逐次二次計画法に基づく非線形最適化プログラムの開発†

小林康弘†† 田村正義†††
野中久典†† 重松洋一†††*

逐次二次計画法を基本アルゴリズムに採用し，汎用的なソフトウェアの枠組みの中で，下記の特長により，個々の問題に対して高い求解性能を実現できる非線形最適化プログラムを開発した．(1)二次計画解法として，効率に優れる Goldfarb-Idnani 法と，精度に優れる最小二乗法とを組み合わせて利用できる．本研究では，二次計画問題，非線形最適化問題を用いて，これら両者の数値的性能を評価し，性能上の特性を明確化した．このように対照的な二次計画解法を用いることにより，解法の選択の範囲を広げ，求解性能を向上させることができる．(2)逐次二次計画法の処理の流れの中の5箇所で行列の正定値性をモニタし，正定値性が悪化した場合に行列を再初期化し，処理をリスタートする．このリスタートの判定に用いる基準は，求解モジュールと分離した制御モジュールで，標準的な仕様から個々の問題に適合するような仕様に容易に調整できる．このようにアルゴリズムの性能上鍵となる部分を容易に調整可能なプログラム構成とすることにより，求解性能を向上させることができる．

1. はじめに

本研究の目的は，ソフトウェアとしての汎用性と個々の問題に対する高い求解性能を両立させた非線形最適化プログラムを開発することにある．

非線形最適化問題に対する数値解法としては，ペナルティ関数法・拡張ラグランジュ関数法に代表される変換法，勾配射影法・簡約勾配法に代表される射影法等いくつかの有力な手法が知られている[1),2)]．その中にあって，最近，逐次二次計画法は，最も有望な手法の一つとして注目されている[3)~7)]．その大きな理由は，高い汎用性とともに，比較的大きな規模の非線形最適化問題を解く上で発揮する数値的な性能にある．非線形最適化のための汎用ソフトウェアという面から逐次二次計画法に対する期待は大きい．

逐次二次計画法は，最近，二次計画解法として最小二乗法[8)~10)]，Goldfarb-Idnani 法[11)~13)]が登場することにより，性能の面で向上がもたらされているが，現在まだ発展途上にあり，いくつかの課題が指摘されている．それらは，大きく次の二点に分けられる．

(1) 近似法であるがために，特殊な状況において収束が遅くなったり，問題が解けなくなりうるという現象を解明する理論的な課題．

(2) 数値的な性能評価や性能改善を通じて，汎用的なソフトウェアの形に実装した場合に，高い求解性能を実現するという応用面での課題．

上記の課題(1)の性能評価に関しては，多くの研究努力がなされている．大域的収束性を実現するための条件については，Powell の研究[14)]があり，制約関数の線形近似のために二次計画問題が可能解を持たなくなる現象については，Powell[15)]，刀根[16)]の研究がある．制約境界近傍で探索点の動きが鈍くなる Moratos Effect と呼ばれる現象については，Mayne ら[17)]の研究がある．逐次二次計画法で用いる二次係数行列の更新方法については，矢部ら[18)]の研究がある．

課題(2)に関しては，複数の計算プログラムの作成あるいは移植作業を伴うため，研究事例が限られているのが現状である．ただし，最小二乗法を用いた逐次二次計画法と他の非線形計画法との広範な比較評価が，Schittkowski の研究[19)~22)]に含まれており，二次係数行列の更新方法の性能評価についても言及されている．しかし，逐次二次計画法の二次計画解法に関する評価や他の求解性能の向上手段は含まれていない．本研究は，Schittkowski の研究を補足し，上記課題(2)に関連して，二次計画解法に注目して逐次二次計画法の数値的評価を行い，求解性能の向上手段を検討し，汎用プログラムの実現をはかるものである．以下必要に応じて，逐次二次計画法を SQP，二次計画法を QP と略記する．

† Development of Nonlinear Optimization Program Based on Sequential Quadratic Programming Method by YASUHIRO KOBAYASHI (Energy Research Laboratory, Hitachi, Ltd.), MASAYOSHI TAMURA (Software Works, Hitachi, Ltd.), HISANORI NONAKA and YOICHI SHIGEMATSU (Energy Research Laboratory, Hitachi, Ltd.).

†† (株)日立製作所エネルギー研究所
††† (株)日立製作所ソフトウェア工場
* 現在 北海道大学工学部
Faculty of Engineering, Hokkaido University

2. 逐次二次計画法に基づく最適化プログラム

2.1 逐次二次計画法

逐次二次計画法に関しては，種々の方法が提案されており，アルゴリズムの細部が異なるバリエーションは多い．ここでは，現状で最も有望と考えられるSchittkowski の方法の概要を説明する．

非線形最適化問題は，次式で与えられる．

$$\text{最小化}\quad f(x) \tag{2.1}$$

$$\text{制　約}\quad g_i(x)\geqq 0\quad(\forall i\in MI) \tag{2.2}$$

$$g_i(x)=0\quad(\forall i\in ME) \tag{2.3}$$

ここで，MI, ME は各々不等式制約，等式制約の添字集合である．

SQP のアルゴリズムを**図1**に示す．概略フローのブロックを以下に説明する．

（1） データの初期化

●ステップ①：各パラメータを初期設定する．

●ステップ②：探索出発点 x_0 と近似ラグランジュ乗数 u_0 を入力する．u_0 は未知なので，ゼロ・ベクトルに初期設定する．

●ステップ③：QP の二次係数行列 B_0 を単位行列に初期設定する．ここで，QP の二次係数行列 B_k は，ラグランジュ関数

$$L(x,u)=f(x)-\sum_{i\in ME\cup MI} u_i g_i(x) \tag{2.4}$$

のヘッセ行列

$$\nabla_x^2 L(x,u)=\left(\frac{\partial^2 L(x,u)}{\partial x_i \partial x_j}\right) \tag{2.5}$$

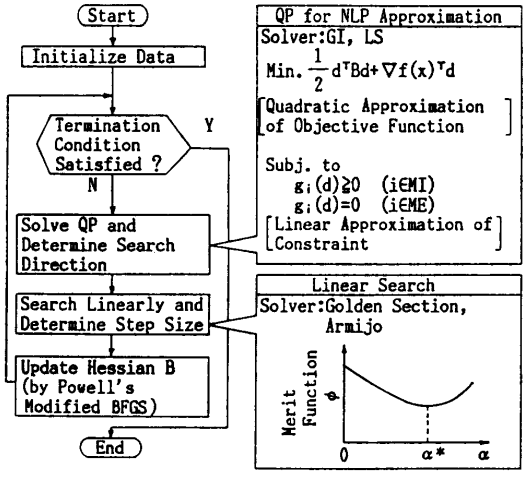

図 1 逐次二次計画法の概略フロー
Fig. 1 General flow of SQP.

の近似行列となるようにする．以下では，行列 B_k を近似ヘッセ行列と称する．下付きの添字 k は，SQP の繰返し数に対応する．

（2） QP の求解

●ステップ④：現在の探索点 x_k におけるラグランジュ関数を最小化するために，ラグランジュ関数の二次近似と制約 g_i の一次近似を用いて，QP 問題を次式のように設定する[3]．

$$\text{最小化}\quad \nabla f(x_k)^{\mathrm{T}} d_k+(1/2)d_k^{\mathrm{T}} B_k d_k \tag{2.6}$$

$$\text{制　約}\quad g_i(x_k)+\nabla g_i(x_k)^{\mathrm{T}} d_k\geqq 0\quad(\forall i\in MI) \tag{2.7}$$

$$g_i(x_k)+\nabla g_i(x_k)^{\mathrm{T}} d_k=0\quad(\forall i\in ME) \tag{2.8}$$

●ステップ⑤：上記(2.6)～(2.8)の QP 問題を解き，解 d_k とラグランジュ乗数 u_k を求める．

（3） 収束の判定1

●ステップ⑥：$d_k=0$ か否かを判断する．$d_k=0$ の場合，(x_k, u_k) が SQP に対する一次の Kuhn-Tucker 条件 (KTC) を満足するので，(x_k, u_k) を局所最適解として SQP を終了する．以下，Kuhn-Tucker 条件を KTC と略記する．

●ステップ⑦：$d_k\neq 0$ の場合，次のブロックの(4)で使うペナルティ・パラメータ r を更新する．

（4） 直線探索

●ステップ⑧：d_k, x_k, をもとにして，メリット関数 $\phi(z)$ の上昇方向を向いた探索方向ベクトル $p_k=(-d_k, u_{k-1}, -u_k)$ を設定する．ここで，メリット関数とは，もとの制約付きの最適化問題を制約なしの最適化問題に変換した目的関数である．また，$z=(x,u)^{\mathrm{T}}$ である．Schittkowski の方法では，メリット関数に拡張ラグランジュ関数を用いている．拡張ラグランジュ関数は，次式で与えられる．

$$\phi(x,u;r)=f(x)-\sum_{i\in M1}\{u_i g_i(x)-(1/2)r g_i(x)^2\}-\sum_{i\in M2} u_i^2/2r \tag{2.9}$$

ここで，

$$M1=ME\cup\{i\in MI;\ g_i(x)\leqq u_i/r\} \tag{2.10}$$

$$M2=\{i\in MI;\ g_i(x)>u_i/r\} \tag{2.11}$$

●ステップ⑨：ϕ の降下方向 $-p_k$ に ϕ を直線探索し，近似的に ϕ を最小化する．

●ステップ⑩：近似的に ϕ を最小にする点を，新たな探索点 z_k とする．

（5） 収束の判定2（図1の概略フローでは省略）

● ステップ⑪：$\nabla_x\phi(z_k; r_k)=0$ か否かを判定する．$\nabla_x\phi(z_k; r_k)=0$ の場合，z_k が，SQP に対する一次の KTC を満たすので，SQP を終了する．

（6） 近似ヘッセ行列の更新

● ステップ⑫：$\nabla_x\phi(z_k; r_k)\neq 0$ の場合，B_k を更新し，ステップ④へ戻る．B_k の更新には，正定値性を保つ Powell の改良型 BFGS 式を用いる．

以下，探索点 z_k が収束するまで，ステップ④～⑫を繰り返す．

2.2 二次計画解法

SQP のステップ⑤で現れる QP 問題の解法として，Goldfarb-Idnani 法（GI 法）と最小二乗法（LS 法）を取り上げ，これらの解法の概要を記述する．SQP アルゴリズムの説明では，QP の解を d としたが，以下の説明では，QP の中での探索点を x とする．等式制約は変数変換によって処理することができるので，不等式制約だけから成る QP 問題の求解のアルゴリズムを，簡単な例を用いて説明する．

Goldfarb-Idnani 法と最小二乗法はともに，有効制約法の一種である．有効制約（Active Set）とは，現在の探索点で等式が成立している制約のことである．制約に対しては添字が与えられているものとし，その添字を用いて制約そのものを指すことにする．SQP の反復の添字（下付きの k）は省略し，QP の反復の添字は上付きの k とする．

（1） Goldfarb-Idnani 法の概要

（a） GI 法のアルゴリズム

GI 法のアルゴリズムを**図2**に示した．GI 法では，目的関数の制約なしの最適点を求めて，それを出発点とする．SQP での探索点を y とするとき，QP の出発点 x^1 は，次式を直接法により解いて求める．

$$Bx^1=\nabla f(y) \qquad (2.12)$$

その後，満足していない制約に注目し，有効制約への追加，有効制約からの削除を行いながら，各時点での最適点を探索する．すべての制約を満足した探索点が，QP 問題の解である．

（b） GI 法の探索点の動き

QP 問題を GI 法によって解く場合の探索点の動きの例を**図3**に示す．図において，探索点の動きは次のようになる．

（i） 与えられた出発点 x^0 とは関係なく，目的関数の制約なしの最適点 x^1 を求め出発点とする．

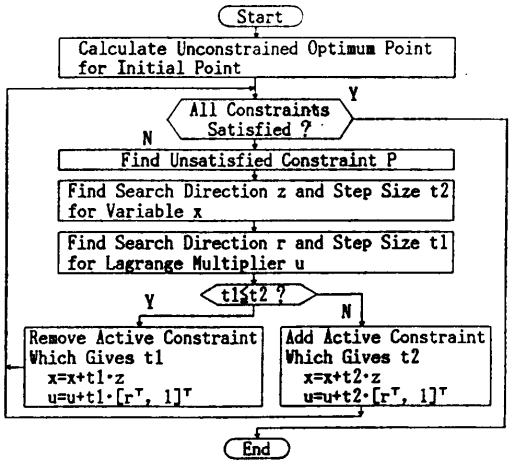

図2 GI 法の概略フロー
Fig. 2 General flow of GI method.

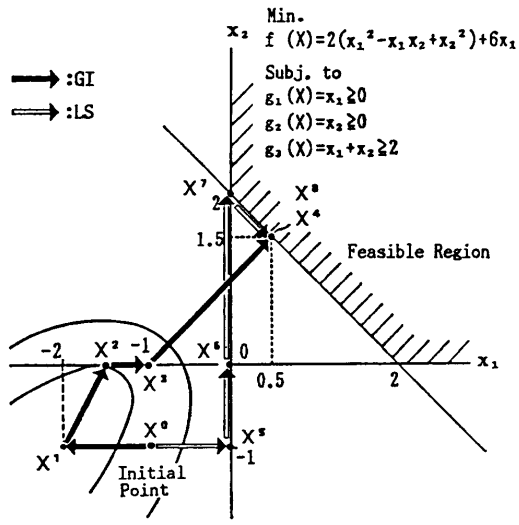

図3 二次計画問題における探索点の動きの例
Fig. 3 Behavior of search paths in QP solution.

（ii） 制約 $g_2(x)\geqq 0$ を有効制約に加えたときの最適点 x^2 を探索点とする．

（iii） 制約 $g_3(x)\geqq 0$ を有効制約に加えるために $g_2(x)=0$ の境界上を進む途中，境界上での最適点 x^3 で停止し，制約 $g_2(x)\geqq 0$ を有効制約から削除する．

（iv） 改めて制約 $g_3(x)\geqq 0$ を有効制約に加えたときの最適点 x^4 を探索点とする．

（v） x^4 において満足していない制約が存在しないので，終了する．

（2） 最小二乗法の概要

（a） LS 法のアルゴリズム

LS 法のアルゴリズムを**図4**に示した．LS 法では，

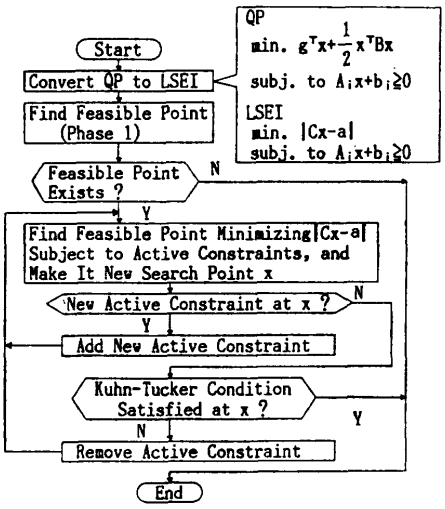

図 4 LS 法の概略フロー
Fig. 4 General flow of LS method.

まず，QP 問題を等価な最小二乗問題 (LSEI) に変換する．次に，二つのフェーズに分けて求解を行う．フェーズ1では，すべての制約を満足する領域（可能領域）の中の点を探索する．フェーズ2では，可能領域の中で最小二乗問題を解くことによって，与えられた有効制約に対し最適点を求める．有効制約への追加，有効制約からの削除を行いながらこの過程を繰り返し，問題の解を得る．

(b) LS 法の探索点の動き

二次計画問題を LS 法を用いて解いた場合の探索点の動きの例を，図3に示す．

(i) 与えられた出発点 x^0 を用いる．ただし，出発点は任意の点でもよい．

(ii) x^0 で制約 $g_1(x)\geqq 0$ が満たされないので，g_1 の増加する法線方向 s^0 に進み，$g_1(x)=0$ の点を次の探索点 x^5 とする．

(iii) x^5 で制約 $g_2(x)\geqq 0$ が満たされないので，$\{s\,;\,g_1(s)=0\}$ かつ g_2 の増加する方向 s^1 に進み，$g_2(x)=0$ の点を次の探索点 x^6 とする．

(iv) x^6 で制約 $g_3(x)\geqq 0$ が満たされないので，$g_2(x)\geqq 0$ を有効制約から落とし，$\{s\,;\,g_1(s)=0\}$ かつ g_3 の増加する方向 s^2 に進む．$g_3(x)=0$ の点を次の探索点 x^7 とする．x^7 は可能解になっている．

(v) x^7 で制約 $g_1(x)\geqq 0$ が KTC を満たさないので，$g_1(x)\geqq 0$ を有効制約から落とし，$\{s\,;\,g_3(s)=0\}$ となる方向 s^3 に進む．$x^7+s^3=x^8$ が KTC を満たすので，x^8 が最適点である．

2.3 求解性能の向上手段

非線形最適化プログラムの求解性能は，基本的な解法の選択とともに，プログラムとしての実現の仕方に依存する．前者については，適用対象とする最適化問題に応じて，原理的に性能が高い基本アルゴリズムを選択することが，後者については，原理的に達成できる性能が数値的な誤差等により損なわれることがないようにアルゴリズムを工夫することが鍵となる．文献20) でも，後者が求解性能の上で重要であることを指摘している．ここでは，後者の点に注目し，求解性能の向上手段について検討する．

(1) プログラムの汎用性と求解性能

種々の応用分野で実際に非線形最適化問題を解く場合，対象とする問題に合わせて専用のプログラムを作る例が少なくない．この理由には，プログラムの求解性能に関連して，汎用プログラムに求め難い次のような利点があるためである．

(a) 個々の問題に合わせて適切な解法を選択できる．

(b) 特定の分野や問題に関する知識を，きめ細かくプログラムに反映させることが容易である．

プログラムの汎用性を重視する立場から，このようなことがらを汎用プログラムという枠組みの中で反映し，高い求解性能を実現可能としていくことが望ましい．以下に，上記の二点に関連して，今回開発した SQP プログラムで採用した求解性能の向上手段について記す．

(2) 解法の選択の範囲の拡大

SQP のアルゴリズムで重要な要素は，図1のブロックにも現れているように，QP，直線探索，近似ヘッセ行列の更新の三点である．

(a) QP 解法については，ここでは，2.2 節で説明した2種類の QP 解法を採用した．その理由は，3.2 節，4.2 節で後述するように，これらの解法は，対照的な特徴を有するので，解法の選択の範囲を広げるのに有効であるためである．また，解法として対照的であるが故に，組み合わせて使うことにより，4.3 節で記述するように両者の長所を発揮させることも可能となる．

(b) 直線探索については，(i) Armijo の方法，(ii) 黄金分割法，(iii) 二次補間による方法等が知られている．ここでは，(i) と (iii)，(ii) と (iii) を組み合わせた手法を採用している．

(c) 近似ヘッセ行列の更新についても，Powell

の改良型 BFGS 式を基本として，バリエーションがあるが，文献 18)，20) で言及されており，ここでは議論しない．

（3） 個々の問題に適合した性能の調整

個々の問題に対して，専用のプログラムのような求解性能を汎用プログラムで実現するためには，解法，解法の特性を決めるパラメータを標準的な仕様から個々の問題に適合するような仕様に容易に調整可能とする必要がある．ここでは，求解モジュールと分離した制御モジュールを用いて，このような修正に対応できるようなプログラム構成を採用している．以下に，このプログラムの特徴的な機能を例として，この手段を説明する．

（a） SQP でのリスタート

SQP において求解が失敗する主な原因の一つに，計算の過程で近似ヘッセ行列の正定値性が数値的に十分でなくなるという現象がある．この問題に対する実際的な対策として，計算の途中で近似ヘッセ行列を単位行列に再度初期化する方法が考えられる．これをリスタートと称する．リスタートにより，数値誤差のために今までの計算の過程で危うくなった行列の正定値性を救うことができる．しかし，近似ヘッセ行列を単位行列にすることは最急降下法により探索することを意味し，頻繁にリスタートを行うことは，収束の特性の良くない所で計算を行う結果となるので，計算の効率を低下させることになる．

したがって，適切なタイミングでリスタートを行うことが，SQP の数値的性能を高める上で重要となる．そのため，ここでは，処理の流れの中の複数の点で近似ヘッセ行列の正定値性をモニタし，リスタートの判定に反映させている．SQP のアルゴリズムの中でリスタートの判定を行う位置を**図 5** に示した．アルゴリズムの各ステップは，2.1 節で説明したステップに対応している．リスタート基準として図に示した次の 5 種類を採用している．

（i） 近似ヘッセ行列のコレスキー分解が失敗すること．

（ii） $d_k^T B_k d_k$ の値が下限値よりも小さくなること．ただし，d_k は QP の解，B_k は近似ヘッセ行列である．

（iii） B_k の関数 δ_k の値が下限値よりも小さくなること．δ_k は次式で与えられる．

$$\delta_k = \min\{d_k^T B_k d_k / |d_k|^2, \delta_{k-1}\} \quad (2.13)$$

（iv） 直線探索のステップ幅が連続して下限値よりも小さくなること．

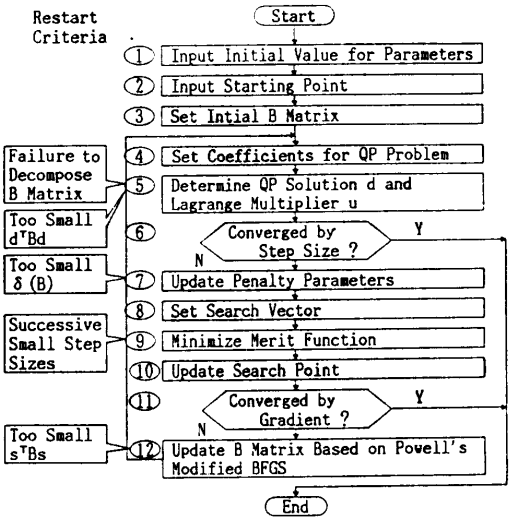

図 5 リスタート基準と逐次二次計画法のフロー
Fig. 5 Restart criteria in detailed SQP flow.

（v） Powell の改良型 BFGS 式に現れる項 $s^T B_k s$ の値が下限値よりも小さくなること．ただし，$s = x_k - x_{k-1}$ である．

これらの各基準は，具体的には，各項目に対応して決まる特定のパラメータが，あらかじめ設定した限界値を越えるか否かで判定する．限界値は，現状では，経験的に決める必要があり，系統的に設定することを支援する手段が有効となる．

（b） 探索方向ベクトルの補正

QP の解において，有効であるべき制約条件が数値的にも十分有効となっていない場合，すなわち，数値誤差のため制約条件の等式が成立しない場合，数値的性能に悪影響を及ぼすと考えられる．このような問題に対して，ここでは，QP の求解の過程で誤差補正ベクトルを付加して探索方向を補正している．このとき，QP の解法に応じて制約条件の等式からの相違の程度に下限値をあらかじめ設定しておき，それよりも相違が小さくなった場合に補正する．

探索点ベクトルおよび探索方向ベクトルを補正する場合について，説明のための簡単な例を**図 6** に示した．図では，①で探索点ベクトルの補正，②で探索方向ベクトルの補正を示している．図の①では，直線探索により x_k から次の探索点を求める際，未定義領域に関する制約条件を満たさない探索点 x_{k+1} を求めようとするが，探索点ベクトルの補正により x_{k+1} を x_{k+1} に引き戻す処理を行う．図の②では，QP を解いて現在の探索点から次の探索方向を求める際，制約

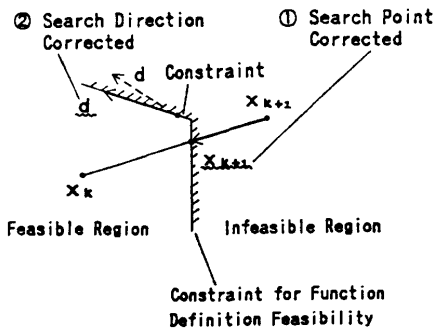

図 6 探索点ベクトルおよび探索方向ベクトルの補正
Fig. 6 Correction of search point and direction vectors.

条件の等式の境界とわずかに相違する探索方向 d を求めようとするが，探索方向ベクトルの補正により d を制約条件の等式の境界の方向 d に修正する処理を行う．

（4） プログラムの高速化

計算の効率は，プログラムの求解性能の尺度の一つである．実際の応用では，類似の問題を繰り返し解くことが多いので，プログラムとしての信頼性を維持した上で，プログラムの高速化をはかる必要がある．特に，(a)マシンの性能が限られ，かつリアルタイム性が求められる場合，(b)問題の規模が大きく計算時間が長くかかる場合には，プログラムの高速化が重要となる．汎用プログラムの高速化については，(b)に関連し，例えば，アルゴリズムのベクトル化等の研究開発が重要な課題と考えるが，本論文では割愛した．

3. 二次計画解法の性能評価

SQP の性能の上で，QP 解法の性能は決定的な影響を与える．ここでは，SQP に組み込んだ場合とは別個に，単独で GI 法，LS 法の性能評価を行った．

3.1 テスト問題生成プログラム

数値的性能の評価にはテスト問題の選択が重要である．標準的テスト問題集[21),22)]にある問題だけではなく，条件を制御したテスト問題を用いることが有効と考える．ここでは，QP 解法の基本性能評価のために，テスト問題生成プログラムで難度の高いテスト問題を作成し，利用した[19)]．テスト問題生成プログラムが，問題の性質や難しさを制御できる条件としては，次のものを考える．

（1） x の次元 n，不等式制約の集合 MI，等式制約の集合 ME，解で有効になる不等式制約の集合 MA，

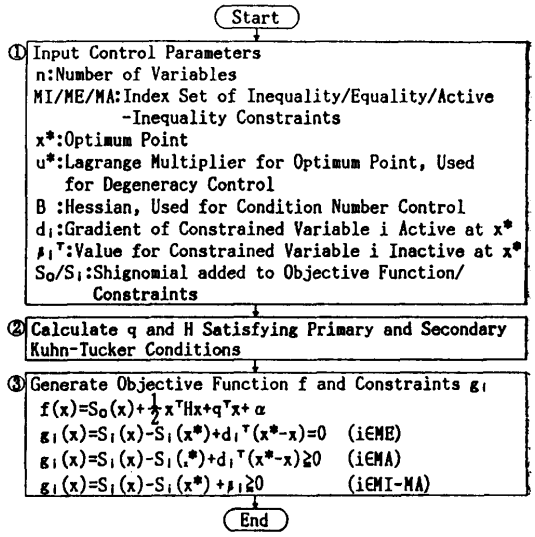

図 7 テスト問題生成プログラムの概略フロー
Fig. 7 General flow of test problem generator.

（2） 解 x^*，対応するラグランジュ乗数 u^*，

（3） 近似ヘッセ行列 B の条件数

ここで，B が正定値の場合，目的関数は，n 次元空間の楕円を表し，条件数は楕円の (長軸)^2/(短軸)^2 を表す．したがって，条件数が大きいほど，目的関数は扁平となり，長軸付近では目的関数の変化が小さい．

テスト問題生成プログラムのアルゴリズムを**図 7** に示す．テスト仕様は，図 7 のステップ①に現れる (n, MI, ME, MA, x^*, u^*, B) で記述される．あらかじめ設定したテスト仕様を満たすテスト問題は，ステップ③に現れる式を用いて生成する．

3.2 二次計画問題解法の性能比較

テスト問題生成プログラムで生成した QP 問題を GI 法，LS 法を用いて解き，それらの性能を比較した．ここでは，参考のため同じ問題を従来解法である微分法でも解いている．

テスト問題 3 題に対する計算結果を**表 1** に示す．この表には，問題を記述するために以下の項目を挙げた．

① n ：変数の数

② m_i：不等式制約条件の数

③ m_e：等式制約条件の数

④ m_a：最適点 x^* で有効な制約条件の数

⑤ λ 比：ラグランジュ関数 $L(x,u)$ のヘッセ行列 $\nabla_x^2 L(x,u)$ を，x^* での一次近似空間 $\{y : y^T\nabla_x g_i(x^*)=0 \text{ for } \forall i\in I(x^*)\}$ に射影したヘッセ行列 H^* の条件数，すなわち，H^* の最大固有値 λ^*_{max} と最小

表 1 二次計画問題の計算結果
Table 1 Numerical results of QP computation.

No.	QP Problem	Solver	Av. Matched Digits x^*	u^*	f^*	$\|\nabla L\|$ r^*	No. of Iterations CPU time*
1	n:5,mi:20, me:2,me:0, B:5x5 Hilbert Matrix, Cond. No.:10^7	LS	10.	10.	10.	5.0×10^{-8} 2.2×10^{-11}	20(P1:5)** 24.(ms)
		GI	8.4	10.	10.	9.5×10^{-8} 1.1×10^{-13}	13 4.8(ms)
2	n:10,mi:20 me:2,me:0, B:10x10 Hilbert Matrix Cond. No.:10^{15}	LS	2.1	10.	10.	4.1×10^{-9} 3.4×10^{-13}	14(P1:3) 28.(ms)
		GI	0.	9.8	4.	14. 1.3×10^{-2}	10 6.3(ms)
3	n:30,mi:20, me:3,me:0, B:30x30 Hilbert Matrix Cond. No.:10^{45}	LS	0.	9.7	6.	2.3×10^{-6} 0.1×10^{-7}	27(P1:10) 220.(ms)
		GI	0.	9.3	0.	4.6×10^{2} 1.7×10^{4}	61 97.(ms)

* Machine:HITAC-M200H
** P1:Phase 1 of Optimization (Search for Feasible Point)

固有値 $\lambda^*_{\min}$ の比．ここで，$I(x^*)$ は x^* での有効制約の添字集合である．

⑥ u^* 比：最適点 x^* でのラグランジュ乗数 u^* 成分の最大絶対値と最小絶対値の比．制約の一次従属性を計る退化度であり，大きいほど一次従属性が強い．

テスト問題では，B を条件数が制御できるヒルベルト行列にした．ヒルベルト行列の条件数は，およそ $\exp(3.5n)$ である．以下，B の条件数を $\mathrm{cond}\,B$ と記す．

表1には，QP 解法の性能を記述するために以下の項目を挙げている．

① 変数の計算解 x^* の平均正解桁数

② ラグランジュ乗数 u^* の計算解の平均正解桁数

③ 目的関数の計算解 f^* の正解桁数

④ $\|\nabla L\|$：ラグランジュ関数 $L(x,u)$ の計算解 (x^*,u^*) における勾配 $\nabla L(x^*,u^*)$ の二乗ノルム．$\|\nabla L\|$ は，計算解での KTC の不満足度を計る指標[8]である．理論解では KTC を満足するので，$\|\nabla L\|=0$ となる．

⑤ r^*：計算解 (x^*,u^*) での制約条件の不満足度を計る指標であり，次式で与える．

$$r^*=\sum_{i\in ME}|g_i(x^*)|-\sum_{i\in MI}|\min\{0,g_i(x^*)\}| \tag{3.1}$$

計算解 x^* が，すべての制約条件を満足すると $r^*=0$ となる．

⑥ 探索の繰り返し回数

⑦ 計 算 時 間

（1） 問題1

問題1は，$n=5$, $u^*_{\max}/u^*_{\min}=1.360$, $\mathrm{cond}\,B=3.9\times10^7$ であり，B だけが悪条件の問題である．計算の結果，LS 法，GI 法だけでなく，微分法でも求解に成功している．解の精度は全般的に LS 法が GI 法よりも優れていたが，計算速度は GI 法が LS 法よりも5倍速い．微分法は，精度は GI 法よりも低く，計算速度は LS 法と同程度であった．

（2） 問題2

問題2は，$n=10$, $u^*_{\max}/u^*_{\min}=11.56$, $\mathrm{cond}\,B=1.5\times10^{15}$ の問題である．計算の結果，LS 法，GI 法でも変数の計算解として高い精度が得られない．LS 法では，KTCの 不満足度が小さく，KT (Kuhn-Tucker) 点に近い点を求めている．しかし，GI 法では，LS 法ほど，KT 点に近い点を求められない．また，LS 法の解では有効制約が正解に一致しているのに対し，GI 法の解では有効制約が一件異なっている．GI 法に比して，LS 法の精度が全般的に優れている．微分法の精度は GI 法よりも低い．

（3） 問題3

問題3は，$n=30$, $u^*_{\max}/u^*_{\min}=442$, $\mathrm{cond}\,B=3.9\times10^{45}$ の問題である．計算の結果，GI 法と微分法では KT 点に近い点を求められないが，LS 法は KT 点に近い点を求めている．この場合，LS 法でも，変数の計算解の精度が低く，有効でない制約を有効制約と見なしている．

（4） 求解が困難となる限界

以上の3題の条件を補間するような次の2題の問題を解いた．

- 問題 1.5：$n=8$, $\mathrm{cond}\,B=1.4\times10^{12}$
- 問題 2.5：$n=12$, $\mathrm{cond}\,B=1.7\times10^{18}$

5題の QP 問題について GI 法と LS 法の求解の性能を**表2**に整理した．ここで，性能の目安を表すための

表 2 二次計画解法の数値的性能の比較
Table 2 Comparison of numerical performance of QP solvers.

Solver	Problem	1	1.5	2	2.5	3
	No. of Variables	5	8	10	12	30
	Hessian Cond. No.	10^7	10^{12}	10^{15}	10^{18}	10^{45}
GI		○	○	△	×	×
LS		○	○	○	○	△

○ : Av. Digits Matched ≧ 1.0
△ : $|\nabla_x L|\leqq10^{-3}$
× : $|\nabla_x L|>10^{-3}$

記号の意味は次のとおりである.

○：変数の計算解の平均正解桁数が1.0以上である.

△：KTC の不満足度が 10^{-3} 以下である.

×：KTC の不満足度が 10^{-3} を越える.

この表の○の意味で，GI 法と LS 法による求解が困難となる限界は，ヘッセ行列の条件数にして，GI 法は 10^{12} であり，LS 法は 10^{18} である.

(5) QP 解法としての特徴

これらの計算結果より，LS 法，GI 法を以下のように特徴づけることができる.

(a) LS 法は，GI 法よりも，ヘッセ行列の条件数が大きい，難しい問題を解くことができ，精度の高い解を与える.

(b) GI 法は，LS 法よりも計算効率が高い．この計算ケースでは，前者の計算効率は後者の数倍であった.

(c) 両者とも従来の微分法よりも QP 解法として高い数値的性能を示し，GI 法は計算効率を，LS 法は計算精度を重視した解法と言える.

4. 逐次二次計画法の性能評価

非線形最適化の標準的テスト問題集の中から規模が大きく難度が高い問題を選択して解くことによって，SQP と SUMT (Sequential Unconstrained Minimization Technique)[1),2),23)] について性能を評価した．ここで，SUMT は，比較のために従来解法の例として用いる.

4.1 テスト問題の内容

テスト問題集[21),22)] の中から，目的関数や制約条件の非線形性が強い問題と，変数の数や制約条件の数が多い問題を選んだ．このような難問に対しては SQP のリスタートが有効に機能する．例題として，次の3題を示す.

(1) 問題4

問題4は，テスト問題集[21)] の No. 117 である．15変数，三次目的関数，二次不等式制約条件5，下限の不等式制約条件 15 の問題である．u^* 比，λ 比ともに，テスト問題集の中では比較的大きい.

(2) 問題5

問題5は，テスト問題集[21)] の No. 109 である．9変数，三次目的関数，二次不等式制約条件2，一次不等式制約条件2，二次関数と三角関数の積の項を含む等式制約条件6，上下限の制約条件 16 の問題である．u^* 比は非常に大きい.

(3) 問題6

問題6は，テスト問題集[21)] の No. 114 である．10変数，二次目的関数，二次不等式制約条件2，三次不等式制約条件2，一次不等式制約条件4，上下限の制約条件 20，一次等式制約条件1，有理式の等式制約条件2の問題である．λ^* 比は大きくないが，u^* 比は大きい.

4.2 計算結果と検討

テスト問題に対する SQP と SUMT の計算結果を**表3～表5**に示す．この表には，SQP 問題を記述するために，3.2 節で説明したもののほかに，以下の項目を挙げている.

- ARMJ：SQP の直線探索に用いる，二次補間法と組み合わせた Armijo 法
- GLD：SQP の直線探索に用いる，二次補間法と

表3 問題4の計算結果
Table 3 Numerical results for problem no. 4.

Method *1)	Objective Function $f(x^*)$	$\|\nabla L\|$ r^*	CPU (s) *3) No. of Iterations	No. of Function Calls *2)
SUMT UO: FP	32.3546	69.7 0.0	11.2 7	F 7844 G 157764 DF 462 DG 9280
SQP QP: GI OS: GLD	32.3487	1.1×10^{-5} 0.0	1.05 16	F 54 G 1080 DF 16 DG 320
SQP QP: GI OS: Armj	32.3486	1.1×10^{-5} 0.0	1.06 16	F 52 G 1040 DF 16 DG 320
SQP QP: LS OS: GLD	32.3487	6.2×10^{-6} 0.0	1.24 18	F 49 G 980 DF 18 DG 360
SQP QP: LS OS: Armj	32.3487	4.4×10^{-6} 0.0	1.18 17	F 54 G 1000 DF 17 DG 340

*1) UO: Unconstrained Optimizer, FP: Fletcher-Powell, QP: QP Solver, OS: 1-Dim. Searcher, GLD: Golden Section, Armj: Armijo

*2) F: Objective Function, G: Constraints, DF: Derivative of Objective Function, DG: Derivative of Constraints

*3) Machine: HITAC-M 200 H

表 4 問題5の計算結果
Table 4 Numerical results for problem no. 5.

QP Solver	Variable x^*	Lagrange Multiplier u^*	Objective Function f^*	$\|\nabla L\|$ r^*	No. of Iterations CPU time *1)
GI	(675.025 , 1134.02 , 0.13349, −0.37119, 252.000 , 252.000 , 201.466 , 426.619 , 368.488)	$u_{16}=3.9\times10^{0}$ $u_{17}=5.9\times10^{0}$ $u_{21}=-8.7\times10^{-2}$ $u_{22}=-8.0\times10^{-2}$ $u_{23}=-1.2\times10^{-1}$ $u_{24}=2.1\times10^{-14}$ $u_{25}=6.9\times10^{-14}$ $u_{26}=1.8\times10^{-2}$ $u_i=0$ $(i\neq16,17,21\sim26)$	5.4×10^{3}	3.4×10^{-9} 8.7×10^{-10}	459 13.1(s)
LS	(675.025 , 1134.02 , 0.13349, −0.37119, 252.000 , 252.000 , 201.466 , 426.619 , 368.488)	$u_{16}=3.9\times10^{0}$ $u_{17}=5.9\times10^{0}$ $u_{21}=-8.7\times10^{-2}$ $u_{22}=-8.0\times10^{-2}$ $u_{23}=-1.2\times10^{-1}$ $u_{24}=3.4\times10^{-14}$ $u_{25}=3.4\times10^{-14}$ $u_{26}=1.8\times10^{-2}$ $u_i=0$ $(i\neq16,17,21\sim16)$	5.4×10^{3}	6.6×10^{-6} 1.4×10^{-9}	395 12.6(s)

*1) Machine: HITAC-M 280 H

組み合わせた黄金分割 (Golden Section) 法

- F : 目的関数の関数評価回数
- DF: 目的関数の勾配の関数評価回数
- D : 制約条件の関数評価回数
- DG: 制約条件の勾配の関数評価回数

これらの表には，SQP，SUMT の性能を記述するために，3.2 節で説明したもののほかに，以下の項目を挙げている．

- α: 直線探索での探索幅
- r: メリット関数のペナルティ・パラメータ

(1) 問題4

表3の計算結果より，次のことがわかる．

(a) SQP は，全般的に，SUMT より変数の計算解の精度が良い．KTC の不満足度と制約条件の不満足度の点より，最も精度が良いと考えられるケースは，LS 法と ARMJ を組み合わせた SQP である．このケースでは，二次係数行列のリスタート機能が動作している．SQP のケースを通じて，変数の計算解は，テスト問題の正解に対して6桁一致している．ラグランジュ乗数の計算解は，LS 法と ARMJ を組み合わせた SQP のケースを基準として7桁まで一致する．目的関数の正解として8桁まで与えられているが，SQP のケースの間では 11 桁まで一致する．SUMT では，最も性能が高い FP を用いたケースでも，変数および目的関数の計算解の正解桁数は3である．

(b) SQP での探索点の個数，関数評価回数，CPU 時間は，各々，SUMT の 1/28 以下，1/100 以下，1/8 以下である．SQP での関数評価回数が，SUMT での関数評価回数の 1/100 以下であるにもかかわらず，CPU 時間は 1/8 以下にしか減少していない．SQP の中では，GI 法，LS 法の計算速度の相違は緩和されているが，GI 法による SQP は，LS 法による SQP よりも4割程度計算速度が速い．

(c) SQP での GI 法，LS 法の各々に関し，ARMJ のほうが GLD より計算時間が若干小さい．

(2) 問題5

表4は，問題5に対する計算結果である．この結果より，次のことがわかる．

(a) ここでは，同じ収束条件の下で計算して，GI 法を用いた SQP がより精度の良い結果を与え，LS 法を用いた SQP がより効率の良い結果を与えている．一般には，GI 法を用いた SQP および LS 法を用いた SQP の求解性能の上での特徴は，大まかな傾向として，GI 法および LS 法の特徴を反映している．

表 5 問題 6 の計算結果
Table 5 Numerical results for problem no. 6.

[a] Calculation process. *1) Machine: HITAC-M 200 H

K	x-distance $\|x_k-x^*\|$	u-distance $\|u_k-u^*\|$	function value f^*	vector length $\|d_k\|$	step size α	penalty parameter r	CPU [ms] *1) (No. of Iterations)
0	3.82×10^{3}	3.12×10^{2}	-8.72×10^{2}	1.30×10	7.43×10^{-1}	2^{11} (=2048)	5.21 (8)
1	3.82×10^{3}	2.84×10^{2}	-1.25×10^{3}	1.24×10	1.0	2^{12} (=4096)	7.50 (13)
2	3.82×10^{3}	1.75×10^{2}	-1.48×10^{3}	6.03	8.19×10^{-1}	2^{16} (=65536)	8.54 (14)
5	3.82×10^{3}	2.62×10^{2}	-1.52×10^{3}	1.63×10	6.18×10^{-1}	2^{16}	9.38 (15)
10	3.82×10^{3}	2.82×10^{2}	-1.55×10^{3}	5.10×10^{-1}	1.11×10^{-4}	2^{16}	8.54 (15)
20	3.77×10^{3}	2.71×10^{2}	-1.56×10^{3}	4.28×10^{2}	1.08×10^{-3}	2^{16}	12.08 (22)
30	3.73×10^{3}	2.57×10^{2}	-1.5659×10^{3}	4.03×10^{2}	8.60×10^{-3}	2^{16}	9.79 (18)
40	3.70×10^{3}	2.46×10^{2}	-1.5706×10^{3}	3.83×10^{2}	7.98×10^{-3}	2^{16}	12.29 (20)
50	3.67×10^{3}	2.37×10^{2}	-1.5746×10^{3}	3.65×10^{2}	7.79×10^{-3}	2^{16}	11.04 (20)

[b] Final result.

QP Solver	Variable x^*	Lagrange Multiplier u^*	Objective Function f^*	$\|\nabla L\|$ r^*	No. of Iterations CPU time *1)
GI	(1698.09 , 15818.6 , 54.1027 , 3031.23 , 2000.00 , 90.1154, 95.0000 , 10.4933, 1.56164, 153.535)	$u_2 = 7.0\times10^{1}$ $u_3 = 3.1\times10^{2}$ $u_5 = 6.8\times10^{-1}$ $u_6 = 2.3\times10^{2}$ $u_{23} = 8.8\times10^{-1}$ $u_{25} = 1.7\times10^{2}$ $u_{29} = -4.2\times10^{0}$ $u_{30} = 7.5\times10^{1}$ $u_{31} = 5.9\times10^{1}$ $u_i = 0$ ($i \neq 2, 3, 5, 6, 23, 25, 29, 30, 31$)	-1.8×10^{3}	-1.7×10^{-7} 4.9×10^{-10}	431 14.7(s)

この結果では，QP の解法の特徴が他の効果に埋もれている．

(b) この問題に対しては，QP の解法として GI 法を用いた場合も，LS 法を用いた場合も，求解の過程で二次係数行列のリスタート機能が動作している．問題 4 でも，ケースによってはリスタート機能の動作が観測されているが，この問題では，リスタート機能がより本質的な役割を果たしている．

(3) 問題 6

表 5 は，問題 6 に対し，GI 法と GLD を組み合わせた SQP を用いて計算した結果の例である．計算の経過を[a]に，結果を[b]に示す．表 5[a]のデータからも，SQP の探索点の収束は遅いことが認められる．速く収束した問題との比較から，表 5 に関し，次のことがわかる．

(a) $\|x_k-x^*\|$ の減少率が $\|u_k-u^*\|$ の減少率より小さい．

(b) $\|d_k\|$ のオーダは $\|x_k-x^*\|$ のオーダより小さい．

(c) k が増加しても，探索幅 α が 1.0 に近づかず，小さいままである．

(d) ペナルティ・パラメータ r が，探索出発点から，既に 10^3 を超えている．

これらのことより，ペナルティ・パラメータ r が大きくなったために，メリット関数が，x_k から d_k 方向に狭い谷になってしまい，直線探索で，ほとんど探索点が動けなくなったものと考える．この現象を防ぐための対策として，メリット関数に狭い谷が生じないようにペナルティ・パラメータ r の更新方法を改良し収束を速くすることが考えられる．

4.3 GI 法と LS 法の併用

GI 法は計算速度に，LS 法は計算精度に有利な面を持つ手法である．GI 法と LS 法を組み合わせて，KT 点への接近のフェーズを GI 法を用いて効率良く行い，KT 点への収束のフェーズを LS 法を用いて精度良く行うアプローチが考えられる．

表 6 GI 法, LS 法, GI・LS 併用法の比較
Table 6 Numerical comparison of GI, LS and GI・LS combined methods.

QP Solver	Variable x^*	Lagrange Multiplier u^*	Objective Function f^*	$\|\nabla L\|$ r^*	No. of Iterations CPU time *1)
GI	(−11.0875 , 425.196 , −557.998 , 381.783 , 132.610 , 794.821 , −3.34855, −677.847 , 791.995 , −2.42416)	$u_4=2.5\times10^{-6}$ $u_8=7.8\times10^{4}$ $u_9=9.0\times10^{5}$ $u_i=0$ $(i\neq4,8,9)$	3.8×10^{3}	12.62 2.7×10^{-13}	10 6.25 (ms)
LS	(1.69267, 132.886 , 196.967 , 104.850 , 86.7039 , 96.6905 , 137.337 , 48.7423 , 130.702 , 194.461)	$u_8=7.8\times10^{4}$ $u_9=9.0\times10^{5}$ $u_i=0$ $(i\neq8,9)$	3.8×10^{3}	3.8×10^{-9} 3.4×10^{-13}	14 (P1: 11) *2) 27.7 (ms)
GI+LS (Combined)	(1.69267, 132.886 , 196.967 , 104.850 , 86.7039 , 96.6905 , 137.337 , 48.7423 , 130.702 , 194.461)	$u_8=7.8\times10^{4}$ $u_9=9.0\times10^{5}$ $u_i=0$ $(i\neq8,9)$	3.8×10^{3}	3.8×10^{-9} 3.4×10^{-13}	12 (GI: 10, LS: 2) 12.71 (ms) (GI: 6.25, LS: 6.46)

*1) Machine: HITAC-M 200 H
*2) P1: Phase 1 of Optimization (Search for Feasible Point)

GI 法と LS 法を併用する方法を，GI 法では解けなかった問題2を例に検討した．**表6**に，GI 法，LS 法，上記の併用した方法 (GI+LS) により問題2を解いた結果を表6に比較した．ただし，LS 法の場合にも，GI 法と同様に制約なしの最適点を出発点として用いた．併用した方法では LS 法で得られた正解を，LS 法よりも効率良く得ることができた．この結果より，GI 法と LS 法を併用する方法がバランスの良い QP 解法を実現しうることを確認した．

5. まとめ

逐次二次計画法を基本アルゴリズムに採用し，汎用的なソフトウェアの枠組みの中で，個々の問題に対して高い求解性能を実現可能とする非線形最適化プログラムを開発した．本プログラムは，次のような特長を有する．

（1） 二次計画解法として，効率に優れる Goldfarb-Idnani 法と，精度に優れる最小二乗法とを組み合わせて利用可能である．テスト問題生成プログラムで作成した二次計画問題を用いて，これら両者の数値的性能を評価し，性能上の特性を明確化した．テスト問題集より選んだ非線形最適化問題を用いて，逐次二次計画法の数値的性能を評価し，逐次二次計画法に組み込まれた段階でも，上記の手法の特長を確認した．このように対照的な二次計画解法を用いることにより，解法の選択の範囲を拡大できる．

（2） 求解過程で近似ヘッセ行列を多重にモニタし，正定値性が悪化した行列を再初期化し，処理をリスタートする．ここでは，逐次二次計画法の処理の流れの中の五つの点で行列の正定値性をモニタし，リスタートの判定に反映させている．この判定に用いるような基準は，求解モジュールと分離した制御モジュールで，標準的な仕様から個々の問題に適合するような仕様に容易に調整可能としている．

また，(1)に関しては，Goldfarb-Idnani 法により最適点近傍に接近させ，最小二乗法により精密解を求めるという形で，両者を併用することにより，バランスの良い二次計画解法を実現しうることを数値的に確認した．

汎用的なソフトウェアの枠組みの中で，個々の問題に対する数値的性能を向上させるためには，本報に記したような求解性能の向上手段をヒューリスティックに組み合わせていく必要がある．このような研究開発を効率良く進めるには，プログラミング環境を整備していくことも重要な課題である．

本論文では触れていないが，計算機自体の発展に対応して，ベクトルプロセッサ向きのアルゴリズムによりプログラムの高速化をはかったり，グラフィックワークステーションの下で，ユーザの思考プロセスとの親和性を考慮した使いやすいマンマシンシステムを

実現していくことも，近い将来における課題と考える．

参考文献

1) 今野，山下：非線形計画法，日科技連 (1984).
2) 志水，相吉：数理計画法，昭晃堂 (1986).
3) Schittkowski, K.: The Nonlinear Programming Method of Wilson, Han and Powell with an Augmented Lagrangian Type Line Search Function Part 1: Convergence Analysis, *Numerische Mathematik*, Vol. 38, pp. 83-114 (1981).
4) Yamashita, H.: Quadratic Programming Approximation for Nonlinear Optimization, 第7回数理計画シンポジウム論文集, pp. 35-58 (1983).
5) Fukushima, M.: A Successive Quadratic Programming Algorithm with Global and Superlinear Convergence Properties, *Math. Program.*, Vol. 35, pp. 253-264 (1986).
6) Powell, M. J. D. and Yuan, Y.: A Recursive Quadratic Programming Algorithm That Uses Differentiable Exact Penalty Functions, *Math. Program.*, Vol. 35, pp. 265-278 (1986).
7) Han, S. P.: Superlinearly Convergent Variable Metric Algorithms for General Nonlinear Programming Problems, *Math. Program.*, Vol. 11, pp. 263-282 (1976).
8) Stoer, J.: On the Numerical Solution of Constrained Least-Squares Problems, *SIAM J. Num. Anal.*, Vol. 8, No. 2 pp. 63-82 (1971).
9) Crane, R. L. et al.: LCLSQ: An Implementation of an Algorithm for Linear Constrained Linear Least-Squares Problems, ANL-80-116, Argonne National Laboratory (1980).
10) Schittkowski, K. and Stoer, J.: A Factorization Method for the Solution of Constrained Linear Least Squares Problems Allowing Subsequent Data Changes, *Numer. Math.*, Vol. 31, pp. 431-463 (1979).
11) Goldfarb, D. and Idnani, A.: A Numerically Stable Dual Method for Solving Strictly Convex Quadratic Programming, *Math. Program.*, Vol. 27, pp. 1-33 (1983).
12) Powell, M. J. D.: Study on the Quadratic Programming Algorithm of Goldfarb and Idnani, *Math. Program.*, Vol. 29, pp. 46-61 (1985).
13) 矢部，高橋，八巻，宮田，本郷：凸2次計画問題に対する Goldfarb and Idnani 法について，青山コンピュータサイエンス，Vol. 14, No. 2, pp. 47-50 (1986).
14) Powell, M. J. D.: Convergence Properties of Algorithms for Nonlinear Optimization, SIAM Rev., Vol. 28, No. 4, pp. 487-500 (1986).
15) Powell, M. J. D.: Variable Metric Methods for Constrained Optimization, in Balchem, A., Grotschel, M. and Korte, B. (eds.), *Mathematical Programming: The State of the Art*, pp. 288-311, Springer-Verlag (1983).
16) Tone, K.: Revisions of Constraint Approximations in the Successive QP Method for Nonlinear Programming Problems, *Math. Program.*, Vol. 29, pp. 144-152 (1983).
17) Mayne, D. Q. and Polak, E.: A Superlinearly Convergent Algorithm for Constrained Optimization Problems, *Math. Program. Study*, Vol. 16, pp. 45-61 (1982).
18) 矢部，高橋，八巻，分散型準ニュートン更新公式の逐次2次計画法への応用，情報処理学会論文誌，Vol. 27, No. 12, pp. 1147-1154 (1986).
19) Schittkowski, K.: Nonlinear Programming Codes, Information, Tests, Performance, *Lecture Notes in Economics and Math. Systems*, Vol. 183, Springer-Verlag (1980).
20) Schittkowski, K.: The Nonlinear Programming Method of Wilson, Han and Powell with an Augmented Lagrangian Type Line Search Function Part 2: An Efficient Implementation with Linear Least Squares Subproblems, *Numerische Mathematik*, Vol. 38, pp. 115-127 (1981).
21) Hock, W. and Schittkowski, K.: Test Examples for Nonlinear Programming Codes, *Lecture Notes in Economics and Math. Systems*, Vol. 187, Springer-Verlag (1981).
22) Schittkowski, K.: More Test Examples for Nonlinear Programming Codes, *Lecture Notes in Economics and Math. Systems*, Vol. 282, Springer-Verlag (1987).
23) Kuester, J. L. and Mize, J. H.: *Optimization Techniques with Fortran*, McGraw-Hill (1973).

(平成元年1月17日受付)
(平成2年1月16日採録)

小林 康弘 (正会員)

昭和22年10月21日生．昭和45年3月東京大学工学部原子力工学科卒業．昭和50年3月同大学院原子力工学専攻博士課程修了．同年4月(株)日立製作所入社．昭和53年4月より同社エネルギー研究所勤務，現在に至る．原子力プラントの信頼性・安全性，新型 BWR 炉心の概念設計，省エネルギー面からのシステム評価，プラント設計自動化の研究開発に従事．工学博士．人工知能学会，電気学会，日本原子力学会，IEEE, AAAI などの会員．

田村 正義（正会員）

昭和37年生．昭和59年筑波大学第3学群情報学類卒業．昭和61年同大学大学院工学研究科修士課程修了．同年(株)日立製作所入社．現在，同社ソフトウェア工場にて，非線形最適化プログラムの開発に従事．日本OR学会会員．

野中 久典（正会員）

昭和37年生．昭和59年大阪大学工学部原子力工学科卒業．昭和61年同大学院修士課程修了．同年(株)日立製作所エネルギー研究所入社．知識工学的手法を用いるプラント建設工程計画支援システムの開発，非線形最適化システムの開発に従事．日本原子力学会会員．

重松 洋一

昭和32年3月24日生．昭和54年3月北海道大学理学部数学科卒業．昭和57年同大学工学部精密工学科卒業．昭和59年3月同大学院工学研究科精密工学専攻修士課程修了．同年4月(株)日立製作所入社．昭和63年4月北海道大学院工学研究科精密工学専攻博士後期課程入学，現在に至る．歩行機械のモデリング，軌道計画の研究に従事．精密工学会，機械学会，ロボット学会などの会員．