tag{3-21}$$

となる．$\boldsymbol{P}_N$ は列ベクトル $(\boldsymbol{P}_{ij},\boldsymbol{e}_i)^t,\ i,j\in N$ から成る行列であったから(3-21)は

$$\boldsymbol{\pi}\boldsymbol{P}_{ij}+\tilde{\pi}_i-c_{ij}\leqq 0\ \text{ for all } i,j \tag{3-22}$$

となる．但し $\tilde{\pi}_i$ はベクトル $\boldsymbol{\pi}$ の第 i 成分である．上の(3-22)式は次式と同値である．

$$\max_{i,j}(\boldsymbol{\pi}\boldsymbol{P}_{ij}+\tilde{\pi}_i-c_{ij})\leqq 0 \tag{3-23}$$

さきに定義した関係式

$$\boldsymbol{P}_{ij}=A_i\boldsymbol{x}_{ij}$$

$$c_{ij}=\boldsymbol{c}_i{}^t\boldsymbol{x}_{ij}$$

を用いると(3-23)は

$$\max_{i,j}\{(\boldsymbol{\pi}A_i-\boldsymbol{c}_i{}^t)\boldsymbol{x}_{ij}+\tilde{\pi}_i\}\leqq 0 \tag{3-24}$$

となる．(3-24)式の中の $\boldsymbol{x}_{ij}$ は凸多面体 $S_i=\{\boldsymbol{x}_i\,|\,\boldsymbol{x}_i\geqq 0;\ B_i\boldsymbol{x}_i=\boldsymbol{b}_i\}$ の端点であったが，S_i 内での超平面 $(\boldsymbol{\pi}A_i-\boldsymbol{c}_i{}^t)\boldsymbol{x}_{ij}$ の最大値は S_i の端点でとられるから(3-24)式は次のような LP 問題に環元される．

$$\left.\begin{aligned}&\max\ (\boldsymbol{\pi}A_i-\boldsymbol{c}_i{}^t)\boldsymbol{x}_i+\tilde{\pi}_i\\ &\text{subject to}\\ &\quad B_i\boldsymbol{x}_i=\boldsymbol{b}_i\\ &\quad \boldsymbol{x}_i\geqq 0\end{aligned}\right\}i=1,\cdots,r \tag{3-25}$$

(3-25)は分割された各部分システム内での最適化問題であり，これを全ての $i=1,\cdots,r$ について解き，

$$\gamma_s=\min_{i=1,\cdots,r}\{\gamma_i=(\boldsymbol{c}_i-\boldsymbol{\pi}A_i)\boldsymbol{x}_i{}^*\}\geqq 0$$

なら最適性の基準(3-20)が満たされていることになるから，問題(3-18)の最適解はそのときの基底逆行列

$$\boldsymbol{P}_B{}^{-1}=\begin{bmatrix}\boldsymbol{P}_{ij}\\ \boldsymbol{e}_i\end{bmatrix}^{-1} \tag{3-26}$$

を用いて，$\boldsymbol{I}$ を r 個の成分1を持つ単位ベクトルとして，

$$\boldsymbol{\alpha}_B=\boldsymbol{P}_B{}^{-1}\begin{bmatrix}\boldsymbol{b}\\ \boldsymbol{I}\end{bmatrix}$$

により求める．但し，(3-26)の $\boldsymbol{P}_{ij}$ は基底変数に対応する列ベクトルの集まりで，$\boldsymbol{e}_i$ は第 i 成分が1である単位ベクトルである．もし $\gamma_s<0$ なら，基底解の更新と評価ベクトルの更新を行なう単体法の反復計算をくり返す．以下に分解原理法によるプログラム DECOMP の流れ図を示す．

1. 分割されたサブシステム $S_i=\{\boldsymbol{x}\,|\,B_i\boldsymbol{x}=\boldsymbol{b}_i,\ \boldsymbol{x}\geqq 0\}$ より $m+r$ 個の端点 $\boldsymbol{x}_{j,k}$ $(j=1,\cdots,r,\ k=k_1,\cdots,k_m)$ を選ぶ．そのときのインデックスの集まりを B とする．

↓ $\boldsymbol{x}_{j,k},\ B$

2. $\left.\begin{aligned}\boldsymbol{P}_{jk}&=A_j\boldsymbol{x}_{jk}\\ c_{jk}&=\boldsymbol{c}_j\boldsymbol{x}_{jk}\end{aligned}\right\}\ j=1,\cdots,r,\ k=k_1,\cdots,k_m$

を計算し，評価ベクトルを

$$(\boldsymbol{\pi},\tilde{\boldsymbol{\pi}})=(c_{jk})\begin{bmatrix}\boldsymbol{P}_{jk}\\ \boldsymbol{e}_j\end{bmatrix}^{-1}$$

により求める．

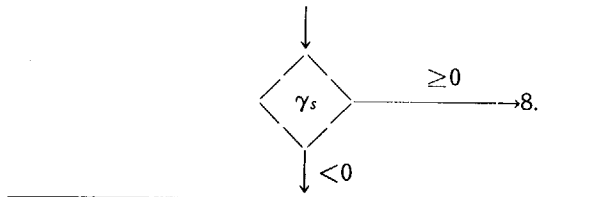
↓ $(\boldsymbol{\pi},\tilde{\boldsymbol{\pi}})$

3. $\left.\begin{aligned}&\min\gamma_j=(\boldsymbol{c}_j-\boldsymbol{\pi}A_j)\boldsymbol{x}_j-\tilde{\pi}_j\\ &\text{subject to}\\ &\quad B_j\boldsymbol{x}_j=\boldsymbol{b}_j\\ &\quad \boldsymbol{x}_j\geqq 0\end{aligned}\right\}\ j=1,\cdots,r$

を解き解を $\boldsymbol{x}_j{}^*$ とする

↓ $\boldsymbol{x}_j{}^*,\gamma_j$

4. $\gamma_s=\min_j\{\gamma_j=(\boldsymbol{c}_j-\boldsymbol{\pi}A_j)\boldsymbol{x}_j{}^*-\tilde{\pi}_j\}$



5. $\boldsymbol{P}_s=\boldsymbol{P}_{is}=A_i\boldsymbol{x}_s^*$

$c_s=\boldsymbol{c}_i\boldsymbol{x}_s^*$

$$\boldsymbol{a}=\begin{bmatrix}\boldsymbol{P}_{ik},\ i,\ k\in B\\ \boldsymbol{e}_i\end{bmatrix}^{-1}\begin{bmatrix}\boldsymbol{b}\\ \boldsymbol{I}\end{bmatrix}$$

$$\boldsymbol{a}^s=\begin{bmatrix}\boldsymbol{P}_{ik},\ i,\ k\in B\\ \boldsymbol{e}_i\end{bmatrix}^{-1}\begin{bmatrix}\boldsymbol{P}_{is}\\ \boldsymbol{e}_i\end{bmatrix}$$

$$\frac{\alpha_q}{\alpha_q{}^s}=\min_l\left\{\frac{\alpha_l}{\alpha_l{}^s},\quad \text{for } \alpha_l{}^s>0\right\}$$

↓ $\boldsymbol{P}_s, c_s, q$

6. 基底行列 $\begin{bmatrix}\boldsymbol{P}_{ik},\ i,\ k\in B\\ \boldsymbol{e}_j\end{bmatrix}$ の第 q 列目を $\begin{bmatrix}\boldsymbol{P}_s\\ \boldsymbol{e}_s\end{bmatrix}$ で置き換える.

基底解 $\boldsymbol{x}_B=(\boldsymbol{x}_{ik}), i,k\in B$ の第 q 番目を $\boldsymbol{x}_s$ で置き換える. (c_{ik}) の q 番目を c_s で置き換える.

↓

7. 新しい評価ベクトルを

$$(\boldsymbol{\pi},\tilde{\boldsymbol{\pi}})=(c_{ik})\begin{bmatrix}\boldsymbol{P}_{ik}\\ \boldsymbol{e}_i\end{bmatrix}^{-1}\ i,k\in B$$

により求める.

↓ $(\boldsymbol{\pi},\tilde{\boldsymbol{\pi}})$

3. へ行く

8. 最適性の基準が満たされたので(3-11)の解を

$$\boldsymbol{a}=\begin{bmatrix}\boldsymbol{P}_{ik},\ i,k\in B\\ \boldsymbol{e}_i\end{bmatrix}^{-1}\begin{bmatrix}\boldsymbol{b}\\ \boldsymbol{I}\end{bmatrix}$$

により求める.

↓ $\boldsymbol{a}$

9. (3-8)の解を

$$\boldsymbol{x}_i=\sum_{j=1}^{k_i}\alpha_{ij}\boldsymbol{x}_{ij},\quad i=1,\cdots,r$$

により求める.

DUOPLX は，分解原理法が変数の数と制約条件式の個数が共に多い問題に対する手法であるのに対し，制約条件式だけが多い問題の解を効率的に探索する duoplex 法[16]を用いたプログラムで，求める最適点が凸多面体を構成している超平面の中のどの超平面上に存在するかを探索する.

CUTPLN は，制約領域が凸多面体ではなく凸領域になるような凸計画法の問題や，変数の全て，あるいは一部に整数制約が課せられる整数計画問題を扱うために切除平面法 (cutting plane method)[14] を用いたプログラムである．整数計画問題に CUTPLN の適用を説明するため，(3-1)の制約領域を S で表し，R を次のように定義する.

$$R=\{\boldsymbol{x}\,|\,\boldsymbol{x}\in S,\ \boldsymbol{x}:\text{integer}\}\qquad(3\text{-}27)$$

この定義から明らかなように

$$R\subset S=\{\boldsymbol{x}\,|\,A\boldsymbol{x}=\boldsymbol{b},\ \boldsymbol{x}\geq 0\}$$

である．このとき，切除平面法の計算手順は以下のようになる．(Fig. 1 参照)

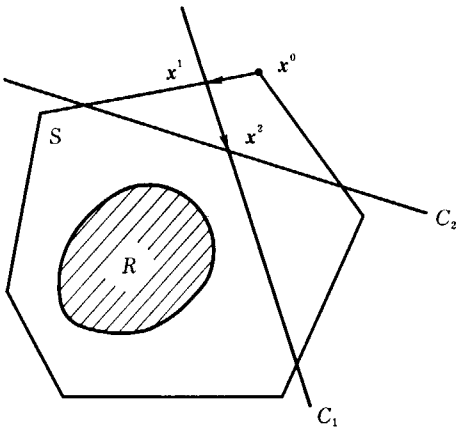


Fig. 1 Generation of a cut c_k to separate the point x^k from R

(i) $k=0$ とし $S^k=S$ とする

(ii) S^k の中で

Minimize $\boldsymbol{c}^t\cdot\boldsymbol{x}$

の解 $\boldsymbol{x}^k$ を求める

(iii) もし $\boldsymbol{x}^k\in R$ なら $\boldsymbol{x}^k$ は求める最適解である.

(iv) もし $\boldsymbol{x}^k\notin R$ なら点 $\boldsymbol{x}^k$ と R を分離する切除平面

$$c_k=\boldsymbol{a}^k\cdot\boldsymbol{x}\qquad(3\text{-}28)$$

を作り $S^{k+1}=\{\boldsymbol{x}\,|\,\boldsymbol{x}\in S^k, \boldsymbol{a}^k\boldsymbol{x}=0\}$ とし $k=k+1$ として(ii)へ戻る.

上記のようにして生成される領域 S^k は,

$$S^0\supset S^1\supset S^2\supset\cdots\supset R$$

であり，探索を行うべき領域 R に限りなく近づいていく．(3-28)式の切除平面の作成の方法については，例えば Gomory[15] などが提唱した方法がある.

SENSE は，LP 計算によって最適解が求められているとき，その解の安定領域を調べたり，その安定領域の中で右辺要素 $\boldsymbol{b}$ や目的関数の係数 $\boldsymbol{c}$ の各成分をどのぐらい変化させることができるかを調べるものであり，このような解析を行なうことにより，例えばエネルギーシステム解析の問題では右辺要素に現れる原子力，石油などの供給制約量や，それらの需要量の単位量当たりの増減が最適解にどのような影響を与えるかを調べることができる.

右辺要素 $\boldsymbol{b}$ に対する感度解析を説明するためにいま(3-1)の最適解 $\boldsymbol{x}^*$ が得られているものとする．このとき，$\boldsymbol{x}^*$ は基底解 $\boldsymbol{x}_B^*$ と非基底解 $\boldsymbol{x}_N^*$ とに分かれており，

$$\boldsymbol{x}_B^*=B^{-1}\boldsymbol{b}\geqq 0,\quad \boldsymbol{x}_N^*=0$$
$$\boldsymbol{c}_B{}^tB^{-1}N-\boldsymbol{c}_N{}^t\leqq 0 \qquad (3\text{-}29)$$

が成り立っている．但し，B^{-1} は基底逆行列で，$\boldsymbol{c}_B, \boldsymbol{c}_N$ はそれぞれ基底解，非基底解に対応するコストベクトルである．(3-29)式は問題 (3-1) に対する最適性の条件である．このとき，目的関数の値は

$$F(\boldsymbol{x}^*)=\boldsymbol{c}_B{}^t\boldsymbol{x}_B^*=\boldsymbol{c}_B{}^t\cdot B^{-1}\boldsymbol{b}$$

である．いま右辺ベクトル $\boldsymbol{b}$ の第 r 成分 b_r を Δb_r だけ変化させた新しい右辺ベクトルを $\hat{\boldsymbol{b}}$ とする．すなわち，

$$\hat{\boldsymbol{b}}=\boldsymbol{b}+\Delta\boldsymbol{b}=\begin{pmatrix} b_1 \\ \vdots \\ b_r \\ \vdots \\ b_m \end{pmatrix}+\begin{pmatrix} 0 \\ \vdots \\ \Delta b_r \\ \vdots \\ 0 \end{pmatrix}$$

とする．このとき，新しい基底解 $\hat{\boldsymbol{x}}_B^*$ と目的関数値 $\hat{F}$ は，

$$\hat{\boldsymbol{x}}_B^*=B^{-1}\hat{\boldsymbol{b}}=B^{-1}\boldsymbol{b}+B^{-1}\Delta\boldsymbol{b}$$
$$=\boldsymbol{x}_B^*+B^{-1}\Delta\boldsymbol{b} \qquad (3\text{-}30)$$
$$\hat{F}=\boldsymbol{c}_B{}^tB^{-1}\hat{\boldsymbol{b}}$$
$$=F(\boldsymbol{x}^*)+\boldsymbol{c}_B{}^tB^{-1}\cdot\Delta\boldsymbol{b} \qquad (3\text{-}31)$$

となる．最適性の条件(3-29)式には右辺ベクトル $\boldsymbol{b}$ が含まれていないから，右辺ベクトルをどのように変化させても(3-29)式は成り立っている．したがって，もし $\hat{\boldsymbol{x}}_B^*\geqq 0$ であれば，この $\hat{\boldsymbol{x}}_B^*$ が新しい問題の最適解に成る．そこで $\hat{\boldsymbol{x}}_B^*\geqq 0$ になるような Δb_r の範囲を求めると(3-30)式より，

$$\hat{x}_i^*=x_i^*+\sum_{j=1}^{m} b_{ij}\Delta b_j,\quad i=1,\cdots,m \qquad (3\text{-}32)$$

である．但し，$\hat{x}_i^*, x_i^*$ はそれぞれ $\hat{\boldsymbol{x}}_B^*, \boldsymbol{x}_B^*$ の第 i 成分，b_{ij} は B^{-1} の (i, j) 成分である．$j\neq r$ に対しては $\Delta b_j=0$ であるから(3-32)は，

$$\hat{x}_i^*=x_i^*+b_{ir}\cdot\Delta b_r,\quad i=1,\cdots,m \qquad (3\text{-}33)$$

となり，したがって全ての i について $\hat{x}_i^*\geqq 0$ であるためには，

$$\underline{\Delta b_r}\leqq\Delta b_r\leqq\overline{\Delta b_r} \qquad (3\text{-}34)$$

であることが必要である．但し(3-34)の $\underline{\Delta b_r}, \overline{\Delta b_r}$ は

$$\underline{\Delta b_r}=\begin{cases}\max\left\{-\dfrac{x_i^*}{b_{ir}}\right\}, & \text{for } b_{ir}>0\\ -\infty, & \text{if all } b_{ir}\leqq 0\end{cases}$$

$$\overline{\Delta b_r}=\begin{cases}\min\left\{-\dfrac{x_i^*}{b_{ir}}\right\}, & \text{for } b_{ir}<0\\ +\infty, & \text{if all } b_{ir}\geqq 0\end{cases}$$

によって計算される量である．したがって，変化する右辺ベクトルの要素 b_r の量 Δb_r が不等式(3-34)の中にあるなら基底解と非基底解の入れ換えを行なわずに（すなわち新しい端点の探索を行なう必要なしに）新しい最適解を(3-33)式によって求めることができる．また(3-34)による計算をすべての b_i, $i=1,\cdots,m$ に対して行なっておくことにより，各 b_i に対する最適解 $\boldsymbol{x}_B^*$ の安定領域を調べることができる．さらに(3-31)式の

$$\boldsymbol{\pi}=\boldsymbol{c}_B{}^tB^{-1} \qquad (3\text{-}25)$$

を調べることによって，変化する右辺要素の単位量が新しいコストに与える影響を調べることができる．この意味で，(3-35)で定義されるベクトル $\boldsymbol{\pi}$ は評価ベクトルと呼ばれているのである．

TABLE 2 Computational result for Wilde's problem by CUTPLN-DUAL

ITN	$F(x_1, x_2)$	x_1, x_2
1	−148.413	2.0 , 0
2	− 38.422	1.5421, 0.1684
3	− 26.505	1.3768, 0.2293
4	− 24.988	1.3476, 0.240
5	− 23.741	1.3589, 0.2569
6	− 23.726	1.3585, 0.2570
7	− 23.722	1.3585, 0.2571

ITN: Number of iterations

なお LPM と SENSE は，既に日本原子力研究所動力炉開発・安全性研究管理部核エネルギーシステム研究室において開発された核エネルギーシステム解析用プログラム JALTES や，総合研究開発機構の原子力エネルギー長期戦略委員会において開発された動的エネルギーシステム解析用プログラムの中に組みこまれ使用されており，その有効性が実証されている．また TABLE 2 には，第4章で紹介する Wilde の問題を切除平面法のプログラム CUTPLN によって線形化し，双対単体法 DUAL を用いて解いた結果が示されている．表に示すとおり，7回の反復計算でこの問題の最適解 $(\boldsymbol{x}_1^*, \boldsymbol{x}_2^*)=(1.3586, 0.2571)$ の良い近似解が得られており，CUTPLN と DUAL の有効性がわかる．次に目標計画法のプログラム GOALP の実用性を調べるため次のような問題を設定した．

Minimize $z=-x_1-x_2$

subject to

$$\left.\begin{aligned} g_1&=3x_1+5x_2-30\leqq 0\\ g_2&=x_1+x_2-2\quad\geqq 0\\ g_3&=x_1-x_2+5\quad\leqq 0\\ g_4&=x_2-4\quad\leqq 0\end{aligned}\right\} \qquad (3\text{-}36)$$

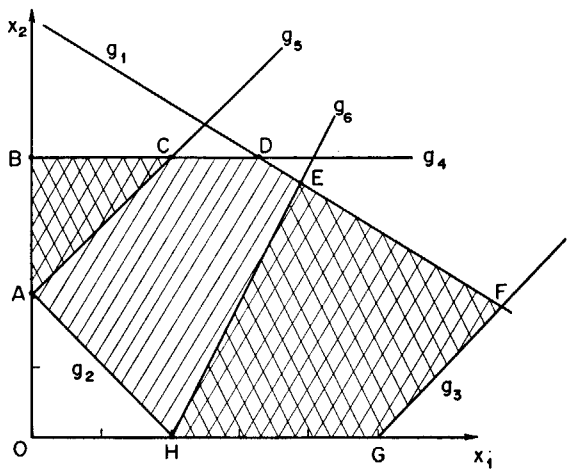


Fig. 2 Figure of infeasible domain

$$\left.\begin{aligned} g_5 &= -x_1+x_2-2 \geqq 0 \\ g_6 &= 2x_1-x_2-4 \geqq 0 \end{aligned}\right\} \quad (3\text{-}37)$$

Fig. 2 からわかるように，制約条件(3-36)によって作られる領域 ABDFGH と(3-37)を同時に満たす領域は存在せず，実行不可能な問題であり，通常の単体法を用いて解くことはできない．そこで目標計画法の問題として解くために，制約条件(3-36)を上位目標とし(3-37)を下位目標として，(3-11)の定式化にしたがって次の問題に変換する．

Minimize $y_0^+ - y_0^- + (y_1^- + y_2^-)$

subject to

$g_1 \leqq 0,\ g_2 \geqq 0,\ g_3 \leqq 0,\ g_4 \leqq 0,$

and

$$\left.\begin{aligned} -x_1-x_2 &= y_0^+ - y_0^- \\ -x_1+x_2-2 &= y_1^+ - y_1^- \\ 2x_1-x_2-4 &= y_2^+ - y_2^- \end{aligned}\right\} \quad (3\text{-}38)$$

GOALP による上記のような変換を経て LPM を用いて解いた結果，

$x_1 = 3.846,\ x_2 = 3.692$

$y_0^+ = 0.,\ y_0^- = 11.231$

$y_1^+ = 0.,\ y_1^- = 2.154$

$y_2^+ = 0.,\ y_2^- = 0.$

が求まった．すなわち，Fig. 2 の E 点がこの問題の満足解として探索され，$(y_1^+, y_1^-) = (0, 2.154)$ であるから下位目標 (3-38) の最初の制約式に対しては 2.154 だけ不足し，$(y_2^+, y_2^-) = (0, 0)$ から 2 番目の制約式は満たされたことがわかる．

線形問題に対する TABLE 1 で示される 7 個のプログラムの中で，LPM, DUAL, GOALP, DECOMP, CUTPLN, SENSE の 6 個は著者が作成したものであり，DUOPLX は参考文献 16) の中のプログラムを整備したものである．

4. 非線形問題に対する最適化手法

4.1 制約条件を持つ問題に対する手法

目的関数 $F(\boldsymbol{x})$ や制約条件式が変数 $\boldsymbol{x}^t=(x_1,\cdots,x_n)$ に関しての非線形関数の場合には，線形の場合と異なり難かしい問題となる．この理由は，線形の場合には制約領域が凸多面体となり目的関数は超平面になっているため，最適点の探索は凸多面体内部の全ての点について行なう必要がなく，有限個しかない端点についてだけ考慮するだけで良かったが，非線形になると，一般には制約領域や目的関数の形状が複雑になり，したがって最適点の探索には制約領域内の全ての点について行なう必要があることによる．したがって，解法としても線形問題の単体法のような万能なものはなく，問題に応じたいろいろな手法が開発されている．それらの開発された手法は大きく分けて発見的手法 (heuristic method) によるもの，変換法あるいはペナルティ法 (penalty function method) と線形近似 (linear approximation) によるものに分類される．TABLE 3 には，この分類に基づいて我々が開発，整備したプログラムが載せられている．

発見的方法では制約領域内に1つの幾何形を考え，その幾何形上に生成された十分多くの点における関数値を評価しながら探索をくり返すもので，我々が開発したプログラムは SIMPLEX, COMPLX, CORASE である．

SIMPLEX は次節で紹介する Nelder と Mead により提唱された simplex 法を flexible tolerance 法[15]に組み入れて作られたプログラムである．

いま非線計画法の一般的な問題として次の形を考える．

$$\left.\begin{aligned}&\text{Minimize } F(\boldsymbol{x}),\ \boldsymbol{x}\in E^n\\&\text{subject to}\\&\quad h_i(\boldsymbol{x})=0,\ i=1\cdots m\\&\quad g_i(\boldsymbol{x})\geqq 0,\ i=m+1\cdots p\end{aligned}\right\}\qquad (4\text{-}1)$$

flexible tolerance 法の特徴は，near-feasible という概念を導入することによりゆるめられた問題を扱い，反復の進行と共に near-feasibility の制限を強めていってもとの制約領域の中の最適点を探索しようというもので，このため以下のような許容基準 Φ と制約違反の基準 $T(\boldsymbol{x})$ を定義する．

$\Phi^{(K)}$: E^n 内に張られた単体の各端点の関数で正値減少関数とする．

例えば次のように定められる．

$$\left.\begin{aligned}&\Phi^{(K)}=\min\{\Phi^{(K-1)},\ \theta^{(K)}\}\\&\Phi^{(0)}=2(m+1)t\end{aligned}\right\}\qquad (4\text{-}2)$$

TABLE 3 Programs for constrained nonlinear problems

Program	Method
SIMPLEX	flexible tolerance-simplex method (heuristic)
COMPLX	complex method (heuristic)
CORASE	controlled random search (heuristic)
MAP	method of approximation program (linear appr.)
CUTPLN	cutting plane (linear appr.)
AJM	modified Allran and Johnsen's method (penalty)
PWM	Powell's method (penalty)
QBEAL	quadratic programming by Beale
QWOLF	quadratic programming by Wolfe
KEELE	nonlinear objective with linear constraints

但し，t は最初の単体の大きさ，$\theta^{(K)}$ は各端点から単体の重心への距離 (L_2 ノルムの意味での) 平均値である．また K は反復の過程を示すインデックスである．単体法では反復計算の過程での単体は，求める最適点へと動かされ小さくなっていくので，$\theta^{(K)}$ は減少していく．したがって(4-2)式で定義される $\Phi^{(K)}$ は

$$\lim_{\boldsymbol{x}\to\boldsymbol{x}^*}\Phi^{(K)}=0$$

である．但し $\boldsymbol{x}^*$ は求める最適点である．

$T(\boldsymbol{x})$:

$$T(\boldsymbol{x})=+\left[\sum_{i=1}^{m}h_i^2(\boldsymbol{x})+\sum_{i=m+1}^{p}u_i\,g_i^2(\boldsymbol{x})\right]^{1/2}\qquad (4\text{-}3)$$

但し，u_i は Heaviside の演算子で，$g_i(\boldsymbol{x})\geqq 0$ に対しては $u_i=0$, $g_i(\boldsymbol{x})<0$ に対しては $u_i=1$ をとる．(4-2), (4-3) により定義された $\Phi^{(K)}$, $T(\boldsymbol{x})$ を用いて near-feasibility の概念は次のように定義される．

$\Phi^{(K)}$ を K 番目の反復過程での Φ の値とし，$\boldsymbol{x}^{(K)}$ を E^n 内の任意のベクトルとする．このとき $\boldsymbol{x}^{(K)}$ は，

1. もし $T(\boldsymbol{x}^{(K)})=0$ なら feasible
2. もし $0\leq T(\boldsymbol{x}^{(K)})\leq\Phi^{(K)}$ なら near-feasible
3. もし $T(\boldsymbol{x}^{(K)})>\Phi^{(K)}$ なら infeasible

という．したがって near-feasibility の領域は

$$\Phi^{(K)}-T(\boldsymbol{x}^{(K)})\geq 0$$

により定められる．

上述のような概念の導入により，問題 (4-1) は次のような問題に変換される．

$$\left.\begin{aligned}&\text{Minimize } F(\boldsymbol{x}),\ \boldsymbol{x}\in E^n\\&\text{subject to}\\&\quad \Phi^{(K)}-T(\boldsymbol{x})\geq 0\end{aligned}\right\}\qquad (4\text{-}4)$$

このような，問題 (4-1) の代りに (4-4) を解く flexible tolerance 法の利点は，

(i) 反復の進行と共に constraint violation の度合

いを減らすことができること．したがって全体的な計算量を減らすことができる．

（ii） $\Phi^{(K)}$ を収束の許容量と定めることができる．すなわち，あらかじめ設定した十分小さな量 ε に対して

$$\Phi^{(K)} \leq \varepsilon$$

なら反復を終息させる．これは探索を行なってきた単体の大きさが半径 ε の超球面の中に含まれたことを意味している．

COMPLX は Box により紹介されたコンプレックス法[17]によるプログラムで，単体法では幾何形として単体を生成するのに対し，この方法では実行可能領域の中に複体 (complex) を生成して探索を行なうものである．

CORASE は，Price により紹介された制御ランダム法 (controlled random search procedure)[18] を用いたプログラムで，従来のランダム探索法と単体法を組み合わせた手法と考えられる．**Fig. 3** に制御ランダム法のアルゴリズムの流れ図を載せる．

図に示すように，Price は生成した $n+1$ 個の端点を持つ単体の極 (pole) として常に $n+1$ 番目の点 $\boldsymbol{x}^{n+1}$ を定め，これを重心に関して折り返すことにより単体を最適点に移動させているが，著者は，その収束を加速させるため，極として定める点を $n+1$ 個の端点の中で目的関数の値を最大にする点をとるように修正した．この意味は，目的関数値を最大にする端点の重心に関する折り返し点は目的関数を最も減小させる点であると期待できるからである．なお，この修正によって 4.2 節でふれるように，計算時間をほぼ 2/3 に短縮することができた．

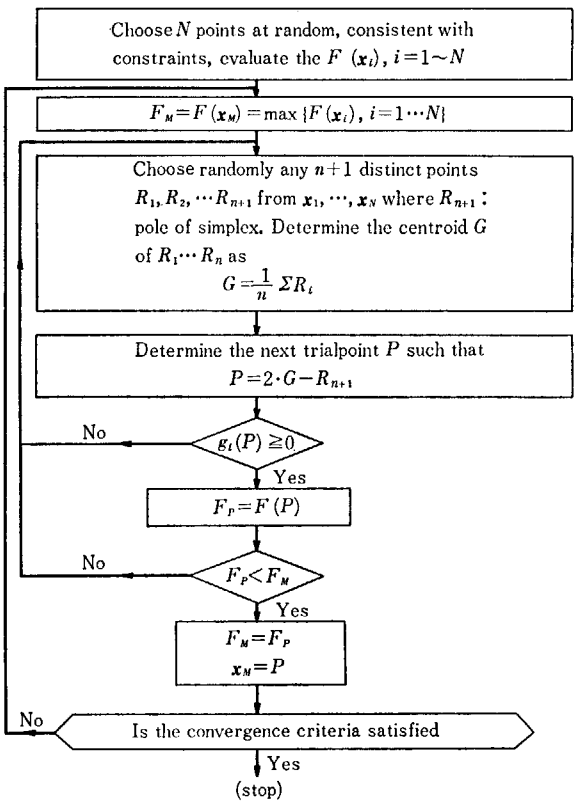


Fig. 3 Flow diagram of Controlled Random Search.

なお，**Fig. 3** の収束判定条件としては，反復回数があらかじめ設定した回数に達したか否かと，複体の各端点 $\boldsymbol{x}^i$ 上での目的関数値の重心 $\boldsymbol{G}$ における値からの標準偏差が十分小さくなったか否か，すなわち

$$\left\{\frac{1}{n}\sum_{i=1}^{n+1}(F(\boldsymbol{x}^i)-F(\boldsymbol{G}))^2\right\}^{1/2} \leq \varepsilon$$

を採用した．また入力量として要求される stored point の数 N は目的関数や制約条件式の複雑さや，変数の数に依存して決められるべきだが，我々の使用経験では変数の数の 20～25 倍にとるのが妥当と思われる．

線形近似法は，各反復ごとに線形近似した問題を作り，線形計画法の問題に帰着させて探索を行なうもので，我々が開発したプログラムは CUTPLN と MAP である．

切除平面法によるプログラム CUTPLN は，既に第3章で説明されている．MAP は，Griffith と Stewart により紹介された近似計画法[19]によるプログラムで，目的関数と制約条件式を各反復の過程で得られる点の回りで Taylor 展開し，2次以上の項を無視することにより，その点の近傍で局所的に線形化された問題を LP を用いて解くものである．すなわち，いま k 番目の反復において $\boldsymbol{x}^k$ が求まっているものとするとき，(4-1) の代わりに解かれる線形問題は

$$\left.\begin{aligned}&\text{Minimize } F(\boldsymbol{x}^k)+(\nabla F(\boldsymbol{x}^k))^t\cdot(\boldsymbol{x}-\boldsymbol{x}^k)\\&\text{subject to}\\&g_j(\boldsymbol{x}^k)+(\nabla g_j(\boldsymbol{x}^k))^t\cdot(\boldsymbol{x}-\boldsymbol{x}^k)\geq 0,\quad j\in M_1\\&h_j(\boldsymbol{x}^k)+(\nabla h_j(\boldsymbol{x}^k))^t\cdot(\boldsymbol{x}-\boldsymbol{x}^k)=0,\quad j\in M_2\\&|x_i-x_i^k|\leq\delta_i^k,\quad i=1\cdots n\end{aligned}\right\}\quad (4\text{-}5)$$

である．但し，記号 $\nabla F(\boldsymbol{x}^k)$ は点 $\boldsymbol{x}^k$ における F の傾斜ベクトルを意味する．

(4-5)式において，最後の制約式 $|x_i-x_i^k|\leq\delta_i^k$ は，接平面によって生成された近似領域の有界性を保証するために設けられたもので，変数 x_i の変域を制限する制約である．この δ_i^k は，反復の進行とともに徐々に小さくされるが，最初の刻み幅 δ_i^0 をどのように定めるかははっきりしていない．大切なことは，初めは実行可能領域が存在するほどに十分大きく定めることである．**Fig. 4** に MAP の流れ図が示されている．

制約条件を持つ問題に対する非線形最適化手法には，前述の発見的方法と線形近似によるものの他に変換法あるいはペナルテイ法と呼ばれるものがある．これは，制約条件式を目的関数の中に組み込んだ変換関数を考え，それを最小にする問題に帰着させるもので，広範囲に研究開発されている制約条件無しの問題に対する最適化手法を利用できるという利点がある．変換関数をどのように定義するかによって種々のペナルティ法が紹介されている．例えば，Fiacco と McCormick による SUMT 法 (Sequential Unconstrained Minimization Technique) などが良く知られている．この方法では，問題 (4-1) を適当な条件の下で次のような問題に変換する．すなわち，

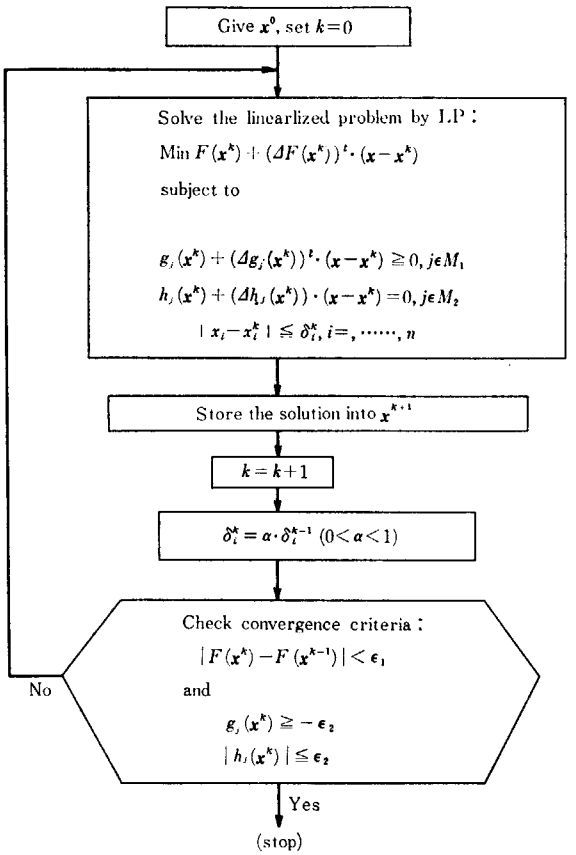


Fig. 4 Flow diagram of MAP.

Minimize

$$P(\boldsymbol{x}, r_k)=F(\boldsymbol{x})+r_k\sum_j[g_j(\boldsymbol{x})]^{-1}+r_k^{-1/2}\sum_i(h_i(\boldsymbol{x}))^2 \tag{4.6}$$

但し r_k は，$0<r_{k+1}<r_k$ で $k\to\infty$ のとき $r_k\to 0$ なる正値減少数列である．

(4-6)では，r^{-1}，$[g_j(\boldsymbol{x})]^{-1}$ という項を含んでいるため，これらが大きな値をとるようになると，反復計算を行なっても目的関数の値が改良されないという欠点がある．PWM が採用した Powell の方法はこの点を改良した手法で，問題 (4-1) において等号制約条件式だけを考えた問題

Minimize $F(\boldsymbol{x})$, $\boldsymbol{x}\in E^n$

subject to

$h_i(\boldsymbol{x})=0$, $i=1\cdots m$

を解くため次のような変換関数を考える．

$$P(\boldsymbol{x}, \boldsymbol{l}, \boldsymbol{s})=F(\boldsymbol{x})+\sum_{i=1}^m l_i(h_i(\boldsymbol{x})+s_i) \tag{4-7}$$

このときアルゴリズムの流れは次のようになる．

(i) $\boldsymbol{l}^0=(1,\cdots,1)^t$, $\boldsymbol{s}^0=(0,\cdots,0)^t$, $k=0$ として，

Minimize $P(\boldsymbol{x}, \boldsymbol{l}^k, \boldsymbol{s}^k)$

を解き，解を $\hat{\boldsymbol{x}}^k$ とおく

(ii) $|h_i(\hat{\boldsymbol{x}}^k)|\leqq\varepsilon$, for all i

なら計算を終了する．

(iii) 収束の速さが速いとき

$s_i^{k+1}=s_i^k+h_i(\hat{\boldsymbol{x}}^k)$, $i=1\cdots m$

とし，$k=k+1$ として(i)にもどる．

遅いときは $\boldsymbol{l}$ の値を増加させ（たとえば $\boldsymbol{l}^{k+1}=10\boldsymbol{l}^k$）て(i)へもどる．

AJM は，Allran と Johnsen[20] によって紹介された変換法によるプログラムで，変換関数として

$$P_k(\boldsymbol{x}, \boldsymbol{t}^k)=F(\boldsymbol{x})+\sum_{i=1}^m\exp[t_i^k\cdot g_i(\boldsymbol{x})] \tag{4-8}$$

を考えている．但し t_i^k は

$$\left.\begin{array}{l} t_i^{k+1}<t_i^k<0 \\ \lim\limits_{k\to\infty} t_i^k=-\infty \end{array}\right\} \text{ for } i=1\cdots n$$

なる減少数列である．

(4-8)の変換関数を考える Allran 等の方法は，制約境界面 $g_i(\boldsymbol{x})=0$ 上でのペナルティ項

$$g(\boldsymbol{x})=\sum_{i=1}^m\exp[t_i^k\cdot g_i(\boldsymbol{x})]$$

の値が1にしかならず，したがって減少数列 t_i^k を定める初期の段階での目的関数 $F(\boldsymbol{x})$ の制約領域外へ向っての減少速度がペナルティ項の増加速度を上回る場合に問題がある．これをみるため，例えば次のような問題を考えてみる．

$$\left.\begin{array}{l} \text{Minimize } F(x)=-e^{x^2} \\ \text{subject to} \\ \quad -1\leq x\leq 1 \end{array}\right\} \tag{4-9}$$

この問題に対して作られる変換関数は

$$P_k(x, \boldsymbol{t}^k)=-e^{x^2}+e^{t_1^k(1+x)}+e^{t_2^k(1-x)}$$

となる．いま t_i^k として

$t_i^k=-k$ $(k=1, 2, \cdots)$ for $i=1, 2$

を定めたとする．するとペナルティ項は $k=1$ のとき

$$g(x)=e^{-(x+1)}+e^{(x-1)}$$

となり，Fig. 5 でみるように実行可能領域（**Fig. 5** の斜線領域）の外側での $F(x)$ の減少速度がペナテイ項 $g(x)$ の増加を上回っている．この結果 (4-9) の問題の解は，$k=1$ の段階で実行可能領域をはみだし，$\pm\infty$ に行ってしまうことになる．**TABLE 4** は実際に問題 (4-9) を (4-8) の変換関数を用いて解いた結果である．表に示すように $k=1$ で9回のイナーイテレーションで $-\infty$ に行ってい

TABLE 4 Computational result of the problem: Min $F(x)=-e^{x^2}$, subject to $-1\leq x\leq 1$, by the non-weighted Allran and Johnsen's technique

k	ITN	x	$P_k(x, t)$
1	1	0.0	−0.2642
	2	0.01	−0.2643
	3	0.04	−0.2652
	4	0.13	−0.2751
	5	0.40	−0.3781
	6	1.21	−2.9803
	7	3.64	−0.5678+6
	8	10.93	−0.7637+52
	9	32.80	−0.2894+77
	10	98.41	〃
	11	〃	〃
	12	〃	〃

TABLE 5 Computational result of the problem: Min $F(x)=-e^{x^2}$, subject to $-1\leq x\leq 1$, by the weighted Allran and Johnsen's technique

k	ITN	x	$P_k(x, t, W)$
1	4	0.0	35.7897
2	2	0.0	−0.9644
3	42	0.7711	−1.7088
4	18	0.8955	−2.1675
5	32	0.9422	−2.3932
6	11	0.9643	−2.5118
7	14	0.9762	−2.5791
8	12	0.9833	−2.6201
9	4	0.9833	−2.6297
10	4	0.9993	−2.7145

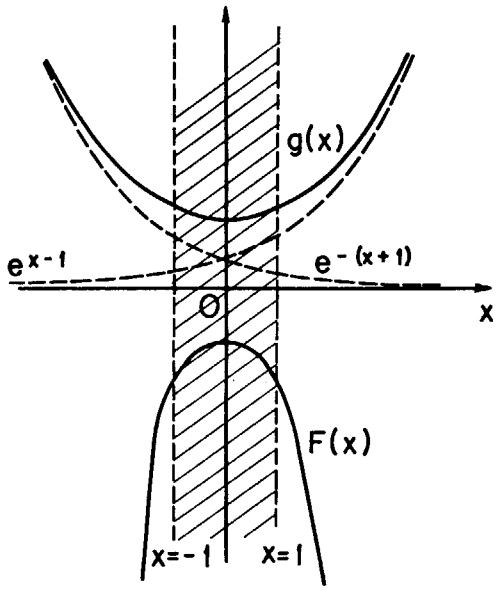


Fig. 5 One example of questions for Allran and Johnsen's technique by the problem to find "min $F(x)=-e^{x^2}$, subject to $-1\leq x\leq 1$"

る.

このような問題点を避けるために，我々は (4-8) 式の変換関数に重みを付けることを提案した[21]．すなわち，(4-8) の代わりに

$$P_k(\boldsymbol{x}, \boldsymbol{t}^k, \boldsymbol{w})=F(\boldsymbol{x})+\sum_{i=1}^{m} w_i \exp[t_i^k \cdot g_i(\boldsymbol{x})] \quad (4\text{-}10)$$

を考えるのである．そして重み $w_i>0$ を十分大きくとってやれば，初期の段階においても境界面の接近に対して大きなペナルティが加えられることになる．TABLE 5 は，問題 (4-9) を (4-10) の変換関数で重みを $w_1=50$, $w_2=50$ として計算した結果である．表に示すとおり，重み付けをした場合は探索が実行可能領域の内部で行なわれ，10 回のアウターイテレーションでこの問題の最適解 $x=1$, $F=-2.7183$ の良い近似解が得られている．

非線形最適化問題の中でも，制約条件式は線形で目的関数だけが非線形である場合には，問題はかなり扱い易くなり，特に目的関数が 2 次形，

$$F(\boldsymbol{x})=\boldsymbol{a}^t\boldsymbol{x}+\boldsymbol{x}^tA\boldsymbol{x}$$

をしている場合には 2 次計画法 (quadratic programming) の問題となる．このような問題に対して我々が整備したプログラムは，Beale の方法[22]による QBEAL と Wolfe の方法[23]による QWOLF である．また目的関数が一般の非線形の問題に対しては KEELE[24] が整備されている．

なお，TABLE 3 に示されているプログラムの中で CORASE と AJM は，それぞれ Price, Allran と Johnsen により提唱された方法に前述のような修正を施こして，著者により作成されたプログラムである．MAP, CUTPLN, PWM, QBEAL, QWOLF も著者が作成したもので，SIMPLEX は文献 15)，COMPLX は文献 25)，KEELE は文献 24) の中のプログラムを整備したものである．

4.2 制約条件を持つ問題に対する手法のベンチマーク・テスト

我々は TABLE 3 に示されたプログラムの中で SIMPLEX, COMPLX, CORASE, MAP, AJM に対するベンチマーク・テストを終了したので，その結果を紹介する．

TABLE 6 は今回ベンチマーク・テストを行なった 5 つのプログラムの特徴を示したものであり，constraints は扱うことができる制約条件式の形である．SIMPLEX と MAP では等号，不等号の両方を扱うことができる．他の 3 つのプログラムは不等号制約式だけを扱うものである．また S.P. は出発点が実行可能である必要があるか否かの情報で，SIMPLEX と MAP は出発点は実行不可能で良く，COMPLX と AJM は実行可能な出発点が必要で，CORASE では出発点が実行可能領域の中に乱数によって生成されるため入力する必要がない．Gradient は目的関数および制約条件式の傾斜ベクトルの計算ルーチンを必要とするか否かの情報で，MAP を除いたプログラムでは必要としない．但し，AJM では制約条件無しの手法に組み込まれるため，傾斜ベクトル計算ルーチンの必要性は組み込まれる手法に依存する．MAP では目的関数および制約条件式の傾斜ベクトル計算ルーチンを用意する必要がある．Peculiar Quantity は必要とする各手法に固有の量で，SIMPLEX では，次章で紹介するように単体を移動する際の折り返し (reflection) 係数 α，縮少 (contraction) 係数 β，拡大 (expansion) 係数 γ が必要である．( ) 内の数字は，標準値で，α, β, γ としてそれぞれ 1.0, 0.5, 2.0 が推薦されている．COMPLX では，複体の端点の数 $K(\geq n+1)$ と折り返し係数 α で，折り返し係数の標準値として 1.3 が妥当と思われる．CORASE では，stored point の数 N と単体の折り返し係数 α で N としては変数の数×20〜25 ぐらい，$\alpha=1.0$ が標準値である．MAP では，変数 x_i の変域を制限する刻み幅 δ と反復ごとに δ を縮少される係数 α が必要で，δ の定め方はある程度大きめに定める必要がある．

TABLE 6 Characteristics of the constrained nonlinear programming method

Program	Constraints	S. P.	Gradient	Peculiar Quantities	Convergence Criteria
SIMPLEX	$g_i(\boldsymbol{x})\geqq 0$ $h_j(\boldsymbol{x})=0$	infeasible	not necessary	reflection factor α (1.0) contraction factor β (0.5) expansion factor γ (2.0)	$\left\{\sum_j^{N+1}(F(\boldsymbol{x}^j)-F(\boldsymbol{x}^c))^2/N\right\}^{1/2}\leqq\varepsilon$
COMPLX	$g_j(\boldsymbol{x})\geqq 0$	feasible	not necessary	No. of extreme points. reflection factor α (1.3)	$F_{\min}-(F_{\max}+\varepsilon)>0$ for r-consecutive iterations
CORASE	$g_j(\boldsymbol{x})\geqq 0$	not necessary	not necessary	No. of stored point. reflection factor α (1.0)	same with SIMPLEX
MAP	$g_i(\boldsymbol{x})\geqq 0$ $h_j(\boldsymbol{x})=0$	infeasible	objective constraint	limitation of step size δ. contraction factor of δ.	$\lvert F(\boldsymbol{x}^k)-F(\boldsymbol{x}^{k+1})\rvert<\varepsilon_1$ and $\begin{cases}g_j(\boldsymbol{x}^{k+1})\geq-\varepsilon_2\\ \lvert h_j(\boldsymbol{x}^{k+1})\rvert\leq\varepsilon_2\end{cases}$
AJM	$g_j(\boldsymbol{x})\geqq 0$	feasible	—	monotonously decreasing sequence t_{jt}. weight of constraints.	—

AJM では $t_i^{k+1}<t_i^k<0$ なる単調減少列を与える必要がある.

これら5つのプログラムのテスト用に用意したテスト問題は以下の6題である.

Test Problems for Constrained Optimization

1. (Box)

Min. $b_0+a_{01}x_1+\left(\sum_{j=2}^{5}a_{0j}x_j\right)x_1$

subject to

$0\leq a_{i1}x_1+\left(\sum_{j=2}^{5}a_{ij}x_j\right)x_i\leq b_i,\quad i=1,2,3$

$\boldsymbol{x}_0=(2.52, 2.0, 37.5, 9.25, 6.8)$

$F^*=-5.280+6$ at $\boldsymbol{x}^*=(4.537, 2.4, 60.0, 9.3, 7.0)$

2. (Wilde)

Min. $-e^{(x_1-1)^2+(x_2-2)^2}$

subject to

$$\begin{cases} x_1-x_2^2\geq 0\\ -e^{-x_1}+x_2\geq 0,\quad x_1, x_2, x_3\geq 0\\ -2(x_1-1)^2+x_2\geq 0\end{cases}$$

$\boldsymbol{x}_0=(1.0, 1.0)$

$F^*=-23.8$ at $\boldsymbol{x}^*=(1.3586, 0.2571)$

3. (Wood)

Min. $100(x_2-x_1^2)^2+(1-x_1)^2+90(x_4-x_3^2)^2$
$+(1-x_3)^2+10.1[(x_2-1)^2+(x_4-1)^2]$
$+19.8(x_2-1)(x_4-1)$

subject to

$-10\leq x_i\leq 10,\quad i=1,2,3,4$

$\boldsymbol{x}_0=(-3.0, -1.0, -3.0, -1.0)$

$F^*=0.0$ at $\boldsymbol{x}^*=(1.0, 1.0, 1.0, 1.0)$

4.

Min. $\sum_j^5 e_jx_j+\sum_i\sum_j c_{ij}x_ix_j+\sum_j d_jx_j^3$

subject to

$\sum_j a_{ij}x_j-b_i\geqq 0,\quad i=1,\cdots,5$

$x_j\geqq 0,\quad j=1,\cdots,5$

$\boldsymbol{x}_0=(0, 0, 0, 0, 1)$

$F^*=-32.349$ at

$\boldsymbol{x}^*=(0.3, 0.3335, 0.4, 0.4285, 0.224)$

5.

Min. $\sum_i^{10}\{[\ln(x_i-2)]^2+[\ln(10-x_i)]^2\}$
$-\left(\prod_i^{10}x_i\right)^{0.2}$

subject to

$2.001<x_i<9.999,\quad i=1,\cdots,10$

$\boldsymbol{x}_0=(9.0,\cdots,9.0)$

$F^*=-45.778$ at $\boldsymbol{x}^*=(9.351,\cdots,9.351)$

6.

Min. $1000-x_1^2-2x_2^2-x_3^2-x_1x_2-x_1x_3$

subject to

$$\begin{cases}x_1^2+x_2^2+x_3^2-25=0\\ 8x_1+14x_2+7x_3-56=0\\ x_1, x_2, x_3\geq 0\end{cases}$$

$\boldsymbol{x}_0=(2.0, 2.0, 2.0)$

$F^*=961.715$ at $\boldsymbol{x}^*=(3.512, 0.217, 3.552)$

問題1は Box の問題と呼ばれるもので，目的関数と制約条件式が2次形式をしている問題である．但し係数 a_{ij} と右辺要素 b_i は与えられる量である．また，$\boldsymbol{x}_0$ は出発値で，$\boldsymbol{x}^*$ は既知な最適点，$F^*=F(\boldsymbol{x}^*)$ である．問題2は，Wilde の問題で，やはり最適解が既知な問題である．問題3は Wood の関数と呼ばれる目的関数の各変数の変域を制限した問題である．問題4は制約条件式が線形で目的関数が3次の関数である．問題5は，対数関数を目的関数とし，変数の変域制限がある問題である．最後の問題6は，等号制約条件式を持つ問題に対する MAP と SIMPLEX の実用性を調べるために設けたものである．TABLE 7 には上記の問題に対するベンチマーク・テストの結果が示されている．但し，この中で No.

TABLE 7 Results of benchmark tests for the Problem No. 1

Program	NFE	F-final	X-final	T (sec.)
SIMPLEX	2, 442	−5. 280+6	4. 537 2. 401 59. 975 9. 302 7. 01	96. 451
COMPLX	4, 026	−5. 280+6	4. 537 2. 401 59. 892 9. 302 7. 01	7. 403
CORASE	6, 001	−5. 280+6	4. 536 2. 40 59. 931 9. 33 7. 01	24. 951
MAP	4	−5. 280+6	4. 537 2. 40 60. 0 9. 30 7. 0	0. 600
AJM	4, 017	−5. 280+6	4. 537 2. 401 59. 90 9. 301 7. 0	11. 341

TABLE 7 (continued) Results of benchmark tests for the Problem No. 2

Program	NFE	F-final	X-final	T (sec.)
SIMPLEX	126	−23. 722	1. 3585 0. 2570	0. 789
COMPLX	37	−23. 722	1. 3585 0. 2570	0. 187
CORASE	3, 004	−23. 705	1. 3584 0. 2570	0. 801
MAP	8	−23. 722	1. 3585 0. 2570	0. 134
AJM	312	−23. 107	1. 3568 0. 2642	0. 270

はテスト問題の番号を表わし，NFE は収束するまでに要した目的関数計算ルーチンの呼びだし回数で，F-final, X-final はそれぞれ収束した点における目的関数値 F, 変数 $\boldsymbol{x}$ の値で，T は計算に要した時間（単位は秒）である．なお使用計算機は IBM 360/158 である．

表に示すとおり，問題 1 に対しては各プログラムとも最適点の近傍に収束しているが，要した計算時間にはかなりの差がみられる．すなわち，SIMPEX では 96. 451, CORASE では 24. 951 秒もの計算時間を要したのに対し，MAP ではわずか 0. 6 秒で計算が終了している．また，他の 2 つのプログラムと比較しても MAP の効率が良く，2 次形式をした問題に対しては MAP が適していることが推定される．問題 2 に対しても，全てのプログラムが最適点の近傍に収束している．計算時間では，問題 1 ほどの差はみられなかったが，MAP, COMPLX が

TABLE 7 (continued) Results of benchmark tests for the Problem No. 3

Program	NFE	F-final	X-final	T (sec.)
SIMPLEX	665	0. 321−15	1. 0 ⋮ 1. 0	2. 507
COMPLX	754	0. 730−9	1. 0 ⋮ 1. 0	2. 356
CORASE	4, 001	0. 374−8	1. 0 0. 997 1. 0 0. 996	3. 188
MAP	32	0. 609−11	1. 0 ⋮ 1. 0	1. 273
AJM	1, 812	0. 449−9	1. 0 ⋮ 1. 0	2. 175

TABLE 7 (continued) Results of benchmark tests for the Problem No. 4

Program	NFE	F-final	X-final	T (sec.)
SIMPLEX	1, 174	−32. 349	0. 3 0. 3335 0. 4 0. 4283 0. 2240	27. 363
COMPLX	1, 739	−32. 332	0. 2998 0. 3316 0. 3997 0. 4338 0. 2320	11. 223
CORASE	6, 001	−32. 337	0. 3 0. 3329 0. 3999 0. 4279 0. 2301	18. 147
MAP	64	−32. 349	0. 3 0. 3335 0. 4 0. 4283 0. 2240	8. 473
AJM	3, 803	−32. 344	0. 3 0. 3331 0. 4 0. 4285 0. 2240	13, 335

TABLE 7 (continued) Results of benchmark tests for the Problem No. 5

Program	NFE	F-final	X-final	T (sec.)
SIMPLEX	1, 154	−45. 778	9. 3502 ⋮ 9. 3502	13. 843
COMPLX	1, 927	−45. 778	9. 3503 ⋮ 9. 3503	20. 886
CORASE	8, 001	−42. 329	9. 01 8. 86 8. 99 ≀ 9. 30	42. 331
MAP	61	−45. 778	9. 3503 ⋮ 9. 3503	6. 128
AJM	3, 411	−45. 778	9. 3502 ⋮ 9. 3502	14. 854

いた4つのルーチンとも実行可能領域の内部で探索を行なっており，とくに COMPLX, CORASE と SIMPLEX の探索の過程に類似性が見られる．これは，前に説明した手法の類似性からもうなずけることである．一方，MAP では一たん実行可能領域の外に出ているが，これは，Fig. 4 の MAP の計算フローからわかるように，LP 計算の際の制約条件式の中には，もとの制約式を接平面で近似した式の他に，有界な凸多面体を作るために各変数に対して

$$|x_i - x_i^k| \leq \delta_i^k$$

なる制限を設け，計算の初期の段階では制限値 δ_i^k はもとの領域（Fig. 6 の斜線部）と共通領域を持つ程度に大きい値が設定され，したがって生成された凸多面体の端点の中に実行可能領域の外にはみ出るものがあって，LP 計算の最適解としてその点が求められることがあるからである．さらに，反復計算と共に δ_i^k は縮少されていくため，最後の段階では凸多面体は実行可能領域の内

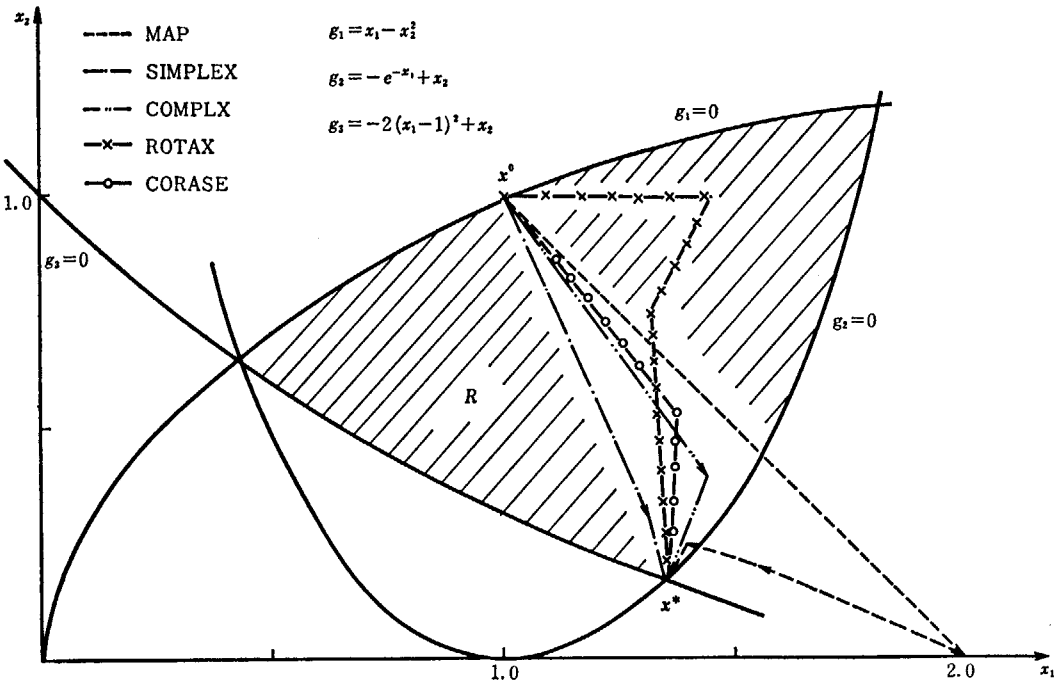


Fig. 6 Comparison of searching trajectories of MAP, CORASE, COMPLX, SIMPLEX and AJM for Wilde's problem.

速かった．

Fig. 6 にはこの問題に対する各ルーチンの探索の過程が示されている．図の中で制約式 g_1, g_2, g_3 に囲まれた斜線の領域が実行可能領域である．図のように MAP を除部に生成されることになる．

問題3に対しては，精度の点では SIMPLEX と MAP が優れ，計算速度では MAP が速かったほかはほぼ同程度といえる．問題4に対しては，COMPLX と CORASE

TABLE 8 Result of benchmark test for the Problem No. 6

Program	NFE	F-final	X-final	$h_i(x)$	T (sec)
SIMPLEX	222	961. 717	3. 547 0. 2143 3. 517	h_1=3. 57−3 h_2=4. 8−3	1. 169
MAP	5	961. 632	3. 444 0. 2176 3. 628	h_1=7. 1−2 h_2=7. 6−3	0. 537

の精度が多少悪かったが，他はほぼ同程度であり，計算時間では MAP が速く逆に SIMPLEX, CORASE が遅く，問題1やこの問題のような2次，3次の多項式を目的関数とする問題に対する単体上での探索の効率の悪さが判明した．問題5に対しては，精度，計算時間ともに CORASE が劣り，変数の数が多くなると CORASE の効率の悪さが目につく．これは，手法の説明にもあったとおり，最初に生成すべき stored point の数が変数の数に比列して多くなり，しかもその各点での目的関数値が評価されるためと思われる．他方 MAP では，多変数の問題に対しても効率が良く，次いで SIMPLEX, AJM の順であった．

なお4.1節で触れたとおり，CORASE は，Price によって紹介された制御ランダム法の収束を加速するように改良したものであるが，加速を行なわずに計算した結果は，要した計算では問題1から問題5に対してそれぞれ，31.122, 0.921, 4.643, 26.472, 47.120 秒であった．これは加速を加えた結果のほぼ1.1倍から1.5倍の計算時間になっており，前述の改良の効果があらわれている．なお目的関数値は各問題ともほぼ同程度であった．

TABLE 8 は，等号制約条件を持つ問題6を SIMPLEX と MAP で解いた結果であるが，表にみられるとおりきわだった差は見られず，計算時間では多少 MAP が速く，解の実行可能性という意味では SIMPLEX が多少優れている程度であった．なお MAP では，LP ルーチンとして第3章で紹介した LPM が用いられた．また AJM では，制約条件無しの手法として次節で紹介する座標回転法によるプログラム ROTAX が用いられた．

4.3 制約条件を持たない問題に対する手法

制約条件を持たない問題に対する最適化の手法は，制約条件を持つ問題に対するものと比べて広く研究され，したがって多くの方法が開発されている．それらの手法は，大別して直接探索法 (direct search method) と傾斜法あるいは降下法 (descent search method) に分類される．また，それらの間にはおおよそ次のような相異がある．

直接探索法	降 下 法
(i)過去の反復で得られた情報を使わないことが多い．	(i)ほとんどの場合，過去の反復で得られた情報を使う．
(ii)前もって決められた1組の方向に沿って系統的に目的関数を調べていく．	(ii)降下方向は新しい反復毎に決められる．
(iii)各反復では目的関数の値を計算するだけである．	(iii)各反復毎に目的関数の値，1階および高階の導関数まで計算する．

4.3.1 直接探索法

TABLE 9 には，我々が整備開発した直接探索法のプログラムの名前と手法が示されている．

ALPS は Hooke と Jeeves により紹介されたパターン探索法[25]に基づくプログラムで，この手法は，与えられた基点の回りでの目的関数の局所的 (local) な動きを探索するパターン決定 (exploratory) と，その局所的な探索によって到達した点から目的関数の大域的 (global) な動きを探索するために次の基点へと移動するパターン移動という二つの操作から成っている．Fig. 7 には二変数の場合のパターン決定とパターン移動の図が載せられている．

図の説明をすると，まず1の点 $\boldsymbol{x}^1$ を出発基点として各座標軸方向に沿ってパターン決定が行なわれた．このとき点 $\boldsymbol{x}^3$ は失敗，すなわち $F(\boldsymbol{x}^3)>F(\boldsymbol{x}^2)$ だった．2, 4 が成功で新しい基点 $\boldsymbol{x}^4$ が求まった．次いでパターン移動により $\boldsymbol{x}^5$ を求めたが，このとき $F(\boldsymbol{x}^4)$ と $F(\boldsymbol{x}^5)$ の比較は行なわず，すぐにパターン決定にはいる．この結果到達した点 $\boldsymbol{x}^8$ と現基点 $\boldsymbol{x}^4$ における関数値が比較され，$F(\boldsymbol{x}^8)<F(\boldsymbol{x}^4)$ だったので，$\boldsymbol{x}^8$ を新しい基点として次のパターン移動が行なわれ $\boldsymbol{x}^9$ が得られた．10, 11, 12 と失敗し 13 で成功したが，$F(\boldsymbol{x}^{13})>F(\boldsymbol{x}^8)$ だったので，$\boldsymbol{x}^{13}$ は基点として採用されず，$\boldsymbol{x}^8$ を再び出発基点として，ステップ幅を縮めてパターン決定を行なった．以下同様の過程で探索が続けられる．

このようなパターン決定とパターン移動の計算は次の

TABLE 9 Programs based on the direct search method

Program	Method
ALPS	pattern search method by Hooke and Jeeves
ROTAX	axes rotation method by Rosenbrock
ALSIM	simplex method by Nelder and Mead
SPASE	spherical search method by Suzuki
CORASE	controlled random search method by Price

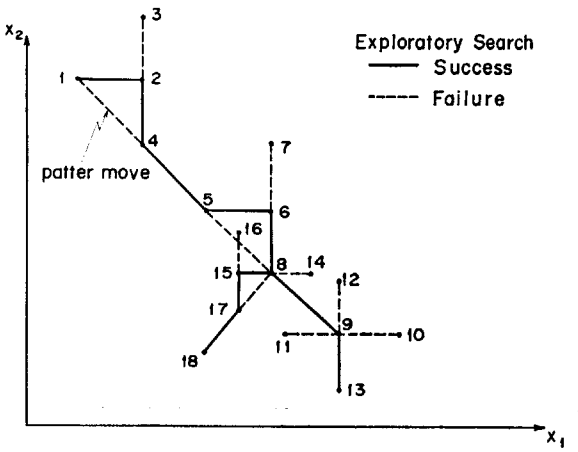


Fig. 7 Trajectories of exploratory and pattern move in the case of two variables.

ように行なわれる.

パターン決定

(i) $i=1$ として目的関数 $F=F(\boldsymbol{x}^1)$ の計算

(ii) 新しい試行点を求める

$\boldsymbol{x}=(x_1, x_2, \cdots, x_i+\Delta x_i, x_{i+1}, \cdots, x_n)$

但し Δx_i は x_i 軸方向のステップ幅である.

(iii) $F(\boldsymbol{x})<F$ のときは, $F=F(\boldsymbol{x})$, $i=i+1$ として(ii)へ戻る.

(iv) $F(\boldsymbol{x})\geq F$ のときは, 仮りに $\boldsymbol{x}=(x_1, \cdots, x_i-\Delta x_i, \cdots, x_n)$ とおき, $F(\boldsymbol{x})$ の評価をし, このとき $F(\boldsymbol{x})<F$ なら $F=F(\boldsymbol{x})$ とおき, 仮りの $\boldsymbol{x}$ を確定し, $i=i+1$ として(ii)へ戻る. このときもまた $F(\boldsymbol{x})\geq F$ なら, x_i はそのままにして, $i=i+1$ として(ii)へ戻る.

パターン移動

ある段階における現基点 $\boldsymbol{x}^B$ と前の基点 $\tilde{\boldsymbol{x}}^B$ から

$\boldsymbol{x}=\boldsymbol{x}^B+(\boldsymbol{x}^B-\tilde{\boldsymbol{x}}^B)$

へと飛ぶ.

ROTAX[21] は, Rosenbrock により紹介された座標回転法[26]を用いたプログラムで, 変数次元空間での直交座標系の1つの座標軸が目的関数の最急降下方向に向くように座標系を回転させながら探索を行なう方法である. 座標回転法のアルゴリズムは次のようになる.

(i) 出発点 $\boldsymbol{x}^0$ と直交する n 個の方向:

$S=(\boldsymbol{s}^0, \boldsymbol{s}^1, \cdots, \boldsymbol{s}^{n-1})$

を与える. 最初は各座標軸方向を与える.

(ii) S の各方向に対し順次, 以下の探索を行なう.

$$\begin{cases} \min\{F(\boldsymbol{x}^0), F(\boldsymbol{x}^0+\lambda_0\boldsymbol{s}^0)\} \text{ なる点 } \boldsymbol{x}^1 \text{ を求める} \\ \quad\vdots \\ \min\{r \text{ 番目の試行の最小値 } F(\boldsymbol{x}^r+\lambda_r\boldsymbol{s}^r)\} \text{ なる点} \\ \quad \boldsymbol{x}^{r+1} \text{ を求める} \\ \quad\vdots \\ \min\{(n-1) \text{ 番目の試行の最小値}, \\ \quad F(\boldsymbol{x}^{n-1}+\lambda_{n-1}\boldsymbol{s}^{n-1})\} \text{ なる点 } \boldsymbol{x}^n \text{ を求める.} \end{cases}$$

各探索毎に λ_j は

探索が成功なら, $\lambda_j=\alpha\lambda_j$

失敗なら, $\lambda_j=-\beta\lambda_j$

とする. 但し $\lambda_0, \cdots, \lambda_{n-1}$ は各方向の刻み幅, $\alpha>1$, $0<\beta<1$ は刻み幅調節係数である.

探索(ii)は各方向で1つでも成功する限りくり返す.

(iii) Λ_j を $\boldsymbol{s}_j$ 方向について成功したステップ幅の合計とするとき, 次の一次独立な方向ベクトルを定義する.

$$Q=[\boldsymbol{q}^0, \boldsymbol{q}^1, \cdots, \boldsymbol{q}^{n-1}]=S\begin{bmatrix} \Lambda_0 & & & 0 \\ \Lambda_1 & \Lambda_1 & & \\ \Lambda_2 & \Lambda_2 & \Lambda_2 & \\ \vdots & \vdots & & \vdots \\ \Lambda_{n-1} & \Lambda_{n-1} & \cdots & \Lambda_{n-1} \end{bmatrix}$$

上式により定義されたベクトルの中で, $\boldsymbol{q}^0$ は出発点 $\boldsymbol{x}^0$ と探索(ii)により到達した試行点を結ぶベクトルで, $\boldsymbol{q}^1$ は $\boldsymbol{s}_0$ 以外の方向への成功したステップ幅の合計, $\boldsymbol{q}^2$ は $\boldsymbol{s}_0, \boldsymbol{s}_1$ 以外の方向への成功したステップ幅の合計, …, である.

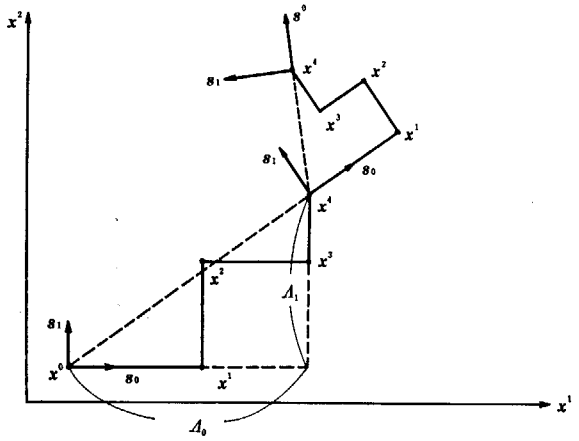


Fig. 8 Trajectories of axes rotation method in the case of two variables.

(iv) (iii)で定義された方向ベクトルから Gram-Schmidt の直交化法により1組の新しい直交ベクトルを生成する.

$\boldsymbol{v}^0=\boldsymbol{q}^0$

$\boldsymbol{v}^i=\boldsymbol{q}^i-\sum_{j=0}^{i-1}[(\boldsymbol{q}^i)^t\hat{\boldsymbol{s}}_j]\hat{\boldsymbol{s}}_i$, $i=1, 2, \cdots, n-1$

但し,

$\hat{\boldsymbol{s}}_i=\boldsymbol{v}^i/\|\boldsymbol{v}^i\|$, $i=0, 1, \cdots, n-1$

(v) (iv)により求まった新しい探索方向

$S=(\hat{\boldsymbol{s}}_0, \hat{\boldsymbol{s}}_1, \cdots, \hat{\boldsymbol{s}}_{n-1})$

と $\boldsymbol{x}^0=\boldsymbol{x}^n$ を用いて(i)からくり返す.

Fig. 8 に二変数の場合の座標回転法の探索の過程が示されている.

ALSIM は, Nelder と Mead により開発されたシンプレックス法[27]を採用したプログラムで, n (変数の数)次元空間に張られる正則単体の $n+1$ 個の端点上での目的関数値を比較しながらこの単体を求める最適点へと動かして行く操作であり, この単体の移動は折り返し (reflection), 縮小 (contraction), 拡大 (expansion) の3つの基本操作によって行なわれる. シンプレックス法のアルゴリズムの説明のため, 次のような量を定義する.

(イ) $\boldsymbol{x}^h$ は単体の端点の中で目的関数が最大値をとる点, すなわち

$F(\boldsymbol{x}^h)=\max_i F(\boldsymbol{x}^i)$, $i=1, \cdots, n+1$

(ロ) $\boldsymbol{x}^s$ は目的関数が二番目に大きい値をとる点.

(ハ) $\boldsymbol{x}^l$ は目的関数が最小値をとる点, すなわち

$F(\boldsymbol{x}^l)=\min_i F(\boldsymbol{x}^i)$, $i=1, \cdots, n+1$

(ニ) $\boldsymbol{x}^0$ は $i=h$ を除く全ての $\boldsymbol{x}^i$ の重心, すなわち

$\boldsymbol{x}^0=\frac{1}{n}\sum_{\substack{i=1 \\ i\neq h}}^{n+1}\boldsymbol{x}^i$

次にシンプレックス法の3つの基本操作の定義をする. (Fig. 9 参照)

(イ) reflection :

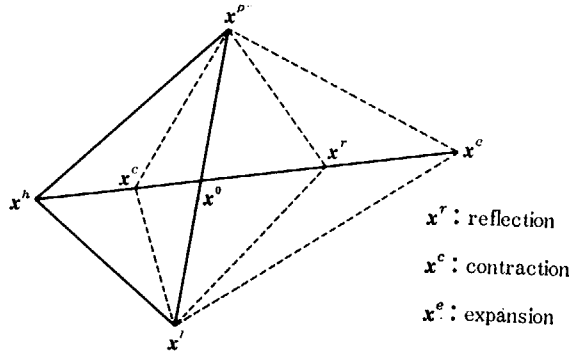


Fig. 9 Three fundamental operations of simplex search

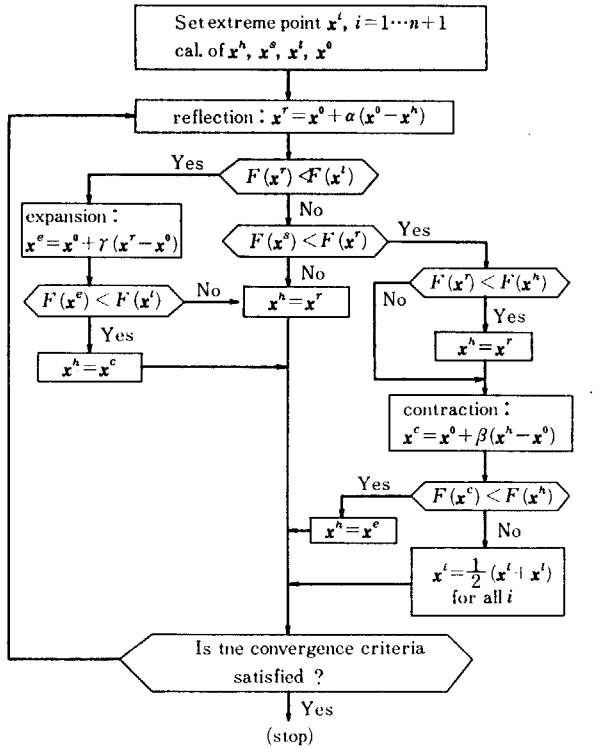


Fig. 10 Flow diagram of simplex search.

端点 x^h を

$$x^r=x^0+\alpha(x^0-x^h),\quad \alpha>0$$

に置き換える操作.

(ロ) expansion :

目的関数をさらに改善すると考えられる x^0-x^r 方向に沿って x^r を

$$x^e=x^0+\gamma(x^\gamma-x^0),\quad \gamma>1$$

に置き換える操作.

(ハ) contraction :

単体を縮めて x^h を

$$x^c=x^0+\beta(x^h-x^0),\quad 0<\beta<1$$

に置き換える操作.

なお，これらの操作に使われる折り返し係数 α，縮少係数 β，拡大係数 γ などの定め方には今のところ基準が与えられていないが，Nelder 等は $\alpha=1, \beta=0.5, \gamma=2$ を提案している．以上のような定義の下でシンプレックス法の計算手順の流れ図は Fig. 10 に示されている．図の中での収束判定法は，新しい単体の $n+1$ 個の端点における目的関数値の標準偏差をあらかじめ設定した許容誤差と比較し，

$$\left\{\frac{1}{n}\sum_{i=1}^{n+1}[F(x^i)-F(x^0)]^2\right\}^{1/2}\leqq\varepsilon$$

により行なわれる．

SPASE は，著者により提案された球面探索法 (Spherical Search Method) によるプログラム[8]で，変数次元空間内に生成された超球面上に一様に分布する探索点を発生させ，最小値を与える点と球の中心を結ぶ方向に球を移動させながら最適点を探索していこうというもので，以下にアルゴリズムを概説する．

(i) 探索の初めに，原点を中心とする単位球面上に N 個の点 x_j を一様に発生させる．但し N は入力である．

(ii) 出発点 x^0，球の半径 r_0 を与える．

(iii) 球の中心を $x^c=x^0$，半径 $r=r_0$ とし，$F_0=F(x^0)$ を計算する．

(iv) (i)により求めた N 個の点を，半径 r，中心 x^c の球面上に次のように対応させる．

$$x^i=r\cdot x_i+x^c,\quad i=1,\cdots,N$$

また $F_m=\min_i F(x^i)$ を求め，そのときの点を x^m とする．このとき

$$\|x^m-x^0\|\leqq\varepsilon$$

なら(vi)へいく．但し，ε はあらかじめ定めた許容量で，$\|\cdot\|$ はユークリッドの距離である．

(v) $F_m<F_0$ なら $F_0:=F_m$, $x^0:=x^m$ とし，x^m-x^c 方向に球を移動させ，新しい球の中心を

$$x^c:=x^c+\beta(x^m-x^c)$$

により求める．但し，β は球の移動幅で，$A:=B$ は A を B で置き換えることを意味している．

(vi) $F_m\geqq F_0$ なら $x^c:=x^0$ とし球の半径を

$$r=\alpha\cdot r$$

とする．但し，$0<\alpha<1$ は球の半径縮少係数である．

(v), (vi)いずれの場合も(iv)に戻る．また(iv), (v), (vi)の反復の過程で球の半径があらかじめ定めた基準以下になったら計算を終了する．

なお，Table 9 のプログラムの中で SPASE, ROTAX, CORASE は筆者が作成したものであり，ALPS と ALSIM は ALPAC[35] ルーチンを整備したものである．

4.3.2 降 下 法

前に述べたとおり，降下法は直接探索法と比べて各反復の過程で使う情報量，例えば，傾斜ベクトル，ヘッセ行列が多くなり，したがって多くの記憶容量を必要とするが，反面情報量が多いということは1回の反復で目的関数をより良く改善できるという長所になっている．

一般に，降下法の反復計算は3つの部分から構成される．すなわち，

(i) 降下方向 s^k の計算．(肩の k は反復回数を示す)

ここで降下方向というのは，$\lambda^*\geqq\lambda>0$ なるすべての

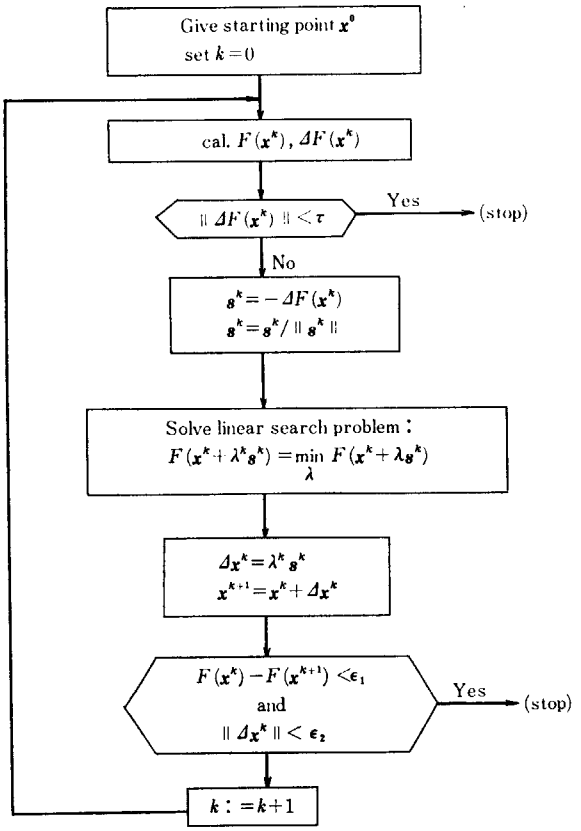


Fig. 11　Flow diagram of the steepest descent method.

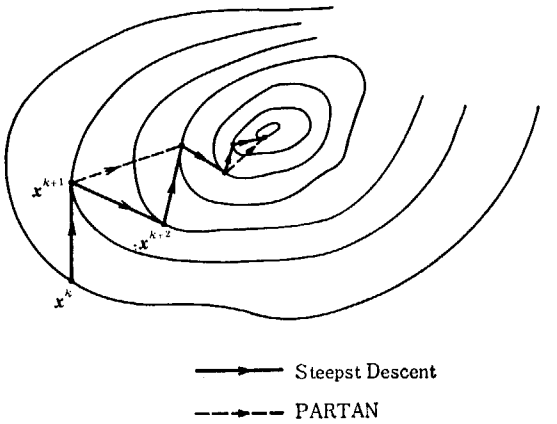


Fig. 12　Comparison of the steepest descent and PARTAN trajectories.

λ 対し

$$F(\boldsymbol{x}^{k+1}) = F(\boldsymbol{x}^k + \lambda \boldsymbol{s}^k) < F(\boldsymbol{x}^k)$$

なる λ^* が存在するとき，方向 $\boldsymbol{s}^k$ を $\boldsymbol{x}^k$ における目的関数 $F(\boldsymbol{x})$ の降下方向ということである．例えば $F(\boldsymbol{x})$ が微分可能なら，

$$(\boldsymbol{s}^k)^t \cdot \nabla F(\boldsymbol{x}^k) < 0$$

のとき，$\boldsymbol{s}^k$ は降下方向になる．

(ii)　降下方向 $\boldsymbol{s}^k$ に沿った降下ステップの刻み幅 λ^k の計算．

(iii)　(i)，(ii) により求められた降下方向と刻み幅により新しい点

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \lambda^k \boldsymbol{s}^k$$

を求める．このような3つの部分から成る降下法の手法は，方向を $\boldsymbol{s}^k$ をどのように定めるか，刻み幅 λ^k をどのように計算するかにより種々の方法が提案されている．以下にそれらの方法をみてみる．

最急降下法

計量行列を単位行列にとり，方向として目的関数が局所的に最も減少する方向 $-\nabla F(\boldsymbol{x}^k)$ と定めるのが最急降下法である．すなわち $\boldsymbol{s}^k$ は

$$\boldsymbol{s}^k = -\nabla F(\boldsymbol{x}^k)$$

とされる．Fig. 11 に最急降下法のアルゴリズムを示す．Forsythe と Motzkin は最急降下法の収束を加速した新しい方法を提し，さらに Shah らはそれを一般化した傾斜 PARTAN 法 (parallel tangents 法) を紹介している．この最急降下法の過程は，Fig. 12 の破線で示されている直線

$$\boldsymbol{s}^k = \boldsymbol{x}^k - \boldsymbol{x}^{k-2} \qquad (4\text{-}7)$$

の間を行き来しながら最小点に収束していく．そこで (4-7) の方向と最急降下方向を交互に用いる，すなわち

$$\boldsymbol{s}^0 = -\nabla F(\boldsymbol{x}^0)$$
$$\boldsymbol{s}^1 = -\nabla F(\boldsymbol{x}^1)$$
$$\boldsymbol{s}^2 = \boldsymbol{x}^2 - \boldsymbol{x}^0$$
$$\boldsymbol{s}^k = \begin{cases} -\nabla F(\boldsymbol{x}^k), & k=3,5,7,\cdots \\ \boldsymbol{x}^k - \boldsymbol{x}^{k-2}, & k=4,6,8,\cdots \end{cases}$$

によって収束を加速しようというものである．

Newton 法

Newton 法では計量行列としてヘッセ行列 $H(\boldsymbol{x}^k)$ を用いる．ここでヘッセ行列とは

$$H(\boldsymbol{x}_k) = \begin{bmatrix} \partial g_1/\partial x_1, \partial g_1/\partial x_2, \cdots, \partial g_1/\partial x_n \\ \partial g_2/\partial x_1 \cdots\cdots\cdots \partial g_2/\partial x_n \\ \vdots \\ \partial g_m/\partial x_1 \cdots\cdots\cdots \partial g_m/\partial x_m \end{bmatrix}_{x=x^k}$$

で定義されるものである．また探索方向を

$$\boldsymbol{s}^k = -H(\boldsymbol{x}^k)^{-1} \cdot \nabla F(\boldsymbol{x}^k)$$

とする．このとき

$$(\boldsymbol{s}^k)^t \cdot \nabla F(\boldsymbol{x}^k) = -(\nabla F(\boldsymbol{x}^k))^t \cdot H(\boldsymbol{x}^k)^{-1} \cdot \nabla F(\boldsymbol{x}^k)$$

であるから，もし $H(\boldsymbol{x}^k)$ が正値対称行列なら

$$(\boldsymbol{s}^k)^t \cdot \nabla F(\boldsymbol{x}^k) < 0$$

となるから，$\boldsymbol{s}^k$ は降下方向となる．ヘッセ行列 $H(\boldsymbol{x}^k)$ が正値でない場合には Newton 法は使えないが，これを補うため，例えば，Fiacco と McCormick 等は，$H(\boldsymbol{x}^k)$ の LU 分解，つまり H の下3角行列と上3角行列への分解を修正することにより $\boldsymbol{s}^k$ を降下方向にする方法を提案している[28]．

可変計量法：

各反復ごとに計量行列を更新していくこの方法は，Davidon-Fletcher, Powell[29),30),31)] により開発されたもので，計量行列の更新の方法によって異なった方法がある．一般には，更新される計量行列 A^{k+1} は，前の計量行列 A^k と $\Delta \boldsymbol{x}^k = \boldsymbol{x}^k - \boldsymbol{x}^{k-1}$ と $\boldsymbol{y}^k = \nabla F(\boldsymbol{x}^k) - \nabla F(\boldsymbol{x}^{k-1})$ の関数として計算される．Fig. 13 に Davidon-Fletcher, Powell による可変計量法の流れ図を載せる．

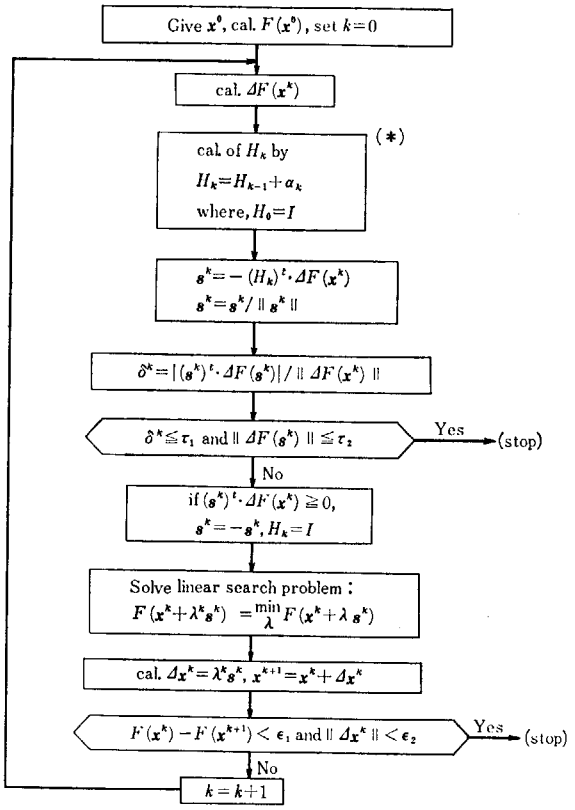


Fig. 13 Flow diagram of Variable Metric by Davidon-Fletcher, Powell

この流れ図の中で，(*)印の付いたところで使われる α_k は次式で計算される．

$$\alpha_k=\frac{\Delta x^{k-1}\cdot(\Delta x^{k-1})^t}{(\Delta x^{k-1})^t\cdot y^{k-1}}-\frac{(H_{k-1}\cdot y^{k-1})\cdot(H_{k-1}\cdot y^{k-1})^t}{(y^{k-1})^t\cdot H_{k-1}\cdot y^{k-1}}$$

共役方向法，共役傾斜法

Powell は，与えられた n 個の共役方向 $s_0,\cdots,s^{n-1}$ に沿った n 回の探索で点 $x^0,\cdots,x^{n-1}$ を生成するサイクルをくり返すことにより，最小点を探索する共役方向法を提唱した[32]．最初のサイクルでは $s^0,\cdots,s^{n-1}$ として各座標軸方向を定め，次のサイクルからは新しい方向として

$$\left.\begin{aligned}&s^k=s^{k+1},\quad k=0,\cdots,n-2\\&s^{n-1}=x^n-x^0\end{aligned}\right\}\qquad(4\text{-}8)$$

を採用しようというものである．一方，共役傾斜法[30]は共役方向の生成に目的関数の傾斜ベクトルを用いるもので，すなわち，(4-8)の代わりに

$$s^k=-\nabla F(x^k)+\alpha^k s^{k-1}$$

但し，$\alpha^k=\dfrac{(\nabla F(x^k))^t\cdot\nabla F(x^k)}{(\nabla F(x^{k-1}))^t\cdot\nabla F(x^{k-1})}$

により降下方向 s^k を求めるのである．

以上で降下方向の定め方によるいろいろな降下法の手法を見てきたが，次に降下法において重要な降下幅の計算法についてふれる．これは，ある反復の過程において点 x^k と方向 s^k が与えられ，次の点を求めるために s^k に沿ってどのくらい降下したら良いかという問題，すなわち

TABLE 10 Programs based on the descent search method

Method	Program
Conjugate Direction by Powell	ALCODR
Conjugate Gradient by Fletcher and Reeves	VA 08A, CGD
Parallel Tangent by Shah	ALPART
Modified Newton	MINIM
Variable Metric by Davidon Fletcher, Powell	VA 01 A, VA 06 A, VA 09 A, VA 10 A, FPD, ALVAM

$$\underset{\lambda}{\text{Minimize}}\ F(x^k+\lambda s^k)\qquad(4\text{-}10)$$

により定式化される．x^k, s^k は既知量であるから，(4-10)はスカラー変数 λ についての最小化問題である．この意味で (4-10) は，一次元探索あるいは直線探索 (linear search) の問題と呼ばれている．さらに次節でもふれるように，降下法の計算の中での目的関数の計算回数はそのほとんどがこの直線探索によるものであり，したがって効率の良い手法を選ぶことが大切である．手法としては，おもに Fibonacci 法，黄金分割法および低次多項式補間法などがあるが，(4-10)が λ 関して単峰性の場合には Fibonacci 法が良い．また一次元探索による関数評価回数を押さえようとする場合には黄金分割法が良い．多項式補間によるものの中では，Powell による二次多項式補間，Davidon 等による三次多項式補間が良く用いられている．これは，「任意の連続関数は多項式によって十分精密に近似できる」という有名な Weierstrass の多項式近似定理により一つの保証が与えられているからであり，また低次多項式で近似することによりその解を根と係数の関係から解析的に求めることができることにもよる．

TABLE 10 に，降下法によるプログラムのうち我々が整備したものの名前を載せる．

表中で，VA シリーズは英国 Harwell 原子力研究所理論物理部において開発されたもの，FPD と CGD は IBM system/360 の科学用サブルーチンパッケージ，頭2文字が AL で始まるプログラムは米国の Princeton 計算センターのソフトウエア，さらに MINIM は Joint Nuclear Research Center のイスプラ研究所（イタリア）で開発されたプログラムを整備したものである．

4.4 制約条件無しの問題に対する手法のベンチマーク・テスト

4.2, 4.3 節において，制約条件無しの最適化問題に対する各種の手法とプログラムをみてきたが，次に，それらのプログラムに対して我々が実施したベンチマーク・テストの結果を紹介する．テストのために採用した関数は以下の9つである．

TEST FUNCTIONS

1. Rosenbrock

$F(\boldsymbol{x})=100(x_2-x_1^2)^2+(1-x_1)^2$

$\min F=0$ at $\boldsymbol{x}=(1, 1)$

2. Beale

$F(\boldsymbol{x})=\sum_{i=1}^{3}\{c_i-x_1(1-x_2^i)\}^2$, c_i : given

$\min F=0$ at $\boldsymbol{x}=(3, 0.5)$

3. Box

$F(\boldsymbol{x})=\sum_{i=1}^{10}\{(e^{-x_1y_i}-e^{-x_2y_i})-x_3(e^{-y_i}-e^{-10y_i})\}^2$,

y_i : given

$$\min F=0 \text{ at } \boldsymbol{x}=\begin{cases}(1, 10, 1)\\(10, 1, -1)\\(x_1=x_2,\ x_3=0)\\(x_1, x_2\to+\infty,\ x_3=0)\end{cases}$$

4. Enzyme

$F(\boldsymbol{x})=\sum_{i=1}^{11}\{v_i-x_1(y_i^2+x_2y_i)/(y_i^2+x_3y_i+x_4)\}^2$,

v_i, y_i : given

$\min F=3.075\times10^{-4}$

at $\boldsymbol{x}=(0.1928, 0.1916, 0.1234, 0.1362)$

5. Watson

$$F(\boldsymbol{x})=\sum_{i=1}^{30}\left\{\sum_{j=1}^{m}(j-1)x_jy_i^{j-2}-\left(\sum_{j=1}^{m}x_jy_i^{j-1}\right)^2-1\right\}^2+x_1^2,$$

y_i : given $(=(i-1)/29)$

$\min F=2.288\times10^{-3}$ at

$\boldsymbol{x}=(-0.016, 1.012, -0.233, 1.260, -1.513, 0.993)$

6. Powell

$F(\boldsymbol{x})=(x_1+10x_2)^2+5(x_3-x_4)^2$
$+(x_2-2x_3)^4+10(x_1-x_4)^4$

$\min F=0$ at $\boldsymbol{x}=(0, 0, 0, 0)$

7. Wood

$F(\boldsymbol{x})=100(x_2-x_1^2)^2+(1-x_1)^2$
$+90(x_4-x_3^2)^2+(1-x_3)^2$
$+10.1[(x_2-1)^2+(x_4-1)^2]$
$+19.8(x_2-1)(x_4-1)$

$\min F=0$ at $\boldsymbol{x}=(1, 1, 1, 1)$

8. Gauss

$F(\boldsymbol{x})=\sum_{i=1}^{15}\{x_1\cdot e^{-(z_i-x_3)^2\cdot x_2/2}-y_i\}^2$

z_i, y_i : given

$\min F=1.128\times10^{-8}$ at

$\boldsymbol{x}=(0.39896, 1.0, 0.0)$

9. Extended Rosenbrock

$F(\boldsymbol{x})=\sum_{i=1}^{m/2}\{100(x_{2i}-x_{2i-1}^2)^2+(1-x_{2i-1})^2\}$

上記のように，2変数の関数として Rosenbrock および Beale，3変数の関数として Box および Gauss，4変数として Enzyme, Powell および Wood，6変数として

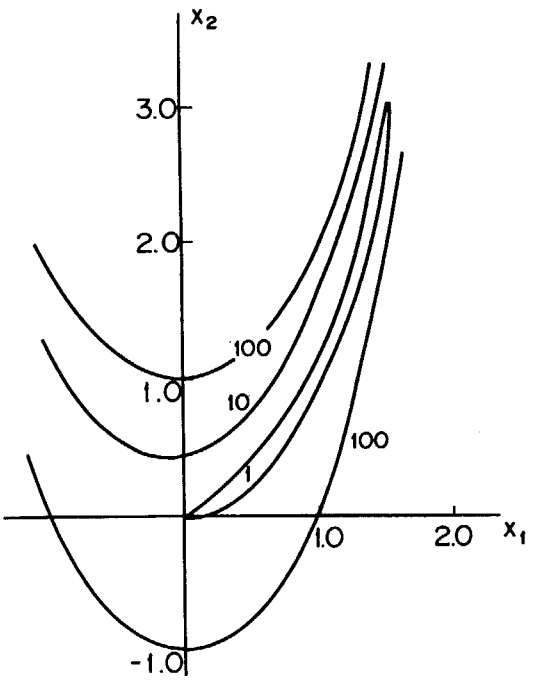


Fig. 14 Contours of Rosenbrock's function

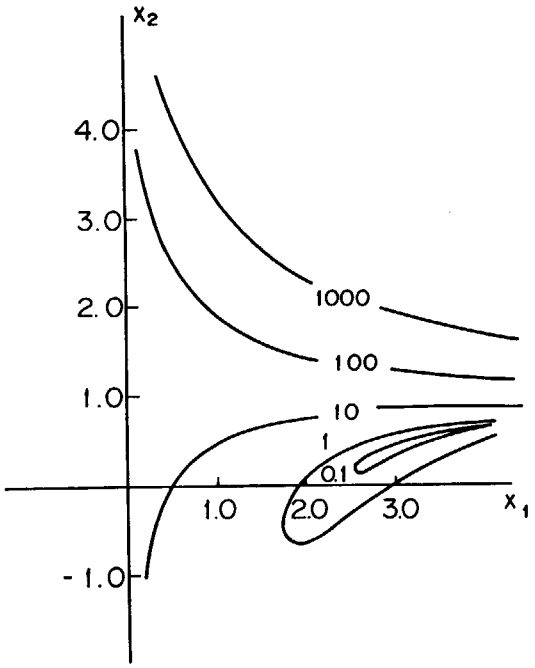


Fig. 15 Contours of Beale's function

Watson，多変数用として拡張 Rosenbrock の関数で，いずれも最小値が既知な関数である．この中で，Rosenbrock の関数は Fig. 14 にみられるように $x_2=x_1^2$ に沿った深い谷を持つことが知られており，また Beale の関数は，Fig. 15 で示されるように $x_2=1$ に沿った険しい谷を持つ最小化問題用の関数としては質の悪い関数である．Fig. 16 および Fig. 17 には，それらの谷の様子を立体的にみるために，EASY. 3 D[34] という三次元等高線プロッターを用いて描いた図が示されている．

また Watson の関数は，微分方程式

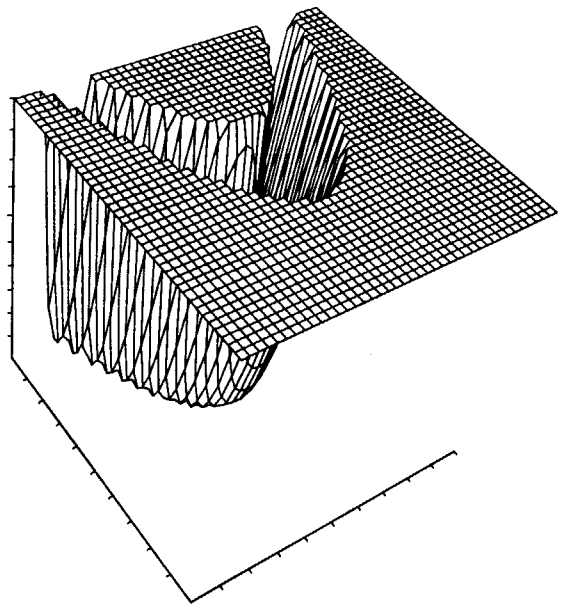


Fig. 16 Three dimensional shape of Rosenbrock's function

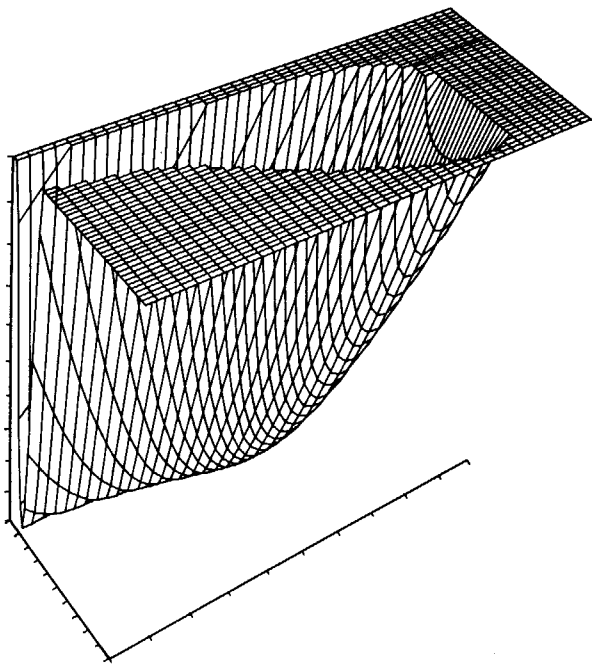


Fig. 17 Three dimensional shape of Beale's function

$$\frac{dz}{dy}=z^2+1,\quad z(0)=0 \tag{4-11}$$

の解を5次の多項式

$$z(y)=\sum_{j=1}^{6} x_j y^{j-1}$$

$$z(0)=x_1$$

に仮定し，選ばれた 30 点における残差の2乗和が最小になるように係数 x_j, $j=1,\cdots,6$ を決定する問題で，(4-11)の解は，$z=\tan y$ になることから多項式近似が適当でなく，この関数による問題は比較的困難と言われている．また関数 8. は，y_i が正規分布しているものとしてその平均と分散を求める問題である．

TABLE 11 Tested routines

ROUTINE	METHOD
VA 01 A	Variable metric (V. M.)
VA 06 A	〃
VA 08 A	Conjugate gradient (C. G.)
VA 09 A	V. M.
VA 10 A	〃
MINIM	Gradient & Newton
CGD	C. G.
FPD	V. M.
ALSIM*	Simplex
ALPS*	Pattern search
ALAG	Accelerated gradient
ALPART	Parallel tangent
ALCODR	Conjugate direction
ALVAM	V. M.
ROTAX*	Axes rotation
SPASE*	Spherical search

* direct search

TABLE 11 は，今回ベンチマーク・テストを実施したプログラム名とその解法を示したものである．表に示すように，VA シリーズの中に同じ可変計量法のプログラムがあるが，これは前に説明したように計量行列の更新の方法に相違があるからで，例えば VA 06 A は Powell の方法[36]，VA 09 A と VA 10 A は同じ Fletcher の方法[37]を用いているが，前者は目的関数の傾斜ベクトル計算ルーチンを必要としているのに対し，後者は傾斜ベクトルの計算を階差近似により行なっている違いがある．

テストは，主に前記 1～9 のテスト関数に対する各プログラムの安定性と収束効率を調べることを目指し，このため収束効率 C. E. を次式で定義した．

$$\text{C. E.}=\ln\frac{|\text{FO}-\text{FM}|}{|\text{FF}-\text{FM}|}\Big/T \tag{4-10}$$

但し，FO ; function value at starting point.
FM ; known minimum of objective function
FF ; function value at converged point
T ; elapsed time

である．また，出発点は，以下の範囲内で，各問題に対して 10 点ずつランダムに生成された．

1. $x_1, x_2\in[-100, 100]$
2. 〃
3. $x_1\in[-10, 10]$, $x_2\in[0, 20]$, $x_3\in[-100, 100]$
4. $x_i\in[-1, 1]$, $i=1,\cdots,4$
5. $x_i\in[-100, 100]$, $i=1,\cdots,6$
6. $x_i\in[-100, 100]$, $i=1,\cdots,4$
7. 〃

TABLE 12 Generated starting points

(Rosenbrock and Beale)

Run No.	x_1	x_2
1	67.673	33.37
2	−86.034	20.627
3	13.527	−18.139
4	27.855	16.69
5	48.636	−21.7
6	−4.22	79.53
7	74.57	64.41
8	78.88	−82.36
9	23.25	−39.07
10	−8.31	0.49

(Box)

Run No.	x_1	x_2	x_3
1	6.37	13.34	27.17
2	2.06	19.31	2.77
3	−3.06	7.97	−46.24
4	−5.36	13.36	84.00
5	3.78	3.71	82.50
6	−8.34	5.58	85.57
7	9.31	4.39	25.57
8	−5.89	1.64	28.52
9	3.38	3.61	−87.45
10	−4.99	5.25	−35.13

(Powell and Wood)

Run No.	x_1	x_2	x_3	x_4
1	63.67	33.37	27.17	62.67
2	98.07	2.77	−65.99	−20.90
3	−76.24	−95.18	45.02	−73.23
4	1.07	50.46	93.08	−95.63
5	22.34	−13.29	−80.78	34.91
6	−28.73	−25.28	−93.11	34.91
7	32.36	−32.05	−83.56	−12.88
8	−19.78	90.23	−80.57	−95.49
9	38.98	13.97	−66.95	72.49
10	41.66	38.22	54.43	−17.46

(Gauss)

Run No.	x_1	x_2	x_3
1	1.037	1.33	0.272
2	0.606	1.981	0.0277
3	0.0942	0.797	−0.462
4	−0.136	1.336	0.84
5	0.778	0.371	0.825
6	−0.434	0.558	0.856
7	1.331	0.439	0.256
8	−0.189	0.164	0.285
9	0.738	0.361	−0.874
10	−0.099	0.525	−0.351

8. $x_1 \in [-0.6, 1.4]$, $x_2 \in [0, 2]$, $x_3 \in [-1, 1]$

9. 1. に準ず.

TABLE 12 には生成された出発点の値が各問題ごとに示されている. 但し, Enzyme の問題に対する出発点には Powell と Wood の出発点を 1/100 にした値を用いた.

TABLE 12 (continued) Generated starting points

(Watson)*

Run No.	x_5	x_6
1	−68.48	25.10
2	−21.85	77.03
3	−58.92	−14.68
4	−39.70	66.75
5	81.96	31.68
6	−16.57	−40.90
7	−99.26	78.93
8	−82.43	39.14
9	42.30	−63.27
10	54.01	55.80

(*) $x_1 \sim x_4$ are the same with Powell and Wood

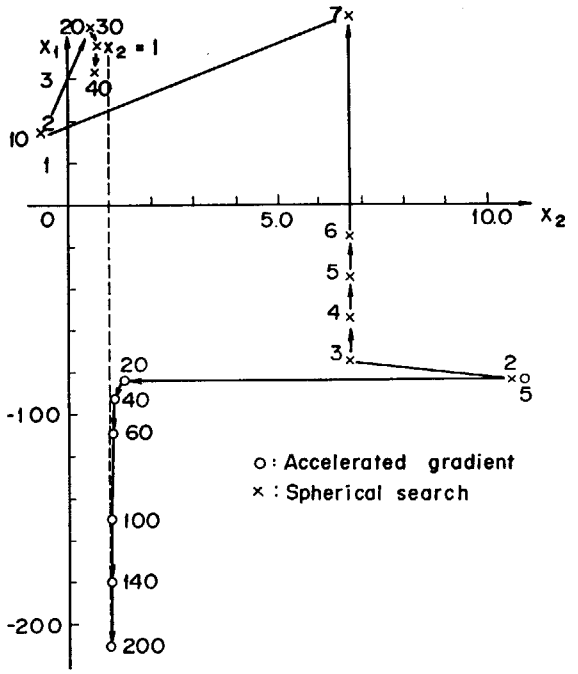


Fig. 18 Trajectories of searching process of Run No. 2 for Beale's function by ALAG and SPASE

TABLE 13〜TABLE 20 には, ROTAX と SPASE を除く 14 個のルーチンのベンチマーク・テストの結果が示されている. なお, ROTAX と SPASE の結果については, 既に参考文献 8) と 21) で報告されているので, ここでは省略した.

TABLE 13 は Rosenbrock の関数に対する計算結果で, この中で, RUN No. の数字は, 生成した 10 個の出発点の番号と対応し, ITN は反復回数, IFN は関数計算ルーチンの呼びだし回数, F-final は収束した点における目的関数値, T は要した計算時間 (単位は秒) である. 表からわかるように, Rosenbrock の関数に対してはほぼ全てのルーチンが最適点へ収束している. また全体的に要した計算時間も少なく, とくに FPD が速かった.

TABLE 14 は Beale の関数に対する結果であるが, この

TABLE 13 Computational results for Rosenbrock's function

Program	Run No.	ITN	IFN	F-final	T
VA 01 A	1	122	175	0.3932−18*	0.781
	2	155	223	0.7398−18	0.931
	3	106	190	0.8700−17	0.662
	4	67	110	0.7082−18	0.421
	5	98	130	0.4211−18	0.575
	6	143	225	0.3989−18	0.900
	7	153	203	0.1457−16	0.879
	8	202	250	0.4570−18	1.172
	9	700	1,341	0.6685−16	4.273
	10	48	91	0.1433−16	0.307
VA 06 A	1	98	98	0.9329−18	0.874
	2	163	163	0.5209−21	1.322
	3	55	55	0.2649−23	0.390
	4	100	100	0.2235−20	0.711
	5	55	55	0.2247−20	0.399
	6	127	127	0.1925−18	0.879
	7	157	157	0.7451−18	1.093
	8	105	105	0.9933−16	0.741
	9	49	49	0.2431−20	0.335
	10	77	77	0.9195−23	0.552
VA 08 A	1	71	219	0.2592−26	0.509
	2	63	204	0.4666−21	0.432
	3	35	113	0.584−24	0.248
	4	59	182	0.4617−19	0.389
	5	25	90	0.525−24	0.176
	6	81	242	0.248−17	0.523
	7	87	275	0.189−16	0.570
	8	27	106	0.145−19	0.197
	9	33	108	0.743−21	0.223
	10	29	90	0.205−28	0.203
VA 09 A	1	363	483	0.135−22	2.287
	2	381	510	0.240−27	2.407
	3	121	164	0.160−26	0.745
	4	186	254	0.339−23	1.177
	5	273	364	0.165−24	1.741
	6	91	131	0.150−23	0.587
	7	378	507	0.770−23	2.386
	8	365	479	0.109−22	2.297
	9	177	236	0.163−21	1.084
	10	90	118	0.197−28	0.548
VA 10 A	1	304	1,393	0.551−20	2.061
	2	413	1,852	0.39−19	2.796
	3	120	578	0.132−19	0.818
	4	172	779	0.16−20	1.166
	5	274	1,247	0.4−19	1.786
	6	37	200	0.123−19	0.256
	7	354	1,588	0.363−19	2.343
	8	398	1,781	0.393−19	2.599
	9	191	882	0.397−19	1.248
	10	99	457	0.302−19	0.655

Program	Run No.	ITN	IFN	F-final	T
MINIM	1	155	212	0.123−29	0.988
	2	192	267	0.578−27	1.204
	3	56	76	0.438−21	0.353
	4	91	124	0.999−26	0.566
	5	129	178	0.107−25	0.804
	6	51	75	0.708−22	0.325
	7	171	235	0.104−22	1.092
	8	179	249	0.0	1.099
	9	80	109	0.138−19	0.492
	10	47	64	0.203−21	0.291
CGD	1	75	180	0.372−21	0.453
	2	58	139	0.496−11	0.334
	3	21	56	0.625−22	0.132
	4	67	156	0.141−13	0.391
	5	22	55	0.303−13	0.139
	6	109	272	0.322−13	0.636
	7	100	257	0.135−12	0.595
	8	19	61	0.82−13	0.122
	9	12	40	0.678−19	0.078
	10	15	40	0.462−20	0.093
FPD	1	38	184	0.512−29	0.284
	2	41	208	0.0	0.294
	3	20	69	0.493−29	0.140
	4	40	196	0.0	0.285
	5	22	109	0.264−29	0.172
	6	55	313	0.0	0.404
	7	49	254	0.0	0.335
	8	18	81	0.153−26	0.131
	9	14	125	0.209−30	0.113
	10	16	107	0.211−30	0.125
ALSIM	1	143	668	0.4962−16	0.854
	2	118	475	0.9057−17	0.475
	3	69	336	0.2267−16	0.414
	4	125	517	0.2883−16	0.517
	5	121	505	0.2663−16	0.674
	6	153	697	0.4025−16	0.872
	7	144	568	0.1714−16	0.797
	8	90	500	0.3524−16	0.585
	9	96	505	0.6660−16	0.506
	10	42	254	0.2525−16	0.296
ALPS	1	680	1,784	0.3274−12	2.450
	2	522	1,371	0.5869−9	1.878
	3	238	772	0.7372−9	0.885
	4	221	706	0.3674−11	0.812
	5	88	390	0.1828−9	0.350
	6	545	1,503	0.1455−12	2.005
	7	701	1,765	0.1462−10	2.537
	8	218	705	0.4547−11	0.831
	9	133	486	0.8062−10	0.512
	10	204	702	0.3274−12	0.748

Program	Run No.	ITN	IFN	F-final	T
ALAG	1	350	1,015	0.7744−19	1.870
	2	351	968	0.7303−15	1.783
	3	110	354	0.4477−18	0.610
	4	307	1,073	0.3658−18	1.753
	5	130	489	0.1338−16	0.782
	6	472	1,283	0.1906−13	2.215
	7	389	1,134	0.2491−18	1.959
	8	141	605	0.1104−16	1.044
	9	100	317	0.4842−18	0.565
	10	272	924	0.1717−14	1.481
ALPART	1	661	1,229	0.1301−9	2.906
	2	997	2,000	0.1959−1	4.277
	3	438	1,050	0.2723−6	2.111
	4	873	2,000	0.2563−2	3.941
	5	833	1,968	0.3274−6	3.893
	6	949	2,000	13.435	4.083
	7	576	1,145	0.2289−6	2.373
	8	717	1,717	0.3505−6	3.163
	9	623	1,475	0.4378−7	2.759
	10	529	1,264	0.1100−6	2.354
ALCODR	1	366	948	0.3249−16	1.716
	2	433	1,061	0.5590−15	1.865
	3	141	504	0.5366−15	0.733
	4	278	818	0.7431−21	1.369
	5	88	384	0.1520−19	0.532
	6	542	1,292	0.4253−15	2.387
	7	461	1,125	0.1620−19	2.099
	8	85	385	0.5794−18	0.557
	9	142	489	0.6616−16	0.809
	10	133	473	0.1314−16	0.719
ALVAM	1	485	942	0.2624−15	2.944
	2	600	1,192	0.2220−18	3.532
	3	425	937	0.3758−15	4.239
	4	230	473	0.5428−15	1.464
	5	355	703	0.2434−17	2.072
	6	195	458	0.2924−18	1.481
	7	745	1,494	0.6543−15	5.489
	8	755	1,500	0.7200−15	5.368
	9	190	400	8.3423−15	1.229
	10	130	315	0.1780−18	1.029

* should be read as 0.3932×10^{-18}

TABLE 14 Computational results for Beale's function

Program	Run No.	ITN	IFN	F-final	T
VA 01 A	1	133	221	0.1631−18	0.891
	2×	1,000	1,035	0.453	5.939
	3	38	65	0.2777−18	0.250
	4	62	109	0.8899−18	0.412
	5	111	202	0.1754−18	0.722
	6×	6	8	7.162	0.04
	7	193	326	0.2675−18	1.215
	8	162	213	0.6845−17	0.96
	9	58	84	0.6399−17	0.371
	10×	1,000	1,048	0.453	5.944
VA 06 A	1	150	150	0.3899−16	1.186
	2×	1,000	1,000	0.452	7.514
	3	94	94	0.1573−17	0.726
	4	115	115	0.1355−24	0.894
	5	121	121	0.1689−17	0.934
	6×	1,000	1,000	0.452	7.796
	7	160	160	0.4510−20	1.237
	8	182	182	0.2320−22	1.356
	9	134	134	0.3832−20	0.990
	10×	1,000	1,000	0.452	7.231
VA 08 A	1	195	563	0.246−30	1.311
	2×	12	84	0.468	0.11
	3×	272	798	0.475	1.813
	4	57	206	0.567−23	0.405
	5	161	498	0.0	1.103
	6×	4	33	7.162	0.042
	7×	5	51	0.168+5	0.059
	8	124	380	0.198−17	0.858
	9	19	100	0.434−16	0.159
	10×	278	826	0.474	1.855
VA 09 A	1×	12	27	0.4449	0.104
	2×	16	28	0.4573	0.111
	3×	13	32	0.4222	0.096
	4×	12	30	0.4357	0.09
	5×	6	25	0.4428	0.052
	6×	7	22	7.182	0.058
	7×	22	44	0.446	0.151
	8×	20	40	0.446	0.143
	9×	10	39	0.433	0.082
	10	23	44	0.561−26	0.158
VA 10 A	1×	14	116	0.4446	0.13
	2×	11	95	0.4577	0.091
	3	67	464	0.552−22	0.487
	4	86	569	0.849−22	0.621
	5	107	694	0.145−21	0.789
	6×	9	89	7.164	0.081
	7×	20	148	0.4453	0.156
	8×	17	132	0.4461	0.133
	9×	16	117	0.4331	0.124
	10×	352	2,000	0.4521	2.478
MINIM	1	11	33	0.735−24	0.099
	2×	596	845	0.452	4.08
	3	8	19	0.1817−20	0.061
	4	24	49	0.564−26	0.168
	5×	474	667	0.452	3.143
	6×	577	807	0.452	3.904
	7	12	36	0.568−28	0.089
	8	13	46	0.149−17	0.098
	9×	445	628	0.452	2.965
	10	10	19	0.461−23	0.074

Program	Run No.	ITN	IFN	F-final	T
CGD	1×	841	2,000	23.13	5.256
	2×	6	23	0.47	0.047
	3	20	52	0.493−31	0.131
	4	60	152	0.3−27	0.372
	5×	846	2,000	0.2897	5.258
	6×	5	14	7.161	0.037
	7	108	289	0.986−30	0.71
	8×	841	2,000	0.355+4	5.393
	9×	281	679	0.473	1.835
	10×	163	407	0.473	1.083
FPD	1	50	503	0.645−26	0.458
	2×	211	2,000	0.452	1.809
	3	23	137	0.278−28	0.178
	4	33	253	0.103−27	0.28
	5	56	474	0.493−31	0.506
	6×	223	2,000	0.453	1.917
	7×	2	10	1.58+9	0.02
	8×	247	2,000	0.452	2.013
	9	13	46	0.986−31	0.106
	10×	170	2,000	0.452	1.625
ALSIM	1×	745	1,996	0.453	0.973
	2×	790	1,999	0.452	0.974
	3	55	514	0.574−11	0.156
	4	65	500	0.0	0.152
	5	60	478	0.409−10	0.146
	6	65	522	0.256−11	0.163
	7	80	600	0.512−14	0.181
	8×	710	1,999	0.453	0.916
	9	100	603	0.0	0.195
	10×	735	1,996	0.453	0.943
ALPS	1	130	494	0.659−17	0.395
	2×	630	1,999	0.452	1.653
	3	70	321	0.133−15	0.244
	4	100	538	0.107−15	0.372
	5	120	490	0.499−16	0.381
	6×	620	1,998	0.452	1.740
	7	135	540	0.722−16	0.435
	8	135	524	0.540−16	0.399
	9	85	376	0.588−16	0.284
	10×	625	1,999	0.452	1.740
ALAG	1	400	1,209	0.109−21	1.168
	2×	235	842	0.459	0.738
	3	200	716	0.198−20	0.696
	4	305	911	0.156−21	0.868
	5	490	1,550	0.649−22	1.428
	6×	620	2,000	0.496	1.589
	7	470	1,867	0.209−21	1.720
	8	405	1,257	0.233−21	1.119
	9	190	607	0.189−21	0.608
	10×	660	2,000	0.540	1.889
ALPART	1×	1,565	2,000	0.284	6.001
	2×	15	89	0.470	0.162
	3	240	397	0.897−14	1.068
	4	320	525	0.139−14	1.428
	5	885	1,184	0.246−15	3.467
	6×	10	82	7.162	0.146
	7	1,250	1,631	0.761−14	4.829
	8	885	1,213	0.143−13	3.432
	9	245	384	0.552−17	1.017
	10×	1,355	1,783	0.469	4.592
ALCODR	1×	865	2,000	0.47	1.566
	2×	865	〃	0.464	1.507
	3×	915	〃	7.338	1.634
	4×	895	〃	0.461	1.516
	5×	890	〃	7.338	1.588
	6×	920	〃	0.494	1.551
	7×	910	〃	0.487	1.574
	8×	895	〃	7.327	1.645
	9×	905	〃	7.333	1.649
	10	65	343	0.244−20	0.299
ALVAM	1	570	880	0.306−17	2.521
	2×	1,345	2,000	0.453	5.333
	3	140	251	0.198−19	0.838
	4	275	442	0.193−19	1.385
	5	405	646	0.361−16	1.987
	6×	1,265	1,998	0.455	5.598
	7	685	1,056	0.659−20	3.047
	8	675	1,046	0.135−17	3.032
	9	120	215	0.120−16	0.731
	10×	1,310	1,998	0.454	5.398

Mark × means the computation has no convergence for the Run No.

TABLE 15 Computational results for Box's function

Program	Run No.	ITN	IFN	F-final	T
VA 01 A	1	37	45	0.4877−19	0.519
	2	24	41	0.4618−21	0.406
	3	82	93	0.1098−20	1.093
	4	111	140	0.1869−19	1.547
	5	102	126	0.5725−22	1.388
	6	111	141	0.2099−17	1.549
	7	34	43	0.9136−20	0.459
	8	47	71	0.3037−21	0.698
	9	109	136	0.2060−19	1.458
	10	72	95	0.1550−20	1.011
VA 06 A	1	55	55	0.3581−18	0.795
	2	37	37	0.4581−19	0.500
	3	36	36	0.5828−14	0.490
	4	54	54	0.2244−19	0.764
	5	83	83	0.1278−16	1.158
	6	83	83	0.3801−17	1.156
	7	19	19	0.3418−26	0.258
	8	49	49	0.1357−16	0.676
	9×	61	61	0.0756	1.168
	10	41	41	0.2823−21	0.571

Program	Run No.	ITN	IFN	F-final	T
VA 08 A	1*				
	2	94	215	0.183−22	1.786
	3*				
	4*				
	5*				
	6*				
	7*				
	8*				
	9*				
	10*				
VA 09 A	1	21	25	0.354−27	0.321
	2	25	34	0.179−23	0.370
	3	15	26	0.494−23	0.252
	4	38	43	0.733−1	
	5*				
	6*				
	7	24	34	0.928−24	0.392
	8*				
	9	44	57	0.121−28	0.663
	10*				
VA 10 A	1	18	122	0.166−29	0.736
	2	25	154	0.151−23	0.949
	3	35	215	0.144−23	1.280
	4	43	260	0.151−23	1.604
	5	31	201	0.149−23	1.217
	6*				
	7	23	154	0.152−23	0.938
	8*				
	9	32	216	0.114−23	1.373
	10	17	123	0.433−20	0.756
MINIM	1	16	49	0.240−14	0.36
	2	7	17	0.548−24	0.178
	3	28	111	0.319−18	0.833
	4	20	27	0.291−23	0.316
	5	14	22	0.548−20	0.255
	6*				
	7	24	63	0.286−17	0.571
	8	19	46	0.120−13	0.36
	9	14	22	0.988−17	0.256
	10*				
CGD	1	11	38	0.9648−32	0.334
	2	44	101	0.5826−22	0.947
	3	37	123	0.2677−11	0.865
	4	44	106	0.1064−21	0.868
	5	15	59	0.3416−32	0.429
	6	46	112	0.6391−22	0.883
	7	7	22	0.5859−19	0.164
	8	15	47	0.8341−33	0.351
	9	15	59	0.4824−32	0.445
	10	41	137	0.3332−11	0.892

Program	Run No.	ITN	IFN	F-final	T
FPD	1	8	35	0.2109−28	0.279
	2	16	86	0.1808−30	0.639
	3	66	355	0.4942−39	2.091
	4	28	148	0.1319−26	0.992
	5×	18	105	0.756−1	0.688
	6	25	185	0.8348−18	1.096
	7	7	35	0.1041−26	0.26
	8	17	96	0.8088−12	0.606
	9	12	68	0.3002−22	0.477
	10	71	408	0.9621−40	2.258
ALSIM	1	165	1,998	0.0	11.838
	2	80	471	0.4751−17	2.752
	3	345	1,998	0.0	18.111
	4	130	1,998	0.2664−32	11.432
	5	145	641	0.9270−17	3.787
	6	245	1,043	0.1199−16	6.173
	7	150	1,999	0.3126−36	11.648
	8	165	913	0.3546−16	5.290
	9	170	762	0.8321−17	4.373
	10	360	1,999	0.0	20.243
ALPS	1*				
	2	130	882	0.1333−32	4.607
	3	70	742	0.7559−1	8.148
	4	175	1,093	0.1420−71	21.381
	5*				
	6	210	984	0.4098−18	5.351
	7	60	650	0.7559−1	8.061
	8*				
	9	120	1,205	0.0	23.242
	10	130	1,317	0.0	26.922
ALAG	1	610	110	0.1470−25	3.746
	2	1,441	435	0.2081−20	9.014
	3	2,000	650	0.1012−18	12.843
	4	1,746	595	0.5425−15	11.055
	5	1,171	450	0.1279−16	7.450
	6	1,391	400	0.7711−16	8.722
	7	2,000	255	0.3045−24	12.960
	8	1,560	545	0.1566−24	9.550
	9	1,999	675	0.3403−14	12.630
	10	1,115	410	0.3399−16	6.920
ALPART	1	140	249	0.2138−16	2.288
	2	465	687	0.1043−12	6.424
	3	975	1,998	0.1724−10	16.688
	4	585	833	0.4703−15	7.649
	5	870	1,227	0.1712−13	11.054
	6	690	978	0.4392−13	8.910
	7	240	377	0.8024−15	3.468
	8	735	1,040	0.2902−12	9.627
	9	835	1,183	0.4141−14	10.600
	10	845	1,199	0.88−14	11.364

Program	Run No.	ITN	IFN	F-final	T
ALCODR	1	200	691	0.1375−26	4.052
	2*				
	3	205	1,153	0.5115−32	6.899
	4*				
	5*				
	6	100	868	0.7559−1	13.063
	7	160	666	0.3020−25	3.929
	8	500	2,000	0.7559−1	17.304
	9	300	987	0.1648−19	5.805
	10	195	1,297	0.8961−22	7.530
ALVAM	1	25	48	0.3927−17	0.804
	2	1,045	1,997	0.1442−3	30.754
	3	1,120	1,998	0.6345−6	30.708
	4	1,060	1,998	0.5971−4	30.996
	5	45	91	0.1372−18	1.471
	6×	45	94	0.0756	3.488
	7	60	134	0.2411−24	2.287
	8	1,050	1,999	0.1319−4	31.917
	9	55	134	0.2594−16	2.276
	10	1,095	1,998	0.8409−9	32.361

The mark × means that the computation had no convergence

* Computation has stopped due to the exponential argument error

TABLE 16 Computational results for Enzyme's function

Program	Run No.	ITN	IFN	F-final	T
VA 01 A	1	27	54	0.3075−3	0.36
	2	30	60	〃	0.393
	3×	132	265	0.1594−2	1.657
	4×	100	208	〃	1.329
	5	34	69	0.3075−3	0.454
	6	33	66	〃	0.446
	7	32	64	〃	0.420
	8×	1000	1,138	0.1028−2	9.958
	9	29	59	0.3075−3	0.376
	10	30	63	〃	0.406
VA 06 A	1	79	79	0.3075−3	1.064
	2	66	66	〃	0.869
	3×	326	326	0.4242−3	4.368
	4×	256	256	〃	3.514
	5	43	43	0.3075−3	0.575
	6	124	124	〃	1.640
	7	70	70	〃	0.939
	8×	95	95	0.1594−2	1.258
	9	44	44	0.3075−3	0.569
	10	84	84	〃	1.131

Program	Run No.	ITN	IFN	F-final	T
VA 08 A	1	129	306	0.30749−3	1.712
	2	93	235	〃	1.180
	3×	80	229	0.4242−3	1.102
	4×	145	443	〃	2.056
	5	63	156	0.30749−3	0.803
	6	78	203	〃	1.011
	7	49	119	〃	0.645
	8×	154	372	0.1594−2	1.915
	9	132	322	0.30749−2	1.644
	10×	401	1,446	0.9635−3	5.844
VA 09 A	1	39	41	0.30749−3	0.437
	2	339	436	〃	3.877
	3×	48	62	0.4242−3	0.568
	4×	75	138	0.1594−2	0.999
	5	41	44	0.30749−3	0.479
	6	57	72	〃	0.668
	7	278	354	〃	3.172
	8×			0.4242−3	
	9	43	50	0.30749−3	0.495
	10×	202	324	0.1626−2	2.44
VA 10 A	1	23	233	0.30749−3	0.488
	2×	54	479	0.1594−2	
	3×	38	266	0.4242−3	
	4×	47	329	〃	
	5	36	255	0.30749−3	0.502
	6	51	356	〃	0.726
	7×	66	512	0.1709−2	0.973
	8×	67	476	0.4242−3	
	9	33	230	0.30749−3	0.477
	10×	71	532	0.1626−2	
MINIM	1	14	24	0.30749−3	0.218
	2×	31	92	0.4242−3	
	3×	302	456	0.063	4.334
	4	29	79	0.30749−3	0.468
	5×	38	150	0.4242−3	
	6	12	21	0.30749−3	0.173
	7×	20	44	0.4242−3	
	8×	13	33	0.1594−2	
	9×	49	137	0.144	0.776
	10×	148	374	0.144	2.252
CGD	1×	12	29	0.446−3	0.168
	2×	994	2,000	35.15	
	3×	1,000	2,000	0.984−2	
	4×	333	773	0.122−2	
	5×	998	1,999	0.363−2	11.225
	6×	977	2,000	0.360−2	11.363
	7×	994	2,000	0.1788−2	11.67
	8×	942	2,000	0.1596−2	
	9×	32	68	0.456−2	0.422
	10×	987	2,000		

Program	Run No.	ITN	IFN	F-final	T
FPD	1	22	97	0.30749−3	0.399
	2×	35	145	0.4242−3	
	3×	34	153	〃	
	4×	77	540	〃	
	5	44	329	0.30749−3	1.012
	6×	45	310	0.3199−2	
	7×	67	457	0.159−2	
	8×	27	131	〃	
	9	26	162	0.30749−3	0.568
	10×	31	140	0.4242−3	
ALSIM	1	130	815	0.30749−3	1.527
	2	130	846	〃	1.519
	3×	145	813	0.1594−2	1.458
	4×	475	1,999	0.3555−1	3.751
	5	70	677	0.30749−3	1.215
	6	90	762	〃	1.396
	7	100	743	〃	1.357
	8×	170	932	0.1594−2	1.773
	9	125	839	0.30749−3	1.630
	10	90	682	〃	1.336
ALPS	1	325	1,511	0.30749−3	1.426
	2×	690	1,997	0.9524−3	2.083
	3×	645	1,977	0.4252−3	2.074
	4×	695	1,993	0.30749−3	2.118
	5	350	1,696	〃	1.621
	6	475	1,994	〃	1.984
	7×	465	1,895	0.4242−3	1.862
	8	515	1,993	0.30749−3	1.914
	9×	700	1,997	0.8799−3	2.043
	10	390	1,668	0.30749−3	1.545
ALAG	1	530	2,000	0.30749−3	3.324
	2×	290	1,301	0.7264−3	1.884
	3×	145	954	0.7723−3	1.330
	4×	145	831	0.1425−1	1.184
	5×	810	2,000	0.1028−2	3.669
	6	605	1,996	0.30749−3	3.419
	7×	410	2,000	0.1742−2	2.906
	8×	505	1,843	0.1084	2.863
	9	635	2,000	0.30749−3	3.319
	10×	155	793	0.4795−2	1.097
ALPART	1	1,130	1,550	0.30749−3	6.637
	2	1,515	1,997	0.5896−3	8.275
	3×	725	1,054	0.1594−2	4.452
	4	1,175	1,610	0.4242−3	6.666
	5	845	1,165	0.30749−3	4.912
	6×	1,570	1,998	0.1008−2	8.648
	7	1,510	1,996	0.3460−3	8.344
	8×	1,260	1,729	0.1594−2	6.764
	9	1,390	1,899	0.30749−3	7.713
	10	1,500	1,998	0.6052−3	8.275
ALCODR	1	310	1,101	0.30749−3	1.685
	2×	325	1,179	0.1594−2	1.748
	3×	355	2,001	0.6023−3	2.518
	4×	585	2,000	0.4471−3	2.704
	5	195	814	0.30749−3	1.159
	6	250	1,032	〃	1.488
	7×	195	1,810	0.2499−2	2.164
	8×	220	1,692	0.1684−2	2.180
	9	245	941	0.30749−3	1.376
	10×	480	1,993	0.4242−3	2.757
ALVAM	1	1,090	1,996	0.30749−3	12.670
	2	1,034	1,998	0.4252−3	12.503
	3×	700	1,996	0.6059−3	11.256
	4×	795	1,999	0.1813−2	12.005
	5	1,100	1,997	0.30749−3	12.875
	6	1,075	1,997	〃	13.409
	7×	795	1,998	0.1713−2	13.171
	8×	1,110	2,000	0.1594−2	12.641
	9	1,095	1,996	0.30749−3	12.605
	10×	710	1,997	0.3019−2	10.904

The mark × means that the computation had no convergence

TABLE 17 Computational results for Watson's function

Program	Run No.	ITN	IFN	F-final	T
VA 01 A	1	191	247	0.2288−2	3.976
	2	186	234	〃	3.811
	3	141	220	〃	3.332
	4	170	217	〃	3.496
	5	212	281	〃	4.521
	6	208	268	〃	4.248
	7	288	350	〃	5.741
	8	252	307	〃	5.003
	9	205	251	〃	4.044
	10	172	221	〃	3.553
VA 06 A	1	208	208	0.2288−2	5.166
	2	238	238	〃	5.985
	3	256	256	〃	6.302
	4	164	164	〃	4.149
	5	172	172	〃	4.422
	6	237	237	〃	6.022
	7	248	248	〃	6.203
	8	191	191	〃	4.620
	9	349	349	〃	8.660
	10	142	142	〃	3.511
VA 08 A	1×	880	2,000	0.6026−2	24.403
	2	876	2,000	0.2289−2	24.182
	3	553	1,304	0.22877−2	15.932
	4*				
	5	742	1,709	0.22877−2	20.989
	6	852	2,000	〃	24.619
	7×	875	2,000	0.1492−1	24.663
	8×	857	2,000	1.077	23.778
	9	770	1,782	0.22877−2	21.535
	10	814	1,869	〃	22.403

Program	Run No.	ITN	IFN	F-final	T
VA 09 A	1	108	125	0.22877−2	2.437
	2	97	114	〃	2.127
	3	111	149	〃	2.582
	4	104	118	〃	2.223
	5	105	119	〃	2.273
	6	102	115	〃	2.175
	7	115	131	〃	2.451
	8	74	86	〃	1.586
	9	96	110	〃	2.025
	10	110	129	〃	2.412
VA 10 A	1	75	697	0.22877−2	3.873
	2	72	660	〃	3.670
	3	64	615	〃	3.351
	4	70	662	〃	3.748
	5	72	660	〃	3.782
	6	58	560	〃	3.044
	7	75	707	〃	3.948
	8	55	511	〃	2.843
	9	59	553	〃	3.075
	10	81	753	〃	4.243
MINIM	1	29	30	0.22877−2	1,210
	2	29	30	〃	1,166
	3	29	30	〃	1,160
	4	27	28	〃	1,099
	5	29	30	〃	1,162
	6	29	30	〃	1,171
	7	30	31	〃	1,190
	8	25	26	〃	0.988
	9	23	24	〃	0.905
	10	28	29	〃	1.109
CGD	1×	945	1,999	0.1423−1	
	2×	937	2,000	0.1517−1	
	3×	966	2,000	0.637−1	
	4×	918	2,000	0.2544−2	
	5×	892	2,000	0.984−2	
	6×	945	2,000	0.401−1	
	7×	922	2,000	1.008	
	8×	907	2,000	0.743−2	
	9×	850	2,000	3.658	
	10×	892	2,000	0.42	
FPD	1	87	397	0.22877−2	5.906
	2	98	440	〃	4.549
	3	62	503	〃	4.449
	4	40	187	〃	3.159
	5	38	150	〃	1.652
	6	56	250	〃	2.569
	7	50	255	〃	6.095
	8	43	180	〃	3.127
	9	46	217	〃	2.338
	10	57	273	〃	2.744

Program	Run No.	ITN	IFN	F-final	T
ALSIM	1×	530	1,998	0.2397+2	12.585
	2	505	1,998	0.2288−2	13.004
	3	530	1,999	0.2671−2	12.846
	4	495	1,998	0.2289−2	12.359
	5×	515	1,998	0.2084+2	12.202
	6×	480	1,999	0.1782+2	12.113
	7×	500	1,999	0.1787+2	12.099
	8	490	1,999	0.2288−2	12.249
	9×	500	1,998	0.3627	12.465
	10	470	1,998	0.2288−2	12.025
ALPS	1×	555	1,995	0.6928	8.764
	2	415	1,883	0.22877−2	7.012
	3	470	1,890	0.2288−2	7.613
	4×	620	1,998	0.7154	8.814
	5×	570	1,993	0.2563	8.565
	6	445	1,911	0.2289−2	8.012
	7	420	1,890	0.2288−2	7.541
	8×	600	1,992	4.027	8.892
	9×	610	1,989	4.469	8.709
	10	490	1,921	0.22877−2	8.104
ALAG	1×	795	2,000	0.4772+1	10.382
	2×	835	2,000	0.1032+3	10.546
	3×	820	2,000	0.7770	10.625
	4×	865	1,998	0.1632+1	10.877
	5×	905	2,000	0.4948+1	10.499
	6×	855	2,000	0.4347	10.587
	7×	810	1,998	0.1897+2	10.258
	8×	815	2,000	0.1061+2	10.471
	9×	905	2,000	0.5557	10.523
	10×	880	2,000	0.3364+2	10.808
ALPART	1×	1,470	1,995	4.257	18.500
	2×	1,480	1,995	0.3237	17.724
	3×	1,490	1,995	0.4769−1	18.341
	4×	1,460	1,998	0.41−2	18.948
	5×	1,470	1,998	0.2205	18.459
	6×	1,465	1,995	0.1602−1	17.941
	7×	1,470	1,996	0.8326	17.995
	8×	1,510	1,996	7.828	17.955
	9×	1,470	1,994	0.2055−1	18.070
	10×	1,460	1,994	2.387	17.727
ALCODR	1×	755	2,000	0.2703	10.030
	2×	775	2,000	0.3653−1	9.990
	3×	830	2,000	0.7209	9.841
	4×	765	2,000	0.2561	9.975
	5×	770	2,000	14.646	9.938
	6×	845	2,000	0.2463	9.996
	7×	785	2,000	4.790	9.564
	8×	775	2,000	0.1453	9.715
	9×	815	2,000	17.09	9.811
	10×	790	2,000	1.317	9.823

Program	Run No.	ITN	IFN	F-final	T
ALVAM	1×	720	1,995	5.775	35.281
	2×	645	1,995	101.84	32.789
	3×	745	1,996	6.536	34.435
	4×	565	1,995	400.12	31.169
	5×	690	1,996	84.68	31.275
	6×	705	1,994	0.8693	32.705
	7×	695	1,994	11.749	31.383
	8×	670	1,996	58.425	30.884
	9×	745	1,997	33.573	31.054
	10×	630	1,995	63.280	31.138

∗ The square-root argument became negative
× The computation had no convergence

TABLE 18 Computational results for Powell's function

Program	Run No.	ITN	IFN	F-final	T
VA 01 A	1	171	247	0.2788−26	1.493
	2	209	303	0.2627−29	1.736
	3	253	325	0.2254−28	2.029
	4	239	310	0.2016−27	1.946
	5	200	281	0.8667−27	1.666
	6	202	272	0.1243−26	1.685
	7	193	273	0.2287−24	1.603
	8	256	329	0.6478−26	2.120
	9	202	289	0.2785−28	1.712
	10	158	230	0.6238−27	1.361
VA 06 A	1	73	73	0.4074−12	0.835
	2	97	97	0.1180−11	1.106
	3	97	97	0.1520−11	1.107
	4	91	91	0.1405−11	1.032
	5	85	85	0.1093−11	0.986
	6	98	98	0.6283−12	1.119
	7	89	89	0.5782−11	0.965
	8	92	92	0.1381−11	1.033
	9	88	88	0.1080−11	1.017
	10	88	88	0.1870−11	1.060
VA 08 A	1	81	242	0.451−10	0.683
	2	108	303	0.661−10	0.896
	3	69	217	0.332−9	0.582
	4	93	274	0.277−9	0.783
	5	73	224	0.361−9	0.64
	6	85	246	0.115−9	0.717
	7	108	320	0.2799−10	0.898
	8	69	208	0.688−9	0.589
	9	81	222	0.318−9	0.666
	10	80	236	0.778−10	0.708
VA 09 A	1	123	137	0.61−30	1.122
	2	136	153	1.049−27	1.212
	3	113	134	4.277−24	1.031
	4	114	129	6.638−22	1.012
	5	117	140	3.938−26	1.053
	6	101	117	2.507−20	0.906
	7	70	83	5.655−14	0.643
	8	112	133	1.267−22	1.027
	9	118	139	3.947−26	1.03
	10	123	135	6.74−23	1.099
VA 10 A	1	59	519	9.405−26	0.608
	2	158	2,000	1.089−29	1.755
	3	73	658	1.905−22	0.742
	4	61	530	0.464−23	0.602
	5	94	933	3.732−39	0.953
	6	95	978	9.192−37	0.999
	7	61	552	3.603−20	0.615
	8	67	554	2.735−21	0.665
	9	69	612	6.667−24	0.712
	10	71	747	0.339−31	0.76
MINIM	1	25	26	0.475−12	0.279
	2	31	32	3.409−11	0.326
	3	30	31	8.744−11	0.309
	4	30	31	8.978−11	0.313
	5	30	31	3.578−11	0.307
	6	30	31	1.166−10	0.305
	7	30	31	2.771−11	0.308
	8	31	32	6.313−11	0.309
	9	30	31	3.631−11	0.307
	10	29	30	5.505−11	0.296
CGD	1	119	293	8.773−9	0.969
	2	72	213	0.36−10	0.606
	3	51	156	7.415−9	0.426
	4	101	278	3.957−9	0.824
	5	116	275	1.923−8	0.928
	6	46	145	2.578−9	0.387
	7	66	194	2.806−9	0.542
	8	121	297	1.133−8	0.941
	9	76	218	1.967−10	0.601
	10	91	245	1.047−11	0.731
FPD	1	49	249	7.424−25	0.56
	2	71	492	5.059−25	0.818
	3	49	265	9.474−26	0.529
	4	58	301	2.164−26	0.616
	5	53	317	3.167−22	0.547
	6	39	183	1.309−22	0.393
	7	35	181	1.038−20	0.357
	8	54	272	9.418−27	0.53
	9	59	287	1.009−27	0.587
	10	72	519	1.167−27	0.788
ALSIM	1	465	1,998	0.9105−62	2.716
	2	500	1,998	0.4076−61	2.670
	3	465	1,997	0.4105−63	2.613
	4	490	1,998	0.8444−63	2.650
	5	485	1,998	0.3348−57	2.772
	6	530	1,999	0.3677−62	2.753
	7	505	1,998	0.9134−62	2.690
	8	425	1,998	0.7834−60	2.554
	9	465	1,998	0.1638−65	2.704
	10	475	1,998	0.1967−61	2.685

Program	Run No.	ITN	IFN	F-final	T
ALPS	1	595	1,993	0.2380−9	0.924
	2	390	1,773	0.1734−13	0.683
	3	650	1,994	0.4177−9	0.989
	4	615	1,994	0.9341−9	0.947
	5	550	1,993	0.9205−9	0.873
	6	565	1,993	0.3301−9	0.904
	7	560	2,000	0.3720−8	0.889
	8	575	1,994	0.1392−8	0.899
	9	565	1,996	0.2156−9	0.884
	10	550	1,995	0.3384−9	0.824
ALAG	1	850	2,000	0.3721−6	2.266
	2	855	2,000	0.3269−8	2.118
	3	690	2,000	0.5244−11	2.103
	4	835	2,000	0.2087−4	2.192
	5	720	1,996	0.2230−7	2.157
	6	710	2,000	0.7670−9	2.220
	7	645	2,000	0.4987−11	2.133
	8	765	2,000	0.6397−7	2.194
	9	835	2,000	0.1289−5	2.176
	10	675	1,994	0.3005−12	2.037
ALPART	1	975	1,367	0.4844−10	4.271
	2	375	591	0.8773−10	1.760
	3	650	932	0.2206−9	2.813
	4	1,040	1,480	0.4835−9	4.528
	5	1,470	1,999	0.2683−9	6.265
	6	1,035	1,497	0.1384−9	4.486
	7	1,470	1,998	0.1549−8	6.325
	8	1,295	1,999	0.6215−9	5.905
	9	1,240	1,763	0.7178−9	5.424
	10	1,465	1,997	0.8487−10	6.339
ALCODR	1	565	2,000	0.3133−20	1.657
	2	565	2,000	0.1401−21	1.533
	3	670	2,000	0.7427−19	1.651
	4	675	2,000	0.1080−17	1.702
	5	580	2,000	0.3210−18	1.627
	6	670	2,000	0.2069−16	1.785
	7	505	2,000	0.3513−18	1.615
	8	600	2,000	0.8320−18	1.580
	9	695	2,000	0.6430−16	1.684
	10	605	2,000	0.6094−20	1.654
ALVAM	1	870	1,998	0.5128−8	6.934
	2	870	1,998	0.6898−8	6.995
	3	895	1,996	0.6904−8	6.813
	4	875	1,996	0.7696−8	7.001
	5	875	1,997	0.7471−8	7.155
	6	890	1,999	0.2755−8	7.045
	7	890	1,998	0.5178−8	6.913
	8	890	1,997	0.5106−8	6.997
	9	875	1,996	0.5733−8	6.999
	10	880	1,996	0.3557−8	6.932

TABLE 19 Computational results for Wood's function

Program	Run No.	ITN	IFN	F-final	T
VA 01 A	1	155	201	0.2398−17	1.315
	2	165	217	0.7979−18	1.336
	3	164	189	0.3592−16	1.341
	4	259	292	0.1330−16	2.044
	5	175	242	0.5185−17	1.406
	6	227	290	0.4528−19	1.791
	7	141	170	0.6573−17	1.095
	8	298	356	0.1925−18	2.375
	9	189	258	0.4172−16	1.558
	10	136	181	0.5563−16	1.082
VA 06 A	1	71	71	0.3615−19	0.856
	2	114	114	0.1535−17	1.275
	3	105	105	0.1507−17	1.218
	4	83	83	0.6834−21	0.949
	5	140	140	0.2929−20	1.557
	6	149	149	0.2719−19	1.710
	7	130	130	0.1021−21	1.470
	8	167	167	0.2192−19	1.899
	9	139	139	0.1600−21	1.573
	10	127	127	0.5628−26	1.460
VA 08 A	1	122	308	2.052−14	1.055
	2	101	257	6.661−16	0.842
	3	165	380	4.352−18	1.345
	4	98	260	2.253−13	0.833
	5	126	301	4.089−15	1.052
	6	197	464	1.255−18	1.657
	7	50	137	8.972−11	0.456
	8	125	318	8.673−18	1.093
	9	170	417	1.329−11	1.432
	10	149	361	2.047−17	1.244
VA 09 A	1	111	145	8.116−19	1.04
	2	103	126	1.816−18	0.913
	3	95	116	1.764−17	0.858
	4	103	127	4.681−20	0.935
	5	93	117	2.573−21	0.848
	6	80	105	6.405−17	0.73
	7	74	97	2.181−20	0.715
	8	76	99	1.277−16	0.706
	9	78	95	5.186−22	0.73
	10	86	104	1.798−18	0.81
VA 10 A	1	82	569	1.186−19	0.887
	2	90	625	1.184−19	0.943
	3	88	602	5.451−20	0.936
	4	109	763	1.964−20	1.15
	5	76	535	1.186−19	0.795
	6	83	586	1.181−19	0.866
	7	79	560	1.183−19	0.827
	8	79	570	1.187−19	0.845
	9	74	529	1.166−19	0.796
	10	75	534	8.313−20	0.796

Program	Run No.	ITN	IFN	F-final	T
MINIM	1	29	31	2.906−19	0.351
	2	29	32	2.470−19	0.350
	3	35	40	1.859−17	0.386
	4	28	52	1.741−17	0.309
	5	40	47	7.514−19	0.443
	6	40	46	1.242−17	0.424
	7	40	46	2.052−18	0.422
	8	41	49	2.382−19	0.452
	9	39	45	3.939−18	0.431
	10	26	27	2.585−17	0.283
CGD	1	50	138	4.445−21	0.426
	2	107	258	1.941−12	0.833
	3	97	222	2.101−13	0.762
	4	70	187	2.066−18	0.567
	5	85	215	6.277−23	0.681
	6	86	244	7.605−15	0.689
	7	32	103	6.563−14	0.259
	8	77	203	1.349−16	0.633
	9	90	234	3.742−16	0.727
	10	92	237	1.030−13	0.725
FPD	1	62	402	2.291−29	0.686
	2×	2	14	2.628+11	0.031
	3	49	256	1.896−22	0.495
	4	76	574	1.522−21	0.895
	5	36	148	1.720−21	0.365
	6	19	54	5.079−28	0.18
	7	77	421	3.627−26	0.781
	8	54	287	3.344−22	0.555
	9	36	166	1.850−19	0.381
	10	102	919	2.784−24	1.155
ALSIM	1	285	1,613	0.1836−15	2.203
	2	265	1,318	0.2271−15	1.850
	3	240	1,195	0.1542−15	1.573
	4	275	1,300	0.1610−15	1.767
	5	215	1,145	0.1674−15	1.546
	6	265	1,278	0.3597−15	1.775
	7	310	1,468	0.3231−15	2.021
	8	305	1,666	0.2367−15	2.259
	9	210	1,663	0.2446−15	2.188
	10	275	1,854	0.3133−15	2.321
ALPS	1	495	1,999	0.6083−11	0.877
	2	300	1,663	0.3056−15	0.635
	3	425	1,938	0.1251−13	0.807
	4	435	1,922	0.5306−12	0.795
	5	375	1,797	0.1072−11	0.720
	6	425	1,926	0.3197−14	0.782
	7	515	1,992	0.1038−9	0.872
	8	490	1,994	0.2672−13	0.888
	9	520	1,999	0.1318−13	0.844
	10	405	1,955	0.4838−13	0.747

Program	Run No.	ITN	IFN	F-final	T
ALAG	1	860	2,000	0.5205−3	2.294
	2	845	2,000	0.1098−4	2.215
	3	825	2,000	0.1435−6	2.183
	4	845	1,996	0.2901−9	2.074
	5	905	2,000	0.1115	2.065
	6	880	2,000	0.6123	2.086
	7	820	2,000	0.1469−2	2.091
	8	910	1,998	4.361	2.109
	9	885	2,000	0.20−2	2.098
	10	825	2,000	0.1785−11	2.181
ALPART	1	560	833	0.2046−12	2.599
	2	835	1,224	0.8878−12	3.696
	3	725	1,052	0.9679−12	3.184
	4	645	969	0.2953−11	2.889
	5	725	1,124	0.2454−12	3.316
	6	965	1,386	0.1933−12	4.214
	7	1,085	1,531	0.9901−13	4.694
	8	1,165	1,668	0.1027−12	5.098
	9	730	1,148	0.6257−11	3.308
	10	1,105	1,560	0.7928−12	4.855
ALCODR	1	905	2,000	0.2797−15	2.065
	2	510	1,500	0.8248−22	1.453
	3	310	1,147	0.2049−23	1.097
	4	635	1,745	0.1476−20	1.792
	5	865	2,000	0.2396−14	1.968
	6	815	2,000	0.5347−20	1.975
	7	805	2,000	0.2577−18	2.033
	8	910	2,000	0.3690−16	1.986
	9	930	2,000	0.3497−1	1.959
	10	660	1,882	0.2154−21	1.822
ALVAM	1	640	1,717	0.1683−14	6.602
	2	690	1,998	0.1320−11	6.970
	3	735	1,996	0.1214−14	6.658
	4	595	1,663	0.1933−14	5.664
	5	720	1,998	0.2302−6	6.337
	6	705	1,997	0.1099−11	6.413
	7	715	1,996	0.5063−15	7.043
	8	680	1,998	0.1414−12	6.876
	9	680	1,930	0.1540−14	6.805
	10	580	1,653	0.5698−14	5.921

× The computation had no convergence

TABLE 20 Computational results for Gauss's function

Program	Run No.	ITN	IFN	F-final	T
VA 01 A	1	11	22	0.1128−7	0.176
	2	12	24	〃	0.170
	3	12	24	〃	0.176
	4	12	24	〃	0.174
	5	14	28	〃	0.195
	6	13	26	〃	0.184
	7	13	26	〃	0.184
	8	12	24	〃	0.168
	9	13	26	〃	0.182
	10	11	22	〃	0.153

Program	Run No.	ITN	IFN	F-final	T
VA 06 A	1	28	28	0.1128−7	0.384
	2	28	28	〃	0.370
	3	17	17	〃	0.224
	4	17	17	〃	0.225
	5	25	25	〃	0.338
	6	19	19	〃	0.255
	7	16	16	〃	0.219
	8	22	22	〃	0.290
	9	19	19	〃	0.253
	10	16	16	〃	0.216
VA 08 A	1*				
	2	16	35	0.11279−7	0.26
	3	16	34	〃	0.243
	4	16	37	〃	0.253
	5*				
	6	19	42	0.11279−7	0.303
	7	16	39	〃	0.253
	8	25	50	〃	0.347
	9	16	38	〃	0.24
	10*				
VA 09 A	1	13	14	0.11279−7	0.167
	2	12	15	〃	0.143
	3	13	15	〃	0.156
	4	14	15	〃	0.159
	5	19	21	〃	0.224
	6	20	25	〃	0.254
	7	19	21	〃	0.23
	8	19	22	〃	0.227
	9	16	20	〃	0.199
	10	16	17	〃	0.183
VA 10 A	1	16	95	0.11279−7	0.393
	2	15	93	〃	0.363
	3	15	89	〃	0.352
	4	13	76	〃	0.306
	5	19	113	〃	0.461
	6	21	125	〃	0.508
	7	20	125	〃	0.49
	8	18	104	〃	0.422
	9	16	100	〃	0.396
	10	15	93	〃	0.368
MINIM	1	9	13	0.11279−7	0.153
	2	6	8	〃	0.096
	3	5	8	〃	0.089
	4	8	17	〃	0.149
	5	7	11	〃	0.124
	6	7	17	〃	0.137
	7	7	13	〃	0.129
	8	8	12	〃	0.134
	9	8	12	〃	0.134
	10	7	10	〃	0.11

Program	Run No.	ITN	IFN	F-final	T
CGD	1	12	26	0.11279−7	0.192
	2	14	30	〃	0.208
	3	14	30	〃	0.206
	4	17	36	〃	0.25
	5	16	36	〃	0.248
	6	16	34	〃	0.241
	7	17	36	〃	0.243
	8	16	36	〃	0.245
	9	16	35	〃	0.234
	10	14	30	〃	0.198
FPD	1	10	28	0.11279−3	0.189
	2	11	35	〃	0.224
	3	11	34	〃	0.200
	4	12	37	〃	0.216
	5	13	39	〃	0.238
	6	12	33	〃	0.206
	7	12	43	〃	0.25
	8	13	41	〃	0.24
	9	13	35	〃	0.212
	10	11	41	〃	0.236
ALSIM	1	65	1,998	0.11279−7	6.632
	2	75	1,999	〃	6.536
	3	75	894	〃	2.961
	4	75	1,998	〃	6.334
	5	70	901	〃	2.877
	6	85	1,998	〃	6.330
	7	80	1,998	〃	6.448
	8	75	1,998	〃	6.190
	9	75	1,999	〃	6.514
	10	65	1,998	〃	6.426
ALPS	1	60	760	0.11279−7	2.125
	2	45	694	〃	1.861
	3	50	715	〃	1.891
	4	50	717	〃	1.861
	5	45	709	〃	1.945
	6	60	786	〃	2.041
	7	45	723	〃	1.887
	8	50	689	〃	1.835
	9	45	687	〃	1.825
	10	45	682	〃	1.840
ALAG	1	145	504	0.11279−7	1.920
	2	185	541	〃	1.992
	3	190	609	〃	2.205
	4	185	541	〃	1.952
	5	230	716	〃	2.593
	6	230	712	〃	2.599
	7	135	413	〃	1.481
	8	265	870	〃	3.625
	9	185	573	〃	2.142
	10	140	461	〃	1.676

Program	Run No.	ITN	IFN	F-final	T
ALPART	1	120	247	0.11279−7	1.464
	2	155	296	〃	1.780
	3	130	255	〃	1.485
	4	120	225	〃	1.325
	5	195	358	〃	2.138
	6	155	291	〃	1.719
	7	155	294	〃	1.703
	8	135	288	〃	1.635
	9	140	273	〃	1.568
	10	130	250	〃	1.395
ALCODR	1	45	310	0.11279−7	1.144
	2	50	312	〃	1.155
	3	50	319	〃	1.162
	4	40	309	〃	1.127
	5	60	363	〃	1.325
	6	55	332	〃	1.212
	7	70	376	〃	1.345
	8	85	420	〃	1.472
	9	65	355	〃	1.262
	10	75	371	〃	1.310
ALVAM	1	40	77	0.11279−7	0.814
	2	40	68	〃	0.674
	3	55	94	〃	0.958
	4	40	74	〃	0.761
	5	45	75	〃	0.763
	6	45	88	〃	0.885
	7	40	66	〃	0.651
	8	45	89	〃	0.917
	9	35	59	〃	0.614
	10	40	74	〃	0.758

＊ Exponential argument error

関数に対しては ROTAX, SPASE を除いた他のルーチンの安定性が悪く，収束しなかったほとんどの場合が Fig. 15 と Fig. 17 にみられる $x_2=1$ に沿った深い谷の中にはいりこんでいた．Fig. 18 には，このようすをみるため，探索の過程に特徴的な相異がみられた直接探索法のルーチン SPASE と降下法のルーチン ALAG の過程を図示する．図からわかるように，ALAG では 20 回の反復で谷にはいりこみ，以後どんどん深い谷底へ降下しているが，SPASE では3回目から x_1 軸に平行に正の方向へ進み，以後 40 回の反復でほぼ最適点の近傍へ収束している．

TABLE 15 は Box の関数の結果で，VA 08 A, VA 09 A, ALCODR の出発点への依存性が強く，収束しなかったほとんどのケースが $x_1\to+\infty$, $x_2\to+\infty$, $x_3\to 0$ なる点へ向かう過程での exponential argument error であった．要した計算時間では MINIM, CGD などが速かった．

TABLE 16 は Enzyme の関数に対する結果であるがやはり ROTAX, SPASE を除いた他のルーチンの安定性が悪いようであった．

TABLE 17 は Watson の関数に対する結果で，ALPAC ルーチンと CGD の安定性が悪かった．計算時間では MINIM, VA 09 A が速かった．

TABLE 18 は Powell の関数の結果であるが，この関数に対しては全てのルーチンが全ての出発点に対して最適点へ収束している．計算時間では MINIM, FPD, VA 10 A などが速かった．

TABLE 21 Comparison of convergence efficiency for Rosenbrock and Beale's functions

ROUTINE	Rosenbrock		Beale	
	Stability	C. E.	Stability	C. E.
VA 01 A	10	0.1	7	0.1
VA 06 A	10	0.2	7	0.1
VA 08 A	10	0.4	5	0.2
VA 09 A	10	0.1	1	0.5
VA 10 A	10	0.1	3	0.1
MINIM	10	0.2	6	1.0
CGD	10	0.6	3	0.4
FPD	10	1.0	5	0.5
ALSIM*	10	0.2	7	0.2
ALPS	10	0.1	6	0.7
ALAG	10	0.1	7	0.1
ALPART	9	0.02	6	0.04
ALCODR	10	0.1	1	0.2
ALVAM	10	0.05	7	0.05
ROTAX*	10	0.5	10	0.3
SPASE*	10	0.2	10	0.1

Programs marked with ＊ are based on the Direct Search Method.
The same explanation is applied to Tables 22～25.

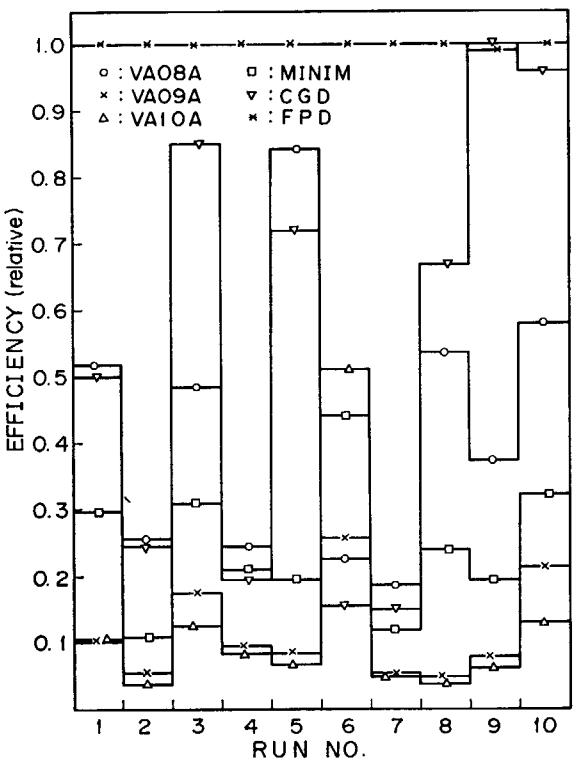


Fig. 19 Comparison of convergence efficiency of VA 08 A, VA 09 A, VA 10 A, MINIM, CGD and FPD for Rosenbrock's function.

TABLE 22 Comparison of convergence efficiency for Box and Enzyme's functions

ROUTINE	Box		Enzyme	
	Stability	C. E.	Stability	C. E.
VA 01 A	10	0.4	7	0.4
VA 06 A	9	0.5	7	0.3
VA 08 A	1	0.2	6	0.2
VA 09 A	5	1.0	6	0.3
VA 10 A	8	0.3	4	0.4
MINIM	8	0.9	3	1.0
CGD	10	0.9	0	—
FPD	9	0.7	3	0.4
ALSIM*	10	0.1	7	0.2
ALPS*	10	0.05	5	0.1
ALAG	10	0.03	3	0.1
ALPART	10	0.03	7	0.03
ALCODR	5	0.1	4	0.1
ALVAM	9	0.1	6	0.01
ROTAX*	10	0.6	10	0.4
SPASE*	10	0.01	10	0.04

TABLE 23 Comparison of convergence efficiency for Watson and Powell's functions

ROUTINE	Watson		Powell	
	Stability	C. E.	Stability	C. E.
VA 01 A	10	0.3	10	0.3
VA 06 A	10	0.2	10	0.3
VA 08 A	6	0.05	10	0.4
VA 09 A	10	0.5	10	0.5
VA 10 A	10	0.3	10	0.7
MINIM	10	1.0	10	1.0
CGD	0	—	10	0.4
FPD	10	0.3	10	0.9
ALSIM*	5	0.1	10	0.4
ALPS*	5	0.2	10	0.3
ALAG	0	—	10	0.1
ALPART	0	—	10	0.1
ALCODR	0	—	10	0.3
ALVAM	0	—	10	0.04
ROTAX*	10	0.4	10	0.8
SPASE*	—	—	10	0.1

TABLE 19, 20 は Wood と Gauss の関数の結果であるが，ともに全てのルーチンが最適点へ収束した．

TABLE 21 から TABLE 24 には前述の計算結果から各ルーチンの収束効率を比較するため，(4-10)式により収束効率の計算した表が示されている．表の中で，"stability" の欄は生成した 10 個の出発点からの計算の中で最適点の近傍へ収束した計算回数，C. E. の欄は各出発点ごとに(4-10)式により収束効率を計算し，各ルーチンごとにその平均をとり，比較が行ない易いように最も良かったもので規格化したものである．表が示すように，Rosenbrock の関数に対しては FPD の収束効率が最も良く，次いで CGD, ROTAX, VA 08 A の順になっており，逆

TABLE 24 Comparison of convergence efficiency for Wood and Gauss's functions

ROUTINE	Wood		Gauss	
	Stability	C. E.	Stability	C. E.
VA 01 A	10	0.2	10	0.7
VA 06 A	10	0.3	10	0.5
VA 08 A	10	0.3	7	0.5
VA 09 A	10	0.5	10	0.7
VA 10 A	10	0.4	10	0.3
MINIM	10	1.0	10	1.0
CGD	10	0.6	10	0.5
FPD	9	0.9	10	0.6
ALSIM*	10	0.2	10	0.04
ALPS*	10	0.4	10	0.1
ALAG	9	0.1	10	0.1
ALPART	10	0.1	10	0.1
ALCODR	10	0.2	10	0.2
ALVAM	10	0.05	10	0.2
ROTAX*	10	0.7	10	0.4
SPASE*	10	0.1	10	0.1

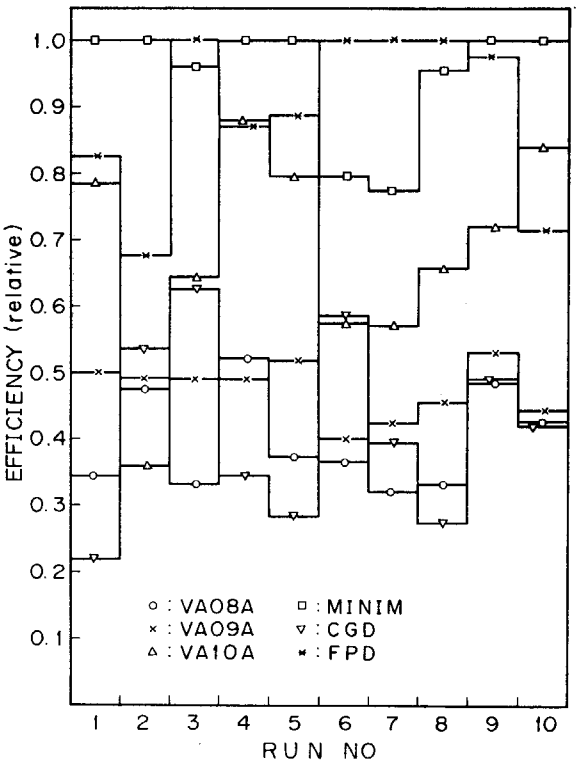


Fig. 20 Comparison of convergence efficiency of VA 08 A, VA 09 A, VA 10 A, MINMI, CGD and FPD for Powell's function

に ALPART, ALVAM の効率が悪い．Fig. 19 は，この関数に対する収束効率を傾斜法による6つのルーチン，VA 08 A, VA 09 A, VA 10 A, MINIM, CGD, FPD の間で出発点ごとに比較した図である．全般的に FPD が良く，特に Run No. 2, 4, 7 に対しては2番目の効率のルーチンの約4倍の収束効率であったことがわかる．

Beale の関数に対する収束効率では MINIM で最も良く，次いで ALPS, FPD, VA 09 A の順で，逆に AL-

TABLE 25 Convergence efficiency for extended Rosenbrock function

Routine \ m	4	8	10	20
VA 01 A	2.15	0.80	0.60	0.17
VA 06 A	0.54	0.17	0.08	0.02
VA 08 A	1.99	0.80	0.68	0.81
VA 09 A	1.23	0.51	0.34	0.09
VA 10 A	1.27	0.45	0.29	0.05
MINIM	3.70	1.78	1.24	0.33
CGD	1.93	1.18	1.07	0.62
FPD	2.44	1.25	0.62	0.22
ALSIM*	0.36	0.05	0.004	0.001
ALPS*	2.79	1.22	0.20	0.07
ALAG	0.25	0.20	0.19	0.13
ALPART	0.29	0.26	0.25	0.19
ALCODR	0.42	0.06	0.03	0.01
ALVAM	0.12	0.06	0.05	0.01
ROTAX*	2.53	0.73	0.11	0.004
SPASE*	0.48	——	——	——

PART, ALVAM が悪かった.

TABLE 22 は, Box と Enzyme の関数に対する収束効率の表である. Box では, VA 09 A, MINIM, CGD, FPD などの降下法のルーチンの効率が良く, 次いで直接探索法の ROTAX であった. 逆に ALPART, ALVAM の効率が悪かった. Enzyme の関数では MINIM が安定性が悪かった反面, 収束効率が良く, 逆に全ての出発点に対して成功した SPASE の収束効率が悪かった.

TABLE 23 は Watson, Powell の関数に対する安定性と収束効率の表で, Watson の関数では MINIM, VA 09 A, ROTAX の順に効率が良く, 逆に VA 08 A が悪かった. なお, SPASE は 4 変数までの関数を対象としているのでこの関数に対しては計算を行なわなかった. Powell の関数に対しては全てのルーチンの安定性が良く, 収束効率では MINIM, FPD, ROTA, VA 10 A の順に良く, 逆に ALPAC ルーチンと SPASE が悪かった. なお, この関数に限らず概して ALPAC ルーチンの収束効率の悪さが目立つが, これは, ALPAC ルーチンで採用している傾斜ベクトル計算を階差近似により行なっていることや直線探索の方法に起因するものと思われる. Fig. 20 は Powell の関数に対する VA 08 A, VA 09 A, VA 10 A, MINIM, CGD, FPD の出発点ごとの収束効率の図で, Run No. 1, 2, 4, 5, 9, 10 に対しては MINIM が最も良く, 残りの出発点に対しては FPD が良かった. しかしながら他のルーチンでも大きな差はなく, テスト関数としては比較的おとなしい関数であった.

TABLE 24 は, Wood と Gauss の関数に対する収束効率で, 全般的に MINIM, FPD, VA 09 A, CGD などの降下法のルーチンの効率が良かった.

TABLE 25 は, 拡張 Rosenbrock の関数に対する各ルーチンの収束効率で, 変数の数を 4, 8, 10, 20 ととり, 原点を出発点として計算した各ルーチンの収束効率を示したものである. 表に見られるとおり, 4 変数では MINIM, ALPS, ROTAX, FPD の順に良く, 8, 10 変数になると MINIM, CGD, FPD, VA 08 A などが良い. また 20 数では VA 08 A, CGD, MINIM, FPD, ALPART の順に変良く, 逆に ALSIM, ROTAX などの効率が 4 変数の場合に比べて急激に悪くなっている.

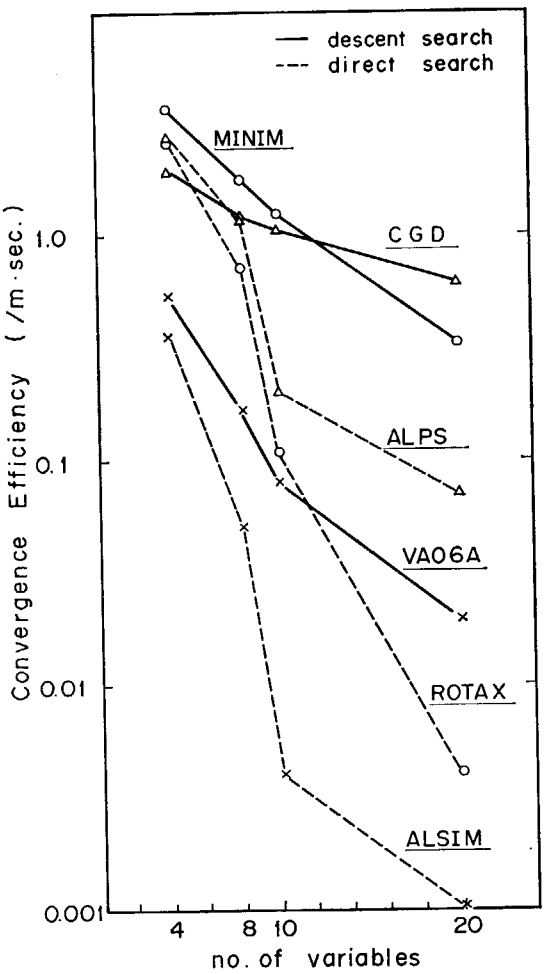


Fig. 21 Comparison of convergence efficiency between direct search and descent search for extended Rosenbrock's function

Fig. 21 は, 収束効率の変数の数に対する依存性を直接探索法のルーチンと傾斜法のルーチンとで比較するため, 直接探索による 3 つのルーチン, ROTAX, ALSIM, ALPS と, 4 変数のときの効率が, それらと同程度であった傾斜法によるルーチン, MINIM, CGD, VA 06 A の収束効率をグラフ化したもので, 縦軸に収束効率, 横軸に変数の数がとられている.

図で見るとおり, 4 変数のときには MINIM, CGD とほぼ同程度の収束効率であった ALPS, ROTAX は, 変数の増加と共に急速に収束効率が悪くなり, 20 変数のときの収束効率を 4 変数のときと比べると ALPS では 1/40, ROTAX では 1/600 になっている. 一方 MINIM, CGD ではそれぞれ 1/10, 1/3 程度にしかなっていない.

同様に ALSIM と VA 06 A ではそれぞれ 1/360, 1/25 である．このように多変数の問題に対しては直接探索法の収束効率が悪くなるが，この理由は，直接探索法では変数の数の次元空間での座標軸方向あるいは幾何形上で探索が行なわれるため，関数評価回数が変数の増加に伴ない急速に増えることによる．これに対し傾斜法がそれほど悪くならないのは，関数評価回数が変数の数に依存せず，その中で行なわれる直線探索だけに依存することにある．

5. 最適化コードシステム SCOOP

冒頭で述べたとおり，我々は制約条件付きあるいは制約条件無しの最適化問題に対する種々の最適化手法のベンチマーク・テストを実施し，各手法の持つ特性，安定性，収束効率を調べることによって，それらの手法を統合化した最適化コードシステムの開発を目指してきたが，現在までに，第3章，第4章で紹介した最適化プログラムより構成される最適化コードシステム SCOOP のバージョンⅠが完成したので，この章では SCOOP の基本的な構想とその流れについて説明する．

Fig. 22 は最適化問題を解く場合の一般的な流れである．(Ⅰ)は解こうとする問題に対するいろいろな情報を(Ⅱ)に伝えるものである．この場合必要とされる情報には次のようなものがある．

(ⅰ) 変数の数（次元）
(ⅱ) 変数の配列
(ⅲ) ワークエリア
(ⅳ) 制約条件式の数
(ⅴ) ユーザ・サプライの関数型
(ⅵ) 変数の定義域
(ⅶ) 共通パラメータ；収束判定パラメータ
　　関数計算回数上限
　　反復計算回数上限
　　プリント・オプション
(ⅷ) 使用するアルゴリズムに特有のパラメータ；
　　simplex 法における α, β, γ
　　complex 法における α
　　etc.

これらの入力量は，あらかじめユーザが使用しようとするアルゴリズムの how to use から設定する量である．(Ⅱ)は，(Ⅰ)によって与えられた情報をもとに最適化を行なうボックスで，一般にはブラックボックスであり，その中でどのように最適化が行なわれているかはユーザが関知しないところである．換言すれば，ユーザは最適化の手法に関する知識を必要とせずに問題を解くことができる．そして(Ⅱ)を経て，求める最適解 $\boldsymbol{x}^*$，目的関数値 $F(\boldsymbol{x}^*)$，その他ユーザの要求に応じた出力量が得られる．

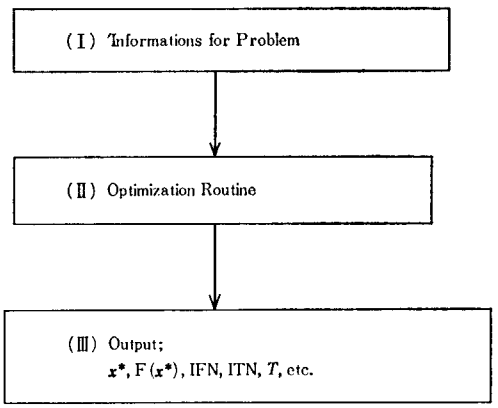


Fig. 22 General calculational flow for a optimization problem

Fig. 22 の流れによる計算は，これまでボックス(Ⅱ)を個々の最適化プログラムを用いて行なわれていたが，このような個々のプログラムによって最適化問題を解く際の問題点としては，得られた解が問題の大域的な最適解ではなく局所的な定常点を与えている危険性があること，および問題の形が変わるたびに別の最適化のルーチンを探さなくてはならないことなどがあり，前者を解決するためには，例えば，十分多くの出発点を生成して計算を行ない，得られた解の中から最も小さい関数値を与えるものを最適解として採用する方法もあるが，この方法が大変効率の悪いことは言うまでもない．また，後者では与えられた最適化問題に最も適した手法を探すことが望ましいが，このことは最適化の手法に対する相当の知識をユーザに要求することになり，問題を解くことだけを目的とする者にとり煩らわしいことであろう．あらゆる問題に適した手法が存在するなら，このようなことは解決されるであろうが，他の分野の問題と同様に最適化問題においても，そのようなオールマイティな手法は存在しない．したがって，どのような問題に対しても，それに適した手法が用意されている最適化コードシステムが望まれるのである．

SCOOP はこのような問題を解決するために開発された最適化コードシステムで，**Fig. 22** のボックス(Ⅱ)を個々のプログラムとしてではなく，システムとして担当することにより，あらゆる最適化問題を処理しようというものである．

5.1 SCOOP の流れ図

前述のような目的で開発された SCOOP は合計 32 個の最適化プログラムにより構成されているが，**Fig. 23** にはその基本的な流れ図が描かれている．

SCOOP では，まず制約条件式の有無が判定され，それに応じて分岐する．制約条件を持たない問題は下に降りるが，この場合探索は大域的探索 (global search) と局所的探索 (local search) に分けて行なわれる．この理由は，直接探索法は一般に前章のベンチマーク・テストの

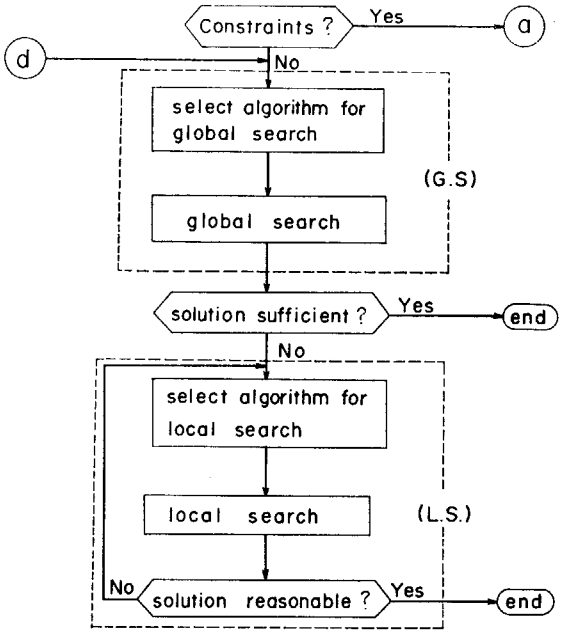


Fig. 23 Configurations of SCOOP

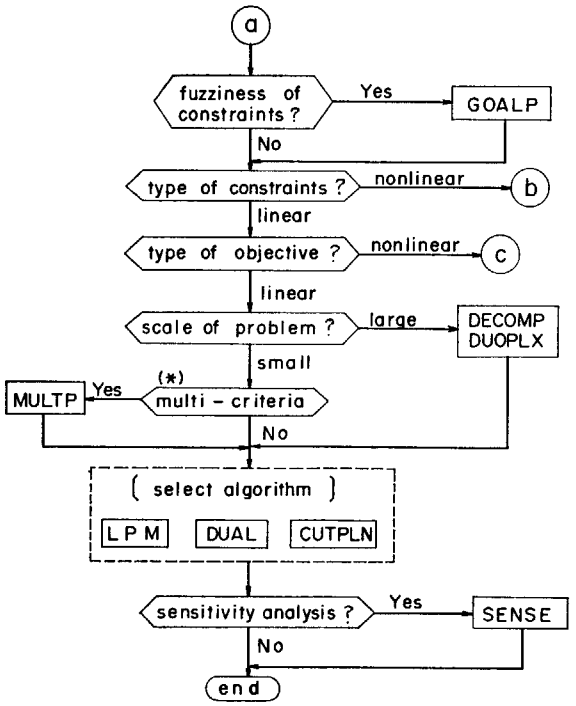


Fig. 23 (continued)

結果から出発点の位置に左右されずに大域的な最適点の探索に適しているが，その反面収束効率が悪いということ，また降下法は収束効率は良いが出発点の近傍の局所的な定常点へ収束する危険性があることなどである．

第一段階の大域的探索は，与えられた出発点から問題の大域的な粗い最適点の探索を目指し，このためのルーチンとしては直接探索による ALPS, ROTAX, ALSIM, SPASE, CORASE, COMPLX が当てられる．次の第二段階の局所的な探索では，第一段階の探索により到達した点を出発点として，その点の近傍の局所的な最適点の探索を行ない，収束判定基準などはさらに厳しくされる．また手法としては収束効率の良い降下法によるルーチン VA 01 A, VA 06 A, VA 08 A, VA 09 A, VA 10 A MINIM, CGD, FPD, ALAG, ALPART, ALCODR, ALVAM が当てられている．また，この局所的探索により得られた解に対しては，それが問題の解として合理的であるか否かが判定される．判定の結果，問題の解として不合理であったり，また判定法が難かしい場合には再度異なった手法により局所的探索が行なわれる．

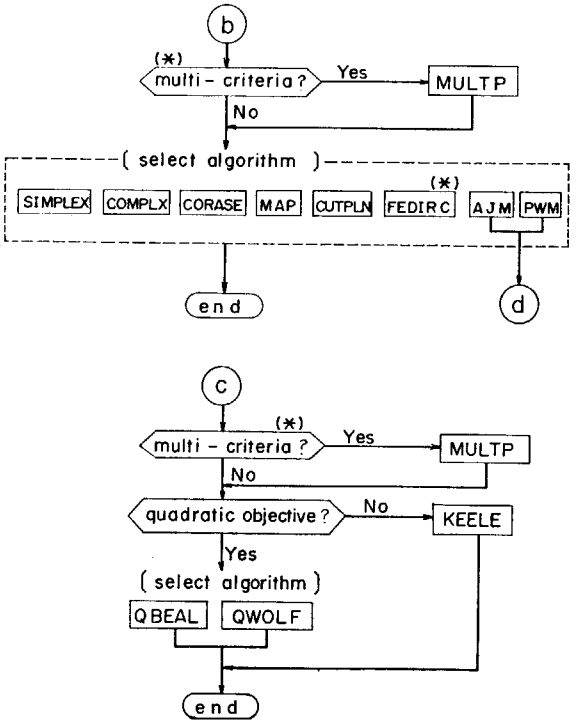


Fig. 23 (continued)

制約条件を持つ問題は，a に分岐し，まず制約条件式のあいまい性や順位付きが判定される．制約式にあいまい性があったり優先順位を持つ問題は，GOALP によって (3-5) 式のように問題の定式化が行なわれる．この場合優先順位や，あいまい性を持つ制約式の番号はあらかじめ入力されているものとしている．あいまい性を持たない問題は，下に降りて制約条件式の形が判定され，非線形制約，線形制約に応じて分岐する．非線形制約式を持つ場合は b に分岐し線形制約式の場合は下に降り，目的関数の線形，非線形性が判定される．目的関数と制約条件式が共に線形の問題は線形計画法の問題とされ，まず問題の規模が判定され，大規模問題に対しては Dantzig と Wolfe の分解原理を用いたプログラム DECOMP が用意されている．また大規模ではあるが，変数の数はそれほど多くない問題に対しては DUOPLX が用いられる．いずれのプログラムも，その中で単体法のルーチンが必要なので LPM を選択する点へ行く．問題の規模が大きくない場合は，まず最適性の基準の多重性が問われる．ここで(*)印が付いているのは多目的計画法によるプログラム MULTP が現在開発中のため，現在のところこの判定ルーチンは採用されていないことを意味して

いる．なお，破線ボックス中のアルゴリズムの選択の中で，CUTPLN は整数計画法と混合整数計画法の問題を解くためのものである．線形計画法の問題として解かれた解に対しては感度解析が必要か否かが判定され，感度解析が必要の場合には SENSE を用いて解析が行なわれる．この場合，コスト係数に対する解析か，右辺要素に対する解析か，また感度解析を必要とする制約条件式番号，コスト係数番号などの入力量が要求される．

制約条件式が非線形である問題はｂに分岐するが，まず問題が多目的計画法のルーチンを必要とするか否かが判定され，次いでアルゴリズムの選択が行なわれる．この選択は，シンプレックス法による SIMPLEX，コンプレックス法による COMPLX，制約ランダム法による CORASE，近似計画法による MAP，切除平面法による CUTPLN，実行可能方向法による FEDIRC，それに変換法による AJM と PWM の中から行なわれる．この中で FEDIRC は現在開発中のプログラムである．また，変換法による AJM と PWM が選択された場合には，制約条件式と目的関数から変換関数を作成したのち制約条件無しの手法で解かれるのでｄに分岐する．

非線形目的関数，線形制約式を持つ場合はｃに分岐し，やはり目的関数はベクトルかスカラーかが判定される．次いで，目的関数が二次形をしている場合には二次計画法の問題となり，Beale の方法による QBEAL と Wolfe の方法による QWOLF が用意されている．目的関数が非線形の場合には KEELE により解かれる．

なお，SCOOP のバージョンⅠでは，アルゴリズムの選択は制約条件式や変数の数を考慮して標準的なルーチンが選ばれることになっているが，最終的には目的関数や制約式の形をも考慮して最も適したルーチンが選択されることになる．同時に SCOOP の対話形式化を計り，計算機と会話をしながら問題が解かれることになる．また SCOOP の中に内蔵されている各ルーチンは独立した形をとっており，したがってユーザの希望するサブルーチン名をコールすることによりサブルーチン形式で使用することも可能である．

5.2 ユーザズ・ルーチン

SCOOP の中の各ルーチンではその機能に応じたユーザ・サプライのルーチンが必要である．TABLE 26 に各ルーチンが必要とするユーザ・サプライの関数型が示されている．この中で ALPAC ルーチンは目的関数型しか要求されていないが，これは目的関数の微係数の計算を階差近似により行なっているからである．また AJM では，目的関数や制約式の微係数の必要性はそれが組みこまれる手法に依存する．なお，線形ルーチンでは目的関数型や制約条件式は係数ベクトルあるいは係数行列として入力量の形で要求される．

TABLE 26 User supply routine for SCOOP

Program	objective	derivative for objective	Hessian	constraints	derivative for constraints
VA 01 A	○	○			
VA 06 A	○	○			
VA 08 A	○	○			
VA 09 A	○	○			
VA 10 A	○				
MINIM	○	○	○		
FPD	○	○			
CGD	○	○			
ALSIM	○				
ALPS	○				
ALAG	○				
ALPART	○				
ALCODR	○				
ALVAM	○				
ROTAX	○				
SPASE	○				
SIMPLEX	○			○	
COMPLX	○			○	
CORASE	○			○	
MAP	○	○		○	○
AJM	○	—	—	○	
PWM	○	—	—	○	
KEELE	○	○		○	
CUTPLN	○	○		○	○
QBEAL	○	○		○	
QWOLF	○	○		○	

This is a blank page.

6. 結　　論

第3章，第4章において見てきたとおり，著者は最適化コードシステムの開発を目標に LPM, DUAL, DECOMP, GOALP, SENSE, CUTPLN の6つの線形問題に対する最適化プログラム，CORASE, MAP, AJM, PWM, QBEAL, QWOLF の6つの制約条件を持つ非線形問題に対する最適化プログラム，SPASE, ROTAX の2つの制約条件無しの問題に対するプログラムを開発し，さらに合計 18 個のプログラムを整備した．開発された各プログラムともベンチマーク・テストを通してその有効性が実証されており，また著者によって提案された球面探索法によるプログラム SPASE も，4.4節で紹介したとおり，手法として安定していることがわかった．また整備された各プログラムとも，その安定性と収束効率が評価され SCOOP の開発に際して多くの情報を得ることができたが，ベンチマークテストの結論として以下のことが言える．

非線形の制約条件付きの問題に対するプログラムでは，

(1) 全体として MAP の効率が良いが，LP ルーチンを用意する必要がある．またユーザーズ・ルーチンとして，目的関数および制約条件式の導関数計算ルーチンが必要である．この手法に固有の量として変数の変域を制限する刻み幅 δ_i を与える必要があるが，どのような基準で選択すべきかということと，各反復の過程でどのように小さくして行くべきかの基準がはっきりしていない．

(2) 発見的方法によるものの中では COMPLX の効率が良い．しかしながら，この手法では等号制約を扱えないということと，複体の端点の数と端点を重心に関して折り返すときのファクター α の決定法が困難である．Box は端点の数として $2\times n$, $\alpha=1.3$ を適当としている．

(3) SIMPLEX が採用した simplex 法は手法としては安定し，等号制約条件を扱うことができるという利点があるが，やはり手法に固有の係数の定め方がむずかしい．

(4) CORASE は使い易さという点で優れているが，効率や精度の点で劣る．

(5) AJM は制約条件無しの任意の手法と結合させて使用できるという利点があるが，当然のことながら収束効率や安定性もその手法の影響を受ける．

また制約条件無しの問題に対するルーチンの中の直接探索法では，次のような利点がある．

(1) 一般にアルゴリズムが簡単で計算機の記憶量が少なくてすむ．

(2) 目的関数に対する微分可能性などの要求はなく，その値だけが評価できれば良い．

(3) 安定性の点で優れており，global search 用の手法に適している．

しかしながら，その反面，次のような不利な点がある．

(4) 収束効率の点で劣り，特に最適点の近くでの収束が遅い．

(5) 多変数問題に対する効率が悪い．

(6) 各手法に固有の係数，例えばシンプレックス法における reflection, expansion, contraction factor，座標回転法，パターン探索法におけるステップ幅調節係数，球面探索法における半径縮小係数などの定め方の基準が与えられていない．

傾斜法では，次の特徴がみられる．

(1) 全般的に収束効率の点で優れており，特に MINIM が良い．

(2) 一般に可変計量法に基づくルーチンの安定性，収束効率が良い．

(3) 直線探索の計算ルーチンが必要で，収束効率や解の精度もその影響を受ける．

(4) ユーザーズ・ルーチンとして目的関数の1階あるいは高階の導関数の計算ルーチンを用意する必要がある．

(5) 収束点が出発点の位置に左右されやすい．

一般にシステムの基本的性能は効率性 (efficiency)，信頼性 (reliability)，有用性 (usefulness)，汎用性 (flexibility) によって評価されると言われており，最適化コードシステム SCOOP は前述のベンチマーク・テストの結果を踏まえ，効率性，信頼性，有用性，汎用性を考慮し，以下のような目的で設計されている．

1. 探索を大域的探索と局所的探索あるいは異なった手法により行なうことによって，得られる解に対する信頼性を高める．(reliability)
2. 与えられた問題に対し，それに最も適した手法を用いることにより，計算効率を高める．(efficiency)
3. 内蔵されている最適化プログラムをサブルーチン形式とすることにより，各プログラムの改良と新しい手法の追加が可能であり，常に最新の手法によってユーザの要求に応えることができる．(usefulness)
4. あらゆる形の最適化問題に対する手法を用意することにより，ユーザの持つどのような問題にも対拠でき，したがってユーザは，問題の形が変わる毎に別の最適化プログラムを探さなくてはならないとい

う煩雑さから解放される．(flexibility)

SCOOP は今後とも新しく開発された手法を整備し，付け加えていくことにより，さらに充実したシステムとなるであろう．以後の課題としては SCOOP のアルゴリズムの自動選択化と対話形式化を計る一方，ユーザのあらゆる要求に対処できるように改善がなされる必要があるだろう．同時に核エネルギーシステムや原子炉システム解析への応用など，現実の問題に対する SCOOP の積極的な応用が計られることを望む．

最適化問題は工学，経済，自然科学等のおよそあらゆる分野に現われる問題であり，したがって設定される最適性の基準は種々様々である．またその規模も，例えば，エネルギーシステム解析の場合では変数の個数が千個，制約条件の数が数百個の規模になる場合もあり，また逆に非線形問題では数個の変数の場合もある．しかし変数が数個だからといって数百個の問題よりも容易に解けるということにならず，目的関数や制約領域の形状が問題の難易度を左右することになる．その典型が線形問題であろう．第3章でみた通り，線形問題では制約領域が凸多面体になり，目的関数が変数次元空間での超平面になることから，求める最適点の探索は制約領域内の全ての点を考慮する必要がなく，有限個しかない端点だけについて調べて行くと良いことになる．そして実行可能領域が有界なら，必らず最適点が存在することが知られており，非線形問題と比べると扱い易くなる．したがって，手法も単体法が中心になっており，非線形の場合には目的関数の形と制約領域の形に応じて多くの手法が開発されているのと対照的である．もちろん，問題の規模が大きくなるにつれ，計算効率や精度を良くするために種々の改良がなされてきた．たとえば，LPM が採用している改訂単体法や変数に対する上限，下限制約を効率良く処理する有界変数法，そして分解原理法などはその例である．しかしながら，単体法の基本的な考え方は Dantzig 以降変っていない．また最近の電子計算機の大型化，高速化にも助けられ，単体法が最適化問題の解法の中に占める比重は大きくなっている．

一方，非線形問題に対しては第4章でみたとおり，数多くの手法が開発されているが，いずれも一長一短があり，あらゆる問題に通用する万能の方法はない．最適化問題のためのコードシステムを開発し，さらにその対話形式化を計る理由はここにあるが，大切なことは，問題に携わる当人が，その問題の目的関数の形，制約領域の形を十分把握しておくということであろう．それにより，問題を解くに際して，より多くのより良い情報の下に出発できるからである．特に非線形問題では，目的関数の形が複雑になるほど出発点の与え方が計算効率，精度に大きな影響を与えることが知られており，SCOOP の応用に際しては上述のような認識が大切である．

謝　辞

本コードシステム SCOOP の開発にあたり，研究の場を与えてくださり，また常に貴重な御助言をいただいた原子炉工学部原子炉制御研究室長篠原慶邦主任研究員に深く感謝いたします．また各種の最適化手法の整備とベンチマーク・テストに際して御助力をいただいた日本情報サービス株式会社科学計算部木島敏男氏に深謝いたします．また，ベンチマーク・テストの結果の解析にあたり，御検討をいただいた原子炉工学部・システム解析手法研究ワーキング・グループのメンバーの方々および本報告のまとめに当たって御助言，御協力いただいた原子炉工学部原子炉システム研究室中原康明主任研究員と藤村統一郎研究員に感謝の意を表します．

参 考 文 献

1) 村田 浩, 林 敏和, 桂木 学: "わが国の核エネルギー開発計画の現状と長期戦略", JAERI-M 6632 (1976)
2) 原子力システム研究委員会: "原子力システムの分析と評価", NIRA, NRO-50-2-1 (1977)
3) 鈴木篤之, 清瀬量平: "長期エネルギー戦略のシステム分析, "J. At. Energy Soc. Jp., **18**, 479 (1976) [in Japanese]
4) 原子炉システム最適化技術研究専門委員会: "原子炉システムの最適化技術", 日本原子力学会報告書 (1971)
5) 清瀬量平: "動力炉の最適化技術, (Ⅲ)燃料交換の最適化", J. At. Energy Soc. Jp., **12**, 541 (1970)
6) 北村正晴, 篠原慶邦: "サマリウム毒作用を考慮した最適炉停止プログラム", JAERI-M 6632 (1976)
7) 原子力システム研究委員会: "BESOM プログラムについて", NIRA, NRO-50-2 (1977)
8) 鈴木忠和: "少変数の非線形最適化問題に対する一手法", JAERI-M 7229 (1977)
9) 猪瀬 博, 茅 陽一, 他: "システム工学", 岩波講座, 基礎工学 21, 岩波書店 (1969)
10) 平本 巌, 長谷 彰: "線形計画法", 培風館 (1973)
11) Hadley, G.: "Linear Programming", Addison Wesley Publishing Company (1962)
12) 福川忠昭: "目標計画法(1), (2), (3)", オペレーションズ・リサーチ, Vol. 20, No. 2, 3, 4 (1975)
13) 関根泰次: "数理計画法Ⅱ", 岩波講座, 基礎工学 5, 岩波書店 (1968)
14) Hadley, G.: "Nonlinear and Dynamic Programming", Addison Wesley Publishing Company (1964)
15) Himmelblau, D. M.: "Applied Nonlinear Programming", McGraw-Hill Book Company (1972)
16) Künzi, H. P., Tzschach, H. G., Zehnder, C. A.: "Numerical Method of Mathematical Optimization", Academic Press (1971)
17) Box, M. J.: "A New Method of Constrained Optimization and a Comparison with Other Methods", Computer J., **8**, 42 (1965)
18) Price, W. L.: "A Controlled Random Search Procedure for Global Optimization", Computer J., **20**, 367 (1977)
19) Griffith, R. E., Stewart, R. A.: "Nonlinear Programming Technique for the Optimization of Continuous Processing Systems", Management Science, **7**, 379 (1961)
20) Allran, R. R., Johnsen, S. E. J.: "An Algorithm for Solving Nonlinear Programming Problems Subject to Nonlinear Inequality Constraints", Computer J., **13**, 171 (1970)
21) 鈴木忠和: "座標回転法による非線形最適化プログラム ROTAX", JAERI-M 7255 (1977)
22) Beale, E. M. L.: "On Minimizing a Convex Function Subject to Linear Inequalities", J. Roy. Stat. Soc., **17 B** (1955)
23) Wolfe, Ph.: "The Simplex Method for Quadratic Programming", Econometrica **27** (1959)
24) Westly, G. W.: "A Linear Constrained Nonlinear Programming Algorithm", ORNL-4644 (1971)
25) Hooke, R., Jeeves, A.: "Direct Search Solution of Numerical and Statiscal Problems", J. ACM, **8**, 212 (1961)
26) Rosenbrock, H. H.: "An Automatic Method for Finding the Greatest or Least Value of a Function", Computer J., **3**, 175 (1960)
27) Nelder, J. A., Mead, R.: "A Simplex Method for Function Minimization", Computer J., **11**, 302 (1968)
28) Fiacco, A. V., McCormick, G. P.: "Computational Aspects of Unconstrained Minimization Algorithms", John Wiley & Sons, Inc. (1968)
29) Davidon, W. C.: "Variable Metric Method for Minimization", ANL-5990 (1959)
30) Fletcher, R.: "A New Approach to Variable Metric Algorithms", Computer J., **13**, 317 (1970)
31) Powell, M. J. D.: "On the Convergence of the Variable Metric Algorithms", Rep. No. T. P. 382, A. E. R. E. (1969)
32) Powell, M. J. D.: "An Efficient Method for Finding the Minimum of a Function of Several Variables without Calculating Derivatives", Computer J., **7**, 155 (1964)
33) Fletcher, R., Reeves, C. M.: "Function Minimization by Conjugate Gradients", Computer J., **7**, 149 (1964)
34) 山崎和彦: "EASY 3 D, TSSによる二変数関数の立体図形表示", 私信
35) 萬金修一, 増田雅彦: "非線形最適化コード ALPACの整備と原子炉プラント制御設計への適用", 私信
36) Powell, M. J. D.: "A Fortran Subroutine for Unconstrained Minimization, Requiring First Derivatives of the Objective Function", AERE-R **6469** (1970)
37) Fletcher R.: "Fortran Subroutines for Minimization by Quasi-Newton Methods", AERE-R 7125 (1972)