TR－H－077

大規模2次計画問題に対する内点法とその数値計算について －逐次線形化法の部分問題への適用を目指して－

山川 栄樹
松原 康博（奈良先端科技院大）

1994. 5. 10

ATR人間情報通信研究所

〒619-02 京都府相楽郡精華町光台2-2　☎07749-5-1011

ATR Human Information Processing Research Laboratories

2-2, Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02 Japan

Telephone: +81-7749-5-1011
Facsimile: +81-7749-5-1008

1994年5月10日

大規模2次計画問題に対する内点法とその数値計算について
――逐次線形化法の部分問題への適用を目指して――

エイ・ティ・アール人間情報通信研究所第六研究室　　指導者 山川 栄樹
実習生 松原 康博 （奈良先端科学技術大学院大学福嶋研究室）
実習期間 1993年10月1日 ～ 1994年2月28日

1　はじめに

大規模非線形計画問題に対する逐次線形化法 [1] では, 各反復で凸2次計画問題を解かなければならない. 大規模問題を取扱う場合には, 逐次線形化法の実行時間の大半が2次計画部分問題の処理に費やされるため, これを高速に解くことが要求される [5]. そこで本実習では, 計算量が理論的に多項式時間で抑えられることで最近注目を集めている内点法の一種である Potential Reduction Algorithm [3] を適用し, 実際の数値計算においてどのような工夫が必要か, またこれにより, 実際にどの程度の性能を引き出せるかについて実証的に検討を行うことにした. 内点法のアルゴリズムでは, 一般にその計算量の大半が連立一次方程式の処理で占められる. ここでは, 元の問題の疎な構造を生かしつつ, これを高速に解くための改良を試みた. また, 内点法の実装においてしばしば問題にされるステップサイズの選択方法についても数値実験により検証を行ったので, その結果を報告する.

2　凸2次計画問題と線形相補性問題

本実習で適用した内点法のアルゴリズムは, 本来線形相補性問題に対して提案されたものである. そこで, ここでは先ず, 凸2次計画問題が等価な線形相補性問題に変換できる [2] ことを示しておく.

次のような凸2次計画問題について考える.

$$\begin{aligned} &\text{目的関数：}\quad \boldsymbol{c}^T\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^T G\boldsymbol{x} \rightarrow \text{最小} \\ &\text{制約条件：}\quad A\boldsymbol{x} \geq \boldsymbol{b},\quad \boldsymbol{x} \geq 0 \end{aligned} \tag{1}$$

ここで, A は $m \times l$ 行列, $\boldsymbol{b}$ は m 次元ベクトル, $\boldsymbol{c}$ は l 次元ベクトル, G は $l \times l$ 正定値対称行列とする. 2次計画問題 (1) に対する Kuhn-Tucker 条件は, 以下のように書下すことができる.

$$\begin{cases} \boldsymbol{c} + G\boldsymbol{x} - A^T\boldsymbol{v} - \boldsymbol{u} = 0 & \\ \boldsymbol{v}^T(A\boldsymbol{x} - \boldsymbol{b}) = 0, & A\boldsymbol{x} - \boldsymbol{b} \geq 0,\quad \boldsymbol{v} \geq 0 \\ \boldsymbol{u}^T\boldsymbol{x} = 0, & \boldsymbol{x} \geq 0,\quad \boldsymbol{u} \geq 0 \end{cases} \tag{2}$$

ただし, $\boldsymbol{v}, \boldsymbol{u}$ はそれぞれ $A\boldsymbol{x} - \boldsymbol{b} \geq 0, \boldsymbol{x} \geq 0$ に対するラグランジュ乗数である. ここで, 線形制約の残差ベクトル $\boldsymbol{y}$ を導入すると, (2) の2行目の第2式より

$$\boldsymbol{y} = A\boldsymbol{x} - \boldsymbol{b} \geq 0$$

であり, また, (2) の最初の等式と最後の不等式から

$$\boldsymbol{u} = \boldsymbol{c} + G\boldsymbol{x} - A^T\boldsymbol{v} \geq 0$$

であることに注意すると, (2) の各式は次のようにまとめることができる.

$$\begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} G & -A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{v} \end{bmatrix} + \begin{bmatrix} \boldsymbol{c} \\ -\boldsymbol{b} \end{bmatrix} \tag{3}$$

$$\begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{y} \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{v} \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{y} \end{bmatrix}^T \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{v} \end{bmatrix} = 0 \tag{4}$$

ここで,

$$\hat{\boldsymbol{y}} = \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{y} \end{bmatrix}, \hat{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{v} \end{bmatrix}, M = \begin{bmatrix} G & -A^T \\ A & 0 \end{bmatrix}, \boldsymbol{q} = \begin{bmatrix} \boldsymbol{c} \\ -\boldsymbol{b} \end{bmatrix} \tag{5}$$

とおくと, (3),(4) は次のように書き直すことができる.

$$\begin{cases} \hat{\boldsymbol{y}} = M\hat{\boldsymbol{x}} + \boldsymbol{q} \\ \hat{\boldsymbol{y}}^T\hat{\boldsymbol{x}} = 0 \\ \hat{\boldsymbol{y}} \geq 0 \\ \hat{\boldsymbol{x}} \geq 0 \end{cases} \tag{6}$$

これを満たす $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ を求める問題は線形相補性問題と呼ばれる.G の正定値性より Kuhn-Tucker 条件 (2) は問題 (1) の最適性の必要十分条件であるから, 線形相補性問題 (6) と問題 (1) とは等価であることがわかる.

3 Potential Reduction Algorithm

2章の議論により, 凸2次計画問題 (1) を解くためには線形相補性問題 (6) を解けばよいことがわかったので, ここでは, 線形相補性問題に対する内点法のアルゴリズムの一種である Potential Reduction Algorithm [3] について述べる.

3.1 アルゴリズムとその計算量

問題 (6) の変数 $\hat{x}, \hat{y}$ の次元を n とする. 以下の議論では, 次のような仮定をおく.

(i) Mは $n \times n$ 半正定値行列
(ii) $S_{++} = \{(\hat{x}, \hat{y}) \in \Re^{2n} : (\hat{x}, \hat{y}) > 0 \text{ and } \hat{y} = M\hat{x} + q\} \neq \phi$

G が正定値対称なので, 仮定 (i) は満たされる. 一方, 仮定 (ii) を満たす点を求めることは一般には容易でないが, 次章で述べるような人工問題を考えることによって, この仮定を満たす点を得ることができる.Potential Reduction Algorithm は, S_{++} で定義されるポテンシャル関数の値を減少させるような点列を生成することにより, 解を得ようとする方法である. 問題 (6) に対するポテンシャル関数を次式で定義する.

$$f(\hat{x}, \hat{y}) = \sqrt{n} \log \hat{x}^T \hat{y} - \sum_{i=1}^{n} \log(n\hat{x}_i \hat{y}_i / \hat{x}^T \hat{y}) \quad , \quad \forall (\hat{x}, \hat{y}) \in S_{++} \tag{7}$$

このとき, Potential Reduction Algorithm は次のように書き下すことができる.

アルゴリズム 1

ステップ 0　　初期探索点 $(\hat{x}^1, \hat{y}^1)$ を S_{++} に選ぶ, $k := 1, \delta := 0.2$ とする.

ステップ 1

$$(\hat{x}^k, \hat{y}^k) - \theta(\Delta\hat{x}, \Delta\hat{y}) \in S_{++}$$

$$f((\hat{x}^k, \hat{y}^k) - \theta(\Delta\hat{x}, \Delta\hat{y})) \leq f(\hat{x}^k, \hat{y}^k) - \delta$$

を満たすような探索方向 $(\Delta\hat{x}, \Delta\hat{y})$ とステップサイズ θ を求め,

$$(\hat{x}^{k+1}, \hat{y}^{k+1}) := (\hat{x}^k, \hat{y}^k) - \theta(\Delta\hat{x}, \Delta\hat{y})$$

とする.

ステップ 2　　ある停止基準を満たせば終了し, そうでなければ
$k := k + 1$ としてステップ 1 へ戻る.

アルゴリズム 1 は S_{++} 上の点列を生成するので, 問題 (6) の条件

$$\hat{y} = M\hat{x} + q \, , \quad \hat{y} \geq 0 \, , \quad \hat{x} \geq 0$$

は常に満たされている. 従って停止基準としては, 例えば十分小さな ε に対して,

$$(\hat{y}^k)^T \hat{x}^k \leq \varepsilon$$

とすればよい. アルゴリズム 1 のステップ 1 の計算量は, 次のように評価できることが知られている [3].

定理 1 アルゴリズム 1 のステップ 1 は,$O(n^3)$ の計算量を必要とする. また, アルゴリズム 1 は反復回数 $O(\sqrt{n}L)$ で, $\varepsilon = 2^{-2L}$ の近似解を与える. ここに L は問題のデータ長である.

$$L = \sum_{i=1}^{n}\sum_{j=1}^{n}\lceil \log(|m_{ij}|+1)+1\rceil + \sum_{i=1}^{n}\lceil \log(|q_i|+1)+1\rceil$$

ただし, m_{ij}, q_i はそれぞれ (6) 式における $M, \boldsymbol{q}$ の成分である. また $\lceil a\rceil$ は a を下回らない最小の整数を表す.

3.2 探索方向とステップサイズの計算

ここでは, アルゴリズム 1 のステップ 1 で決定しなければならない探索方向とステップサイズについて, 問題 (1) の変数／係数の値を用いて具体的に計算する方法を示す.

3.2.1 探索方向の計算

ステップ 1 で要求される探索方向 $(\Delta\hat{\boldsymbol{x}}, \Delta\hat{\boldsymbol{y}})$ は, 次の連立 1 次方程式を解くことによって求められる.

$$\Delta\hat{\boldsymbol{y}} = M\Delta\hat{\boldsymbol{x}} \tag{8}$$

$$\hat{Y}\Delta\hat{\boldsymbol{x}} + \hat{X}\Delta\hat{\boldsymbol{y}} = Z\Delta\boldsymbol{w} \tag{9}$$

ただし

$$\Delta\boldsymbol{w} = -\frac{Z^{-1}\boldsymbol{e} - ((\sqrt{n}+n)/\|\boldsymbol{z}\|^2)\boldsymbol{z}}{\|Z^{-1}\boldsymbol{e} - ((\sqrt{n}+n)/\|\boldsymbol{z}\|^2)\boldsymbol{z}\|}$$

$$\boldsymbol{z} = (\sqrt{\hat{x}_1\hat{y}_1}, \sqrt{\hat{x}_2\hat{y}_2}, \cdots, \sqrt{\hat{x}_n\hat{y}_n})^T$$

であり, $\|\cdots\|$ はベクトルのユークリッドノルムを表す. (8) 式により, $(\hat{\boldsymbol{x}}^k, \hat{\boldsymbol{y}}^k) \in S_{++}$ ならば $(\hat{\boldsymbol{x}}^{k+1}, \hat{\boldsymbol{y}}^{k+1})$ においても, 条件$\hat{\boldsymbol{y}} = M\hat{\boldsymbol{x}} + \boldsymbol{q}$ の成立が保証される. また (9) 式により, ポテンシャル関数の値は少なくともδ だけ減少することが保証される. $\hat{X}, \hat{Y}, Z$ は $\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \boldsymbol{z}$ を対角成分とする対角行列とする.

さて, ここで

$$\Delta\hat{\boldsymbol{y}} = \begin{bmatrix} \boldsymbol{du} \\ \boldsymbol{dy} \end{bmatrix}, \Delta\hat{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{dx} \\ \boldsymbol{dv} \end{bmatrix}, \Delta\boldsymbol{w} = \begin{bmatrix} \boldsymbol{dw}_1 \\ \boldsymbol{dw}_2 \end{bmatrix}$$

とおこう. 但し, $\boldsymbol{dx}, \boldsymbol{du}, \boldsymbol{dw}_1$ は何れも l 次元ベクトル, $\boldsymbol{dy}, \boldsymbol{dv}, \boldsymbol{dw}_2$ は何れも m 次元ベクトルである. この時, (5) より 2 次計画問題 (1) に対して連立方程式 (8), (9) は次のように書下すことができる.

$$\begin{bmatrix} \boldsymbol{du} \\ \boldsymbol{dy} \end{bmatrix} = \begin{bmatrix} G & -A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{dx} \\ \boldsymbol{dv} \end{bmatrix} \tag{10}$$

$$\begin{bmatrix} U & 0 \\ 0 & Y \end{bmatrix} \begin{bmatrix} \boldsymbol{dx} \\ \boldsymbol{dv} \end{bmatrix} + \begin{bmatrix} X & 0 \\ 0 & V \end{bmatrix} \begin{bmatrix} \boldsymbol{du} \\ \boldsymbol{dy} \end{bmatrix} = \begin{bmatrix} \sqrt{UX} & 0 \\ 0 & \sqrt{YV} \end{bmatrix} \begin{bmatrix} \boldsymbol{dw}_1 \\ \boldsymbol{dw}_2 \end{bmatrix} \tag{11}$$

ここで X, V, U, Y はそれぞれ $\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{u}, \boldsymbol{y}$ を対角成分とする対角行列である. これは $\boldsymbol{dx}, \boldsymbol{du}, \boldsymbol{dy}, \boldsymbol{dv}$ についての連立一次方程式であるが, $\boldsymbol{du}, \boldsymbol{dy}, \boldsymbol{dv}$ について解くことにより, $\boldsymbol{dx}$ に関する連立一次方程式

$$(X^{-1}U + G + A^T Y^{-1} V A)\boldsymbol{dx} = X^{-1}\boldsymbol{s} + A^T Y^{-1}\boldsymbol{t} \tag{12}$$

と, $\boldsymbol{du}, \boldsymbol{dy}, \boldsymbol{dv}$ の計算式

$$\boldsymbol{dy} = A\boldsymbol{dx}$$

$$\boldsymbol{dv} = Y^{-1}(\boldsymbol{t} - V\boldsymbol{dy})$$

$$\boldsymbol{du} = G\boldsymbol{dx} - A^T\boldsymbol{dv}$$

が得られる. ここに,

$$\boldsymbol{s} = \sqrt{UX}\boldsymbol{dw}_1,$$

$$\boldsymbol{t} = \sqrt{YV}\boldsymbol{dw}_2$$

である. アルゴリズムの全反復を通して $\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{y}, \boldsymbol{v}$ の値は正に保たれるので, $X^{-1}U, Y^{-1}V$ は何れも正定値対角行列である. 従って, $\boldsymbol{dx}$ に関する連立一次方程式の係数行列 $G + X^{-1}U + A^T Y^{-1} V A$ は常に正定値対称となり, 適当な反復解法によって解を求めることができる.

3.2.2 ステップサイズ

ステップサイズ θ は, $z_{\min}$ を

$$z_{\min} = \min(z_1, z_2, \cdots, z_n) \tag{13}$$

となるように $z_{\min}$ を選んで

$$\theta = z_{\min}\tau\ , \qquad \tau \in (0, 1) \tag{14}$$

とする.

(7) 式で定義したポテンシャル関数に対してθを (14) 式で定義すると, 次のような不等式が成り立つことが知られている [3].

$$f(\hat{\boldsymbol{x}}^{k+1}, \hat{\boldsymbol{y}}^{k+1}) - f(\hat{\boldsymbol{x}}^k, \hat{\boldsymbol{y}}^k) \leq -\frac{1}{2}\sqrt{3} + \max\left\{\frac{n+\sqrt{n}}{2n}, \frac{1}{2(1-\tau)}\right\}\tau^2$$

特に $n \geq 2$ で $\tau = 0.4$ の時, $f(\hat{\boldsymbol{x}}^{k+1}, \hat{\boldsymbol{y}}^{k+1}) - f(\hat{\boldsymbol{x}}^k, \hat{\boldsymbol{y}}^k) \leq -0.2$ であることが言えるので, アルゴリズム 1 に示す条件を満たす θ は実際に存在することがわかる.

4 数値計算

ここでは, アルゴリズム 1 のステップ 0 において S_{++} 上の初期探索点を得る方法と, ステップ 1 で探索方向を求める際に必要となる連立一次方程式の解法, そしてステップサイズの決定方法について, 具体的に述べる.

4.1 初期探索点の選択

アルゴリズム1のステップ0では, 式(3)を満足する非負象限の点 $(\boldsymbol{x}^1, \boldsymbol{v}^1, \boldsymbol{u}^1, \boldsymbol{y}^1)$ を一つ見つけなければならないが, これは一般に容易なことではない. そこで人為変数 x_0 を導入し, 問題(1)の代わりに次のような凸2次計画問題を考える.

$$\begin{array}{ll} \text{目的関数：} & \boldsymbol{c}^T\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^T G\boldsymbol{x} + c_0 x_0 + \frac{1}{2}x_0^2 \rightarrow \text{最小} \\ \text{制約条件：} & A\boldsymbol{x} + x_0\boldsymbol{e} \geq \boldsymbol{b}, \\ & \boldsymbol{e}^T\boldsymbol{x} \leq b_0, \\ & \boldsymbol{x} \geq 0\ ,\ x_0 \geq 0 \end{array} \tag{15}$$

ここに, $\boldsymbol{e}$ は全ての要素が1の l 次元ベクトルである. $\boldsymbol{v}, v_0, \boldsymbol{u}, u_0$ をそれぞれこの順に問題(15)の各制約条件に対応するLagrange乗数とする. また, 非負制約を除く各制約の残差を表す変数として,

$$\boldsymbol{y} = A\boldsymbol{x} + x_0\boldsymbol{e} - \boldsymbol{b}$$

$$y_0 = b_0 - \boldsymbol{e}^T\boldsymbol{x}$$

を導入すると, 2章と同様な議論により, 問題(15)は次のような等価な線形相補性問題に変換できる.

$$\begin{bmatrix} \boldsymbol{u} \\ u_0 \\ \boldsymbol{y} \\ y_0 \end{bmatrix} = \begin{bmatrix} G & 0 & -A^T & \boldsymbol{e} \\ 0^T & 1 & -\boldsymbol{e}^T & 0 \\ A & \boldsymbol{e} & O & 0 \\ -\boldsymbol{e}^T & 0 & 0^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ x_0 \\ \boldsymbol{v} \\ v_0 \end{bmatrix} + \begin{bmatrix} \boldsymbol{c} \\ c_0 \\ -\boldsymbol{b} \\ b_0 \end{bmatrix} \tag{16}$$

$$\begin{bmatrix} \boldsymbol{u} \\ u_0 \\ \boldsymbol{y} \\ y_0 \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \boldsymbol{x} \\ x_0 \\ \boldsymbol{v} \\ v_0 \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \boldsymbol{u} \\ u_0 \\ \boldsymbol{y} \\ y_0 \end{bmatrix}^T \begin{bmatrix} \boldsymbol{x} \\ x_0 \\ \boldsymbol{v} \\ v_0 \end{bmatrix} = 0 \tag{17}$$

(3), (4) と (16),(17) とを比較することにより, (16),(17) の解 $(\boldsymbol{x}^*, x_0^*, \boldsymbol{v}^*, v_0^*, \boldsymbol{u}^*, u_0^*, \boldsymbol{y}^*, y_0^*)$ において,

$$\begin{aligned} x_0^* &= 0, \\ v_0^* &= 0, \\ u_0^* &= c_0 - \boldsymbol{e}^T\boldsymbol{v}^*, \\ y_0^* &= b_0 - \boldsymbol{e}^T\boldsymbol{x}^* \end{aligned}$$

であれば, $(\ \boldsymbol{x}^*, \boldsymbol{v}^*, \boldsymbol{u}^*, \boldsymbol{y}^*\ )$ は (3), (4) の解となることがわかる.

$u_0^* \geq 0,\ y_0^* \geq 0$ より $c_0 \geq \boldsymbol{e}^T\boldsymbol{v}^*, b_0 \geq \boldsymbol{e}^T\boldsymbol{x}^*$ でなければならないことは明らかであろう. 実際, 問題(15)の目的関数の構造より, c_0 を十分大きな正の数ととれば $x_0^* = 0$ となり, ま

た b_0 を十分大きな正の数ととれば制約条件 $\boldsymbol{e}^T\boldsymbol{x} \leq b_0$ は最適解で inactive となり, $v_0^* = 0$ となることもわかる.

このことを念頭において, (16), (17) の初期探索点を次のようにして構成する. 即ち, $\boldsymbol{x}^1 > 0, \boldsymbol{v}^1 > 0$ を任意に選び, (16) 式から $\boldsymbol{u}^1, \boldsymbol{y}^1$ が正となるように x_0^1, v_0^1 を十分大きな正の数に選ぶ. このような $(\boldsymbol{x}^1, x_0^1, \boldsymbol{v}^1, v_0^1)$ に対して, (16) 式から u_0^1, y_0^1 が正となるよう, また最適解において $x_0^* = v_0^* = 0$ が得られるように, b_0, c_0 を十分大きな正の数に設定する. 尚, 線形相補性問題 (16),(17) に対しては,

$$\begin{bmatrix} \boldsymbol{x} \\ x_0 \end{bmatrix}, \begin{bmatrix} \boldsymbol{v} \\ v_0 \end{bmatrix}, \begin{bmatrix} \boldsymbol{u} \\ u_0 \end{bmatrix}, \begin{bmatrix} \boldsymbol{y} \\ y_0 \end{bmatrix}$$

を改めて $\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{u}, \boldsymbol{y}$ とおき,

$$\begin{bmatrix} G & 0 \\ 0^T & 1 \end{bmatrix}, \begin{bmatrix} A & \boldsymbol{e} \\ -\boldsymbol{e}^T & 0 \end{bmatrix}, \begin{bmatrix} \boldsymbol{b} \\ -b_0 \end{bmatrix}, \begin{bmatrix} \boldsymbol{c} \\ c_0 \end{bmatrix}$$

を改めて $G, A, \boldsymbol{b}, \boldsymbol{c}$ とおけば, アルゴリズム 1 をそのまま適用することができる.

4.2 探索方向を求めるための連立一次方程式の解法

ここでは, このアルゴリズムの計算の大半を占める連立一次方程式 (12) の解法について考える.

連立一次方程式の解法としては, 大きく分けて, ＬＵ分解等に基づく直接解法, 係数行列の分割に基づくＳＯＲ法等の反復解法, そして凸２次関数の最小化の技法を適用した共役勾配法の３種類がある [4]. 大規模問題では, 行列 G や A が疎である場合が多い. こうした大規模問題を現実的に取扱うためには, アルゴリズムが原問題の疎な構造を反復を通して維持できることが必須条件となる. ＬＵ分解においてこのような条件を満たすためには, 行列の行や列の順番を必要に応じて入換えるオーダリングと呼ばれるテクニックが要求される. 一方, ＳＯＲ法や共役勾配法は, 原問題の係数行列や定数ベクトルに対する計算のみで構成できる為, それらの値を保持するデータ構造を工夫することにより, 原問題の疎な構造を維持したまま計算を続けることができる. 更に共役勾配法の場合, 理論的には係数行列の相異なる固有値の数と等しい反復数――即ち有限回で収束することが保証されているので, 適当な前処理を施すことにより, 実用的な時間で解くことができるものと期待できる. よってここでは, 共役勾配法を用いて連立一次方程式 (12) を解くことにする.

4.2.1 共役勾配法

以下では, 表記を簡単にするため,

$$\begin{aligned} \bar{A} &= (X^{-1}U + G + A^TY^{-1}VA), \\ \bar{\boldsymbol{b}} &= X^{-1}\boldsymbol{s} + A^TY^{-1}\boldsymbol{t}, \\ \bar{\boldsymbol{x}} &= \boldsymbol{dx} \end{aligned}$$

とおき, 解くべき連立方程式を

$$\bar{A}\bar{\boldsymbol{x}} = \bar{\boldsymbol{b}} \tag{18}$$

として議論を進める. 今, $\bar{A}$ は正定値対称行列であることに注意しよう.

連立方程式 (12) に対する共役勾配法のアルゴリズムは, 以下のように書下すことができる.

アルゴリズム 2

ステップ 0 適当な初期ベクトル $\bar{\boldsymbol{x}}^1$ を選んで

$$\boldsymbol{r}^1 := \bar{\boldsymbol{b}} - \bar{A}\bar{\boldsymbol{x}}^1$$

$$\boldsymbol{p}^1 := \boldsymbol{r}^1$$

とし, $k := 1$ とする. また. 停止基準である $\varepsilon > 0$ をセットする.

ステップ 1 以下の計算を順次行なう.

$$\alpha^k := \frac{< \boldsymbol{r}^k, \boldsymbol{r}^k >}{< \boldsymbol{p}^k, \bar{A}\boldsymbol{p}^k >}$$

$$\bar{\boldsymbol{x}}^{k+1} := \bar{\boldsymbol{x}}^k + \alpha^k \boldsymbol{p}^k$$

$$\boldsymbol{r}^{k+1} := \boldsymbol{r}^k - \alpha^k \bar{A}\boldsymbol{p}^k$$

$$\beta^k := \frac{< \boldsymbol{r}^{k+1}, \boldsymbol{r}^{k+1} >}{< \boldsymbol{r}^k, \boldsymbol{r}^k >}$$

$$\boldsymbol{p}^{k+1} := \boldsymbol{r}^{k+1} + \beta^k \boldsymbol{p}^k$$

ただし $< \cdot, \cdot >$ は内積を表す.

ステップ 2 停止基準 $\|\boldsymbol{r}^{k+1}\|_\infty \leq \varepsilon \|\bar{\boldsymbol{b}}\|_\infty$ を満たせば終了する. さもなければ, $k := k+1$ としてステップ 1 へ戻る.

4.2.2 前処理付共役勾配法

前にも少し触れたように, 理論的には, 共役勾配法の反復は $\bar{A}$ の相異なる固有値の数と等しい回数の反復で停止することが知られている. 正定値対称行列 $\bar{A}$ の逆行列が既知であれば, これを連立方程式 (18) の両辺に左から掛けることにより, 係数行列を単位行列とすることができる. 単位行列の固有値は全て 1 だから, 共役勾配法の反復は 1 回で終了する. 一般に $\bar{A}$ の逆行列を予め知ることはできないので, その近似行列を何らかの方法で生成して連立方程式の係数行列の固有値を「固める」ことにより, 共役勾配法の反復数を削減することが考えられる. これが, 前処理と呼ばれる手法である.

前処理としては, 不完全コレスキー分解を用いる方法がよく知られている [4] が, 大規模問題に対しては, 直接解法について述べたのと同様な理由で, その効率的な実現には様々な

テクニックが必要となる. ここでは, $\bar{A}$ の対角成分のみを取出した行列 $\bar{D}$ の逆行列 $\bar{D}^{-1}$ で $\bar{A}$ の逆行列を近似する, いわゆる Diagonal Scaling の手法を用いることにする. この手法は, 連立方程式の係数行列の対角要素をすべて 1 に揃えるような前処理である.

Diagonal Scaling を組込んだ共役勾配法のアルゴリズムは, 以下のようになる.

アルゴリズム 3

ステップ 0 適当な初期ベクトル $\bar{\boldsymbol{x}}^1$ を選んで

$$\boldsymbol{r}^1 := \bar{\boldsymbol{b}} - \bar{A}\bar{\boldsymbol{x}}^1$$

$$\boldsymbol{p}^1 := \bar{D}^{-1}\boldsymbol{r}^1$$

とし, $k := 1$ とする. また. 停止基準である $\varepsilon > 0$ をセットする.

ステップ 1 以下の計算を順次行なう.

$$\alpha^k := \frac{< \boldsymbol{r}^k, \bar{D}^{-1}\boldsymbol{r}^k >}{< \boldsymbol{p}^k, \bar{A}\boldsymbol{p}^k >}$$

$$\bar{\boldsymbol{x}}^{k+1} := \bar{\boldsymbol{x}}^k + \alpha^k \boldsymbol{p}^k$$

$$\boldsymbol{r}^{k+1} := \boldsymbol{r}^k - \alpha^k \bar{A}\boldsymbol{p}^k$$

$$\beta^k := \frac{< \boldsymbol{r}^{k+1}, \bar{D}^{-1}\boldsymbol{r}^{k+1} >}{< \boldsymbol{r}^k, \bar{D}^{-1}\boldsymbol{r}^k >}$$

$$\boldsymbol{p}^{k+1} := \bar{D}^{-1}\boldsymbol{r}^{k+1} + \beta^k \boldsymbol{p}^k$$

ステップ 2 停止基準 $\|\boldsymbol{r}^{k+1}\|_\infty \leq \varepsilon \|\bar{\boldsymbol{b}}\|_\infty$ を満たせば終了する. さもなければ, $k := k+1$ としてステップ 1 へ戻る.

4.3 ステップサイズの選択

式 (13), (14) によるステップサイズ θ の決定規則は, ポテンシャル関数が一反復につき少なくとも δ だけ減少することを保証し, 大域的収束性と多項式性を証明するための一つの十分条件となっている. しかしながら実際の計算においては, 探索方向が勾配の負の方向を向いている場合, 非負性が保たれる範囲でステップサイズをできるだけ大きくとることによって, より速く収束する場合も多いことが知られている.

ここでは, 非負制約の境界までの「距離」 α を

$$\alpha = \max\{\gamma | (\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) - \gamma(\Delta\hat{\boldsymbol{x}}, \Delta\hat{\boldsymbol{y}}) \geq 0\} \tag{19}$$

により求め, ステップサイズ θ を,

$$\theta = \alpha * \eta\ , \quad \eta \in (0,1) \tag{20}$$

と決定する方法について, (13),(14) 式による場合との比較を試みる. 尚, 本実習では $\eta = 0.95$ として実験を行った.

5 実験結果と考察

数値実験は, SPARCstation 2 上で行った. テスト問題は, ランダムに生成された凸2次計画問題 (1) である. 但し便宜上, 最適解と最適 Lagrange 乗数が予めわかるように, 次のような方法で各係数を生成した. 先ず, A の要素を, それぞれ指定した密度で区間 [-9, 9] の一様乱数として発生させる. G の要素は, 区間 [-9,9] で発生させた一様乱数を用いた行列とその転置行列を掛け合わせた後, 正の対角成分を足すことによって発生させる. また, 最適解 x^* と 最適 Lagrange 乗数 v^*, u^* の各要素を, それぞれ区間 [0, 9], [0, 9] の一様乱数として発生させる. 但しその際, 最適解で値が 0 となる変数, 最適解で値が 0 となる乗数 (inactive な制約) が必ず幾つか存在するようにする. そして, (2) 式に基づき $\boldsymbol{c}$ と $\boldsymbol{b}$ の値を計算する. $\boldsymbol{b}$ の決定にあたっては, 区間 [0, 9] の一様乱数で生成した最適解での残差 $\boldsymbol{y}^*$ の値を用いた. 尚, 最適解での各変数の値の生成に際しては, 狭義の相補条件が成立つように 0 となる変数を選択するようにした. ここでは, 中規模の問題に対して, ステップサイズの選択方法によるアルゴリズムの挙動の違いと, 共役勾配法における前処理の効果についての考察を加えながら, 数値実験の結果を示すことにする.

5.1 ステップサイズの選択方法と実験結果

100 変数 50 制約の凸 2 次計画問題を生成し, 前章までに述べた 2 通りステップサイズの決定方法でアルゴリズム 1 を実行した. 探索方向の決定には, アルゴリズム 2 (前処理なしの共役勾配法) を用いている. 停止基準の ε は 1.0×10^{-3} とした. 以下の表 1 は, 原論文通り (13), (14) 式に基づいてステップサイズを決定した場合, 表 2 は (19), (20) 式に基づき, より大きなステップサイズを採用した場合の結果である. 表中で, 「密度」は係数行列Ｇ及びＡの非零要素の density , 「ＣＰＵ」は停止基準が成立するまでに要した処理時間, 「ＩＴ」はアルゴリズム 1 の反復回数, 「ＣＧ」は共役勾配法の反復回数の累積値, 「ＣＰＵ／ＣＧ」はＣＰＵタイムを共役勾配法の反復回数の累積値で割ったもので, 共役勾配法の 1 反復あたりにかかるＣＰＵタイムを表している.

表１, ２からわかるように, (19), (20) 式に基づき, より大きなステップサイズを採用した場合の方が, アルゴリズム 1 の反復回数にして約 1/6, 全体のＣＰＵタイムにして 1/4 程度に改善されていることがわかる. 反復回数の減少に比べてＣＰＵタイムの減少度合が少ないことの理由としては, α を求めるために余分な手間が必要となること, 速く制約領域の境界面に近付くために, 数値的悪条件下での計算が全処理の中でより大きな割合を占めるようになること等が考えられる.

共役勾配法のアルゴリズムの主たる処理は, 行列とベクトルのかけ算である. 同じサイズの行列でも, 実際に行わなければならない乗算／加算の数は, 行列の密度に比例して大きくなる. 実験に用いたプログラムでは, 非零要素のみを保持するデータ構造を利用しているため, 共役勾配法の 1 回の反復に要する時間は, 密度が大きくなるにつれて多くなっていることが, 表の「ＣＰＵ／ＣＧ」欄からわかる.

表1：(13),(14) 式による結果

密度	CPU（秒）	IT	CG	CPU／CG（秒／反復）
1％	91.0	740	48359	1.88×10^{-3}
5％	169.6	749	65569	2.59×10^{-3}
10％	303.2	746	88167	3.44×10^{-3}
50％	5043.0	765	483875	1.04×10^{-2}
100％	4751.3	767	333484	1.42×10^{-2}

表2：(19),(20) 式による結果

密度	CPU（秒）	IT	CG	CPU／CG（秒／反復）
1％	20.0	118	8312	2.41×10^{-3}
5％	39.3	126	12978	3.03×10^{-3}
10％	59.8	126	15796	3.79×10^{-3}
50％	868.9	133	83361	1.04×10^{-2}
100％	915.7	129	63122	1.45×10^{-2}

次に，2通りのステップサイズの決定方法について $\hat{x}^T\hat{y}$ の値がどのように減少しているかを，10までは減少の仕方がわかりにくいので，同じ結果を対数グラフを用いて付録の図3，4に示す．これらの図より，相補条件の誤差は指数的に減少していることがわかる．図の横軸の尺度に注意すると，ステップサイズを大きく採ることによって，誤差の減少具合も非常に良くなることがわかる．最後に，アルゴリズム1の反復が進むにつれて，そのステップ1で実行される共役勾配法の反復数がどのように変化しているかを付録の図5，6に示す．何れの場合も探索が進むにつれて共役勾配法に要する反復数が増加していくことが認められる．

これは，探索が進むにつれて X, Y の要素のいくつかが0に近づいて，対角行列 X^{-1}, Y^{-1} の対応する要素の値が極端に大きくなるため，連立方程式(12)の係数行列 $\bar{A}$ が悪条件になるためと考えられる．更に，4．1節で導入した人為変数の値は，本来の変数の値と比べてそのオーダーが2～3桁違ってくる為，特に悪影響を及ぼしているものと考えられる．

5.2 共役勾配法と前処理

前節の実験から，ステップサイズを大きくとる方が実際上は速い収束性が得られることがわかったので，以下では(19), (20)式に基づいて θ を決定する場合について，更に実験を続けることにする．これまでの一連の結果からもわかるように，連立方程式をいかに速く解くかが内点法成功の鍵である．ここでは，共役勾配法の収束を加速するために，4.2.2節で述べた前処理の効果について実験により検証することを試みる．前節と同じ問題に対して，連立方程式を解く際にアルゴリズム3（前処理付共役勾配法）を用いた結果を表3に示す．

表３：前処理付共役勾配法を用いた結果

密度	ＣＰＵ（秒）	ＩＴ	ＣＧ	ＣＰＵ／ＣＧ（秒／反復）
１％	8.93	118	1908	4.68×10^{-3}
５％	44.22	126	14592	3.03×10^{-3}
１０％	138.68	127	38111	3.64×10^{-3}
５０％	918.99	134	90427	1.02×10^{-2}
１００％	615.93	130	42549	1.45×10^{-2}

表３を見る限り，収束性が良くなる場合と却って悪くなる場合があるように思われる．

また，問題の規模による違いについて考察するため，密度 1200 変数 100 制約，300 変数 150 制約，400 変数 200 制約，500 変数 250 制約の問題について実験した．この結果を表４に示す．

表４：前処理なし共役勾配法を用いた結果

	ＣＰＵ（秒）	ＩＴ	ＣＧ	ＣＰＵ／ＣＧ（秒／反復）
１００変数	20.0	118	8312	2.41×10^{-3}
２００変数	78.0	132	17044	4.58×10^{-3}
３００変数	430.0	200	61027	7.05×10^{-3}
４００変数	1112.0	218	104711	1.06×10^{-2}
５００変数	2086.0	248	151396	1.38×10^{-2}

表５：前処理付共役勾配法を用いた結果

	ＣＰＵ（秒）	ＩＴ	ＣＧ	ＣＰＵ／ＣＧ（秒／反復）
１００変数	8.9	118	1908	4.68×10^{-3}
２００変数	14.0	129	3752	3.73×10^{-3}
３００変数	404.0	201	114373	3.53×10^{-3}
４００変数	1973.0	222	392207	5.03×10^{-3}
５００変数	3223.0	252	477720	6.75×10^{-3}

表４，５を見る限り，200 変数までは前処理の効果が比較的顕著に出ているが，400 変数より大規模になると前処理が逆効果となっている．

200 変数，400 変数の場合について，それぞれ前処理をしない場合，前処理を行った場合の共役勾配法の反復数の変化の様子を，付録の図７，８及び図９，１０に示す．解くべき連立方程式 (18) のサイズは，凸２次計画問題の変数の数（人為変数を付け加えた場合は，１つだけ多くなる）である．従って理論的には，最大でも原問題の変数の数に等しい反復回数でこれを解くことができる筈である．200 変数の場合は，前処理を行うことによって 200 回

よりもはるかに少ない回数で収束していることがわかるが, 前処理を行わない場合には, 内点法の探索が進むにつれて実際には 200 回を超える反復数を要していることがわかる. 一方,400 変数の場合には, 前処理の実行如何にかかわらず,400 回をはるかに超える反復を必要としている. このような現象が生じる理由として, 次のような点が考えられる. 即ち, ここで行った実験では, 変数の数に対する制約条件の数の比を１／２，係数行列の密度を１％に固定したため, 連立方程式の係数行列 $\bar{A}$ は, 問題の規模が小さいほど対角行列に近くなる. 例えば,$\bar{A}$ の構成要素の一つである G について考えてみると,100 変数で密度が１％なら G は対角行列となるが,500 変数になると G は 2000 個の非対角要素を持つことになる. 今回用いた前処理である Diagonal Scaling は, 元の係数行列が対角行列ならばそれを単位行列に変換するような前処理である（この場合, 共役勾配法は１回で収束する）. しかし, 非対角要素が多くなれば, その効果は急速に低下する場合があると考えるべきもののようである.

6　大規模な問題への適用

Potential Reduction Algorithm は, 多項式時間アルゴリズムであることから, 特に大規模な問題に対して高速な解法であることが, 理論的には期待できる. そこで, 本実習では密度３％の 1000 変数 500 制約と密度 1 の適用した. 結果としては, 多大な時間を要したが解けることが分かった.

7　今後の課題

今回の実習の目的は, 最近注目を集めている内点法を用いて大規模２次計画問題がどの程度の速さで解けるかを実際に確かめることにあった. 内点法の理論的な優秀性は最悪計算量の評価に関するものであるが, 大規模問題に適用することにより, 実際的な性能においても他の方法に勝るという結果が徐々に報告されているのも事実である. しかし, 実際に取り掛かってみると, 数値的悪条件に対する対応をはじめ, 様々なテクニックが必要となることも明らかになってきた. 今回の実習では, ２次計画問題に対する他のアルゴリズムとの比較は行わなかったが, 実際の使いやすさを含めて内点法のよさを本当にきちんと評価するためには, アルゴリズムの構築の際に必要となる技法も含めて, 比較検討する必要があるかもしれない. また内点法にも, 今回とりあげた potential reduction algorithm の他に,path following と呼ばれる方法や,affine scaling algorithm といった方法が存在する. これらとの比較も今後の課題である. 一方, 内点法の計算量の大半を占める連立方程式の解法として用いた共役勾配法は, 基本的に array operation として記述することができるため, 細粒度型の並列計算機での実現が可能と考えられる. 内点法の並列化といった観点も含めて, 並列計算の可能性を探ることも一つの課題である.

参考文献

[1] M.Fukushima, K.Takazawa, S.Ohsaki and T.Iabaraki, "Successive Linearization Methods for Large-Scale Nonlinear Programmming Problems", *Japan Journal of Industrial and Applied Mathematics,* Vol. 9, No.1 ,(1992),pp.117-132.

[2] 茨木俊秀・福島雅夫 , *FORTRAN 77* 最適化プログラミング , 岩波書店,1991.

[3] M.Kojima, S.Mizuno and A.Yoshise,"An O($\sqrt{n}L$) Iteration Potential Reduction Algorithm for Linear Complementarity Problems", *Mathematical Programming,* Vol. 50, (1991),pp.331-342.

[4] 森正武 ,*FORTRAN 77* 数値計算プログラミング , 岩波書店,1992.

[5] 山川栄樹, 大規模非線形計画問題に対する逐次線形化法の適用, 第 *28* 回 *SSOR* 予稿集 ,1993,pp.47-52.

8 付録

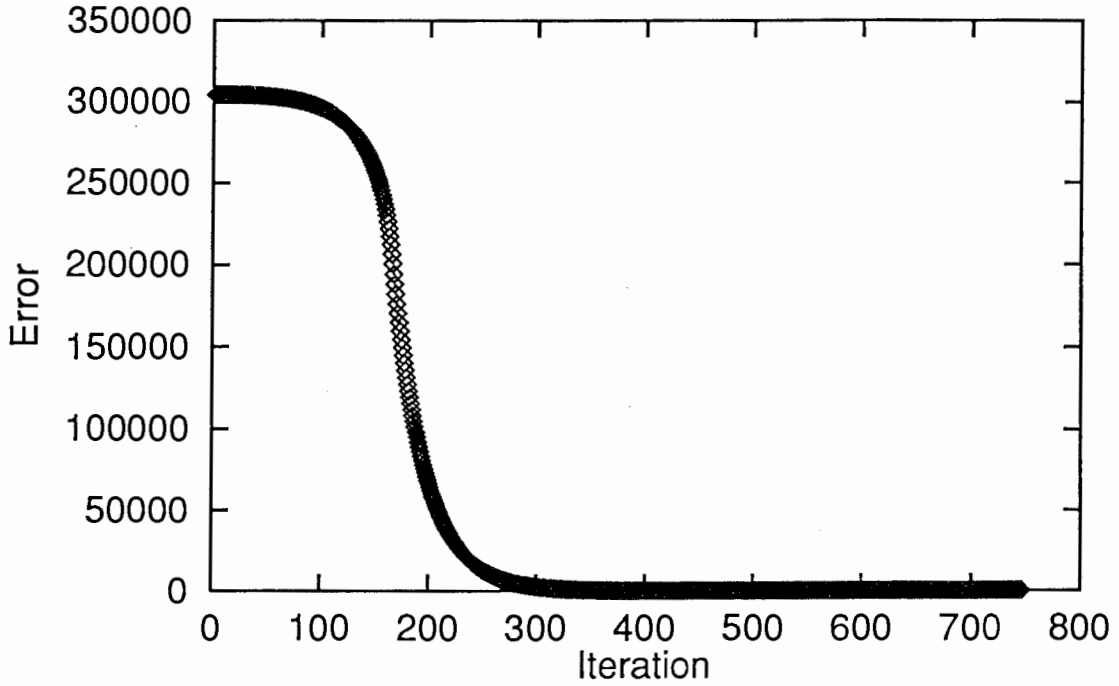

図 1: 誤差の減少（(13),(14) 式）

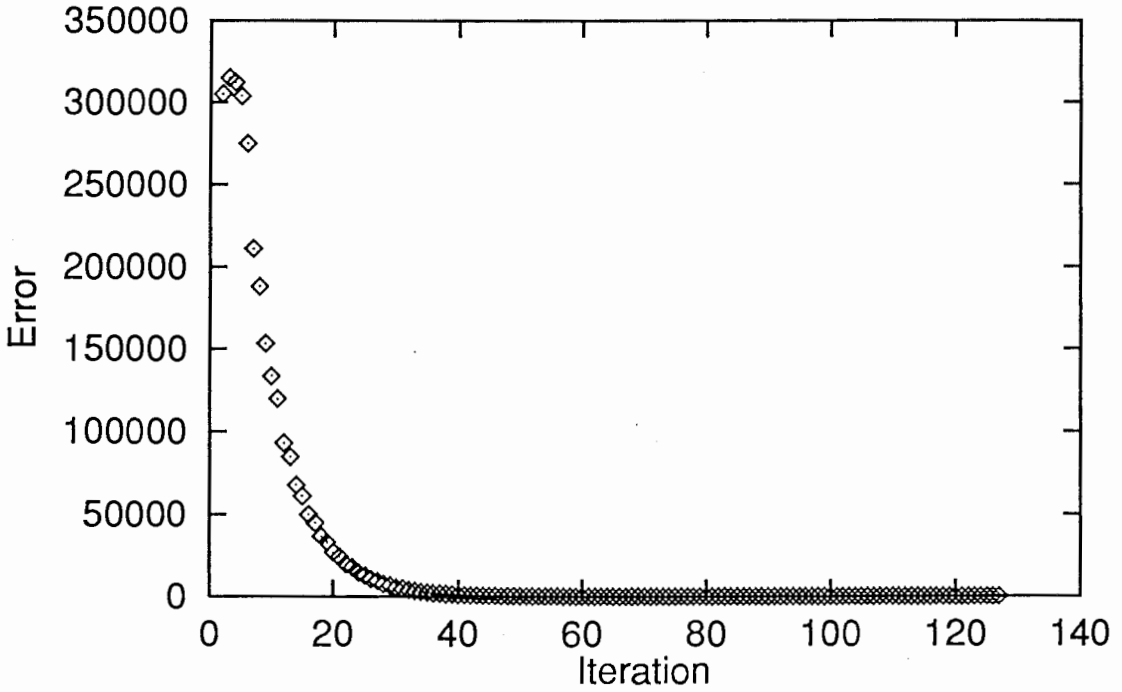

図 2: 誤差の減少（(19),(20) 式）

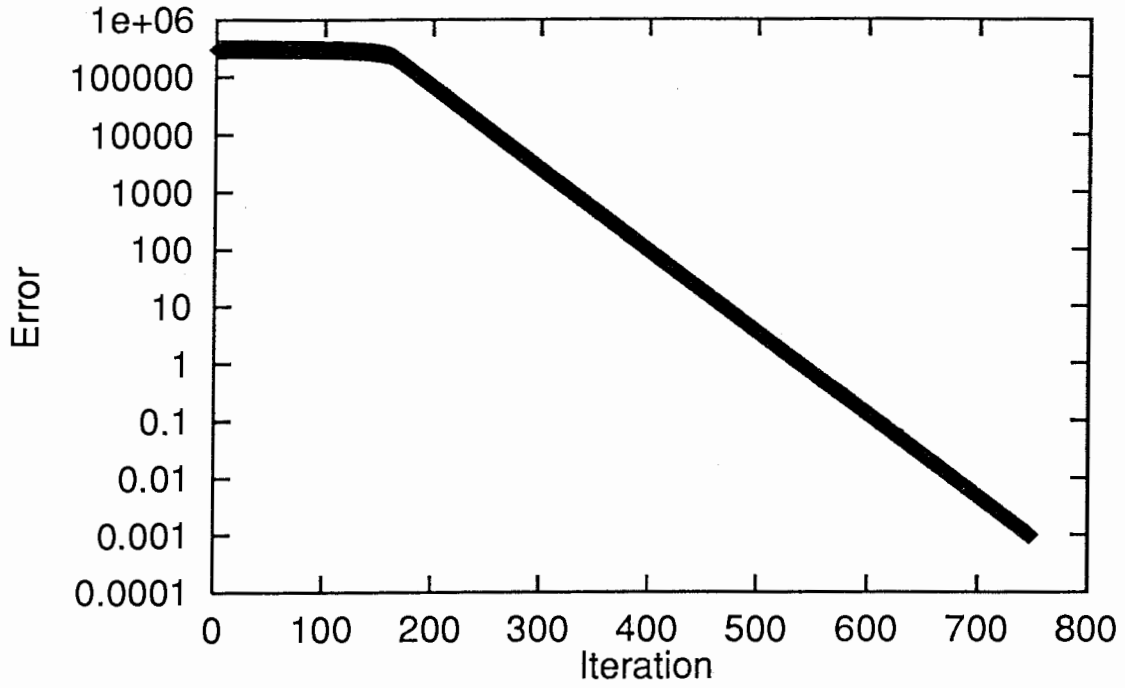

図 3: 誤差の減少（(13),(14) 式, 対数軸）

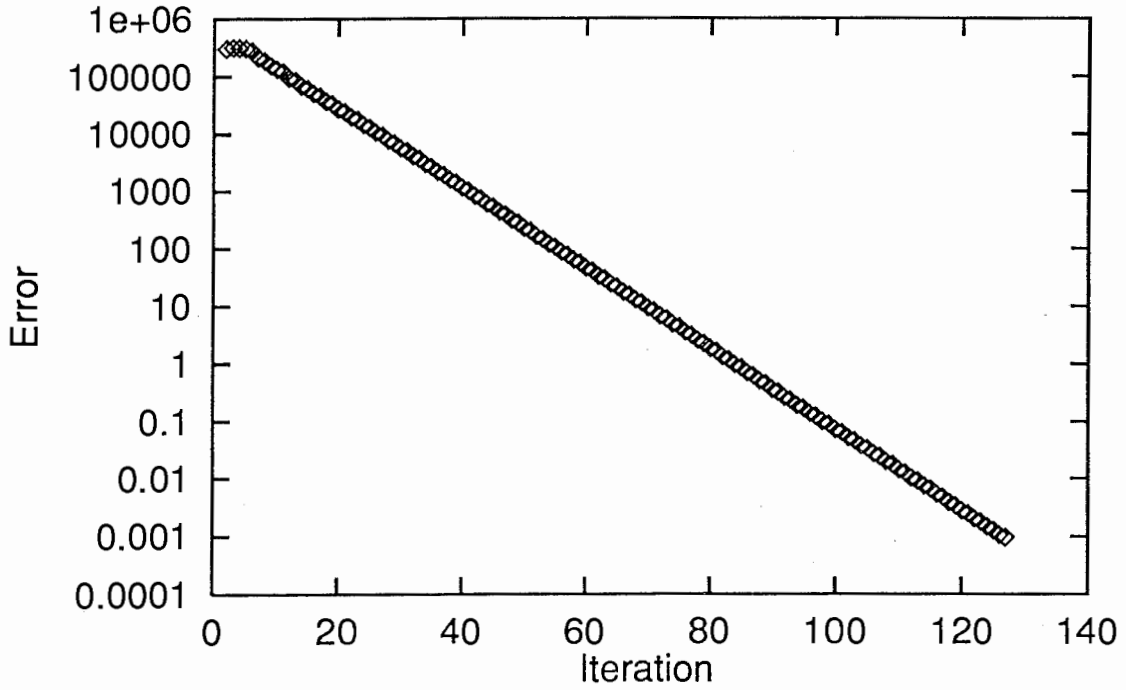

図 4: 誤差の減少（(19),(20) 式, 対数軸）

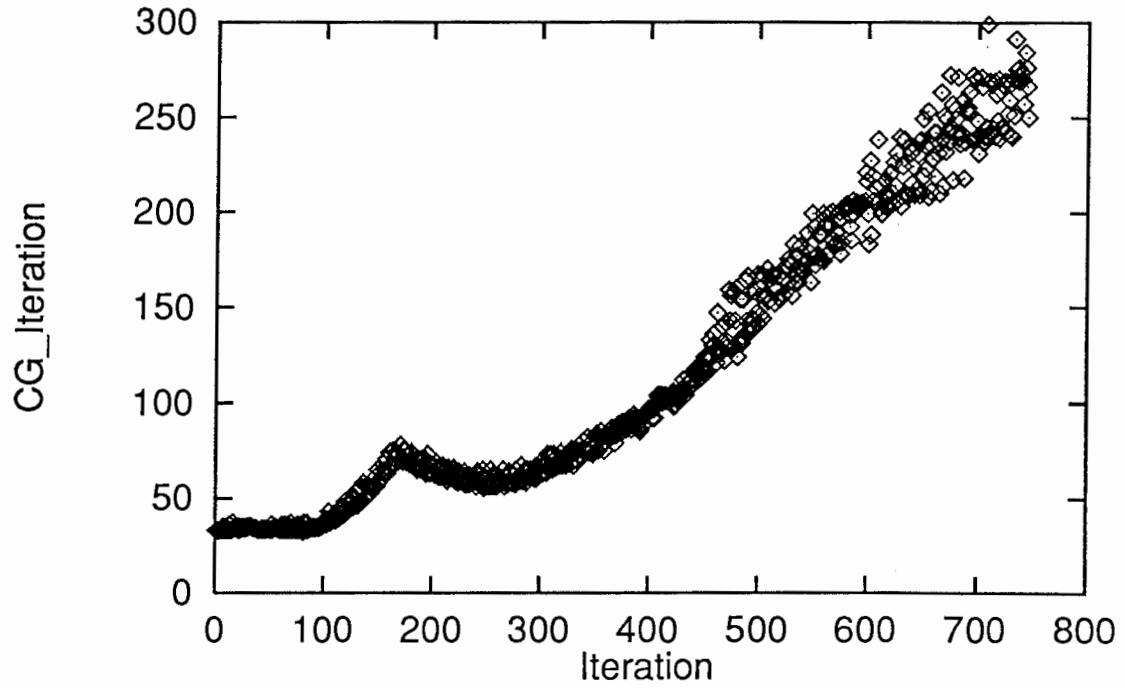

図 5: ＣＧの反復回数（(13),(14) 式）

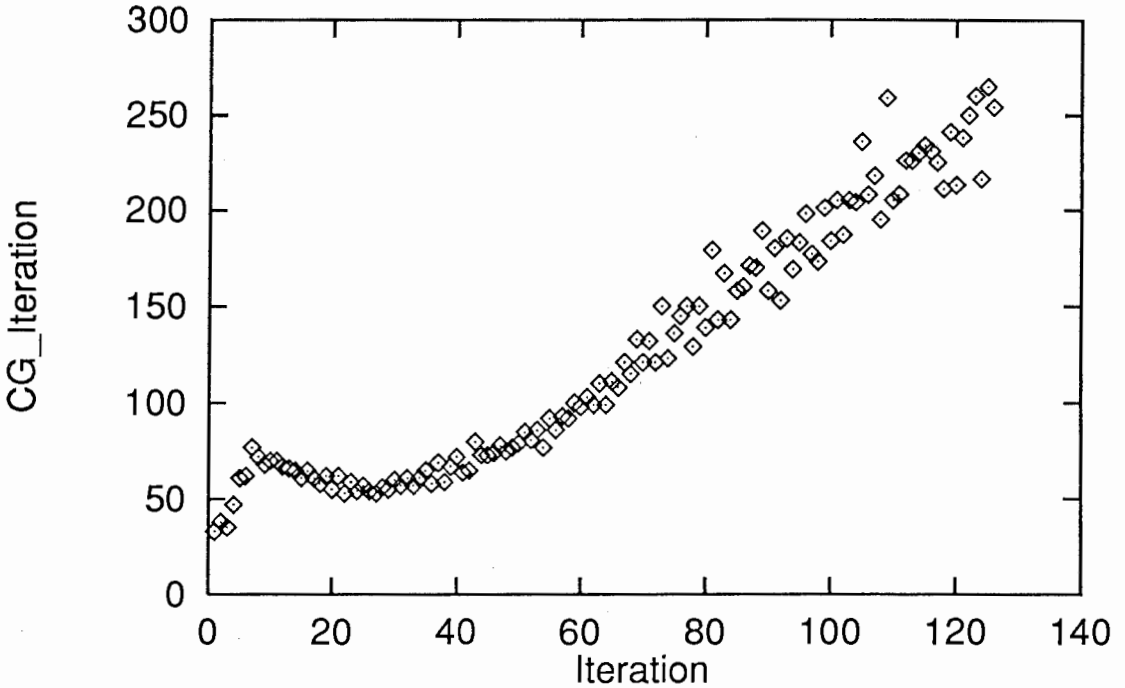

図 6: ＣＧの反復回数（(19),(20) 式）

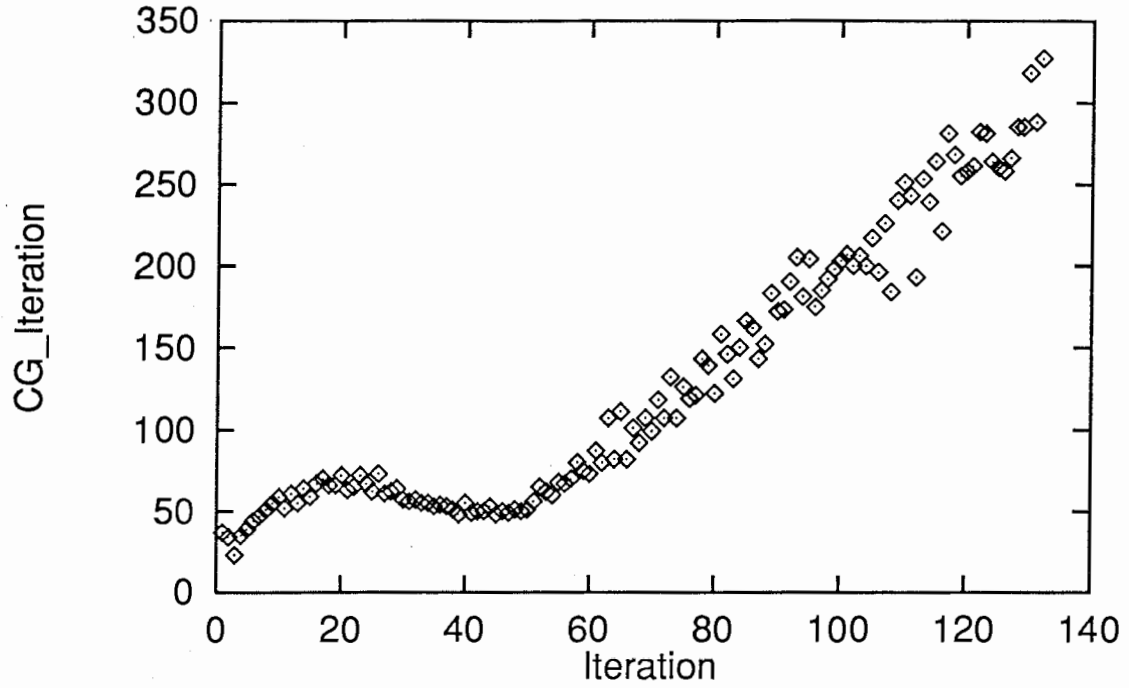

図 7: ＣＧの反復回数（前処理無し，２００変数）

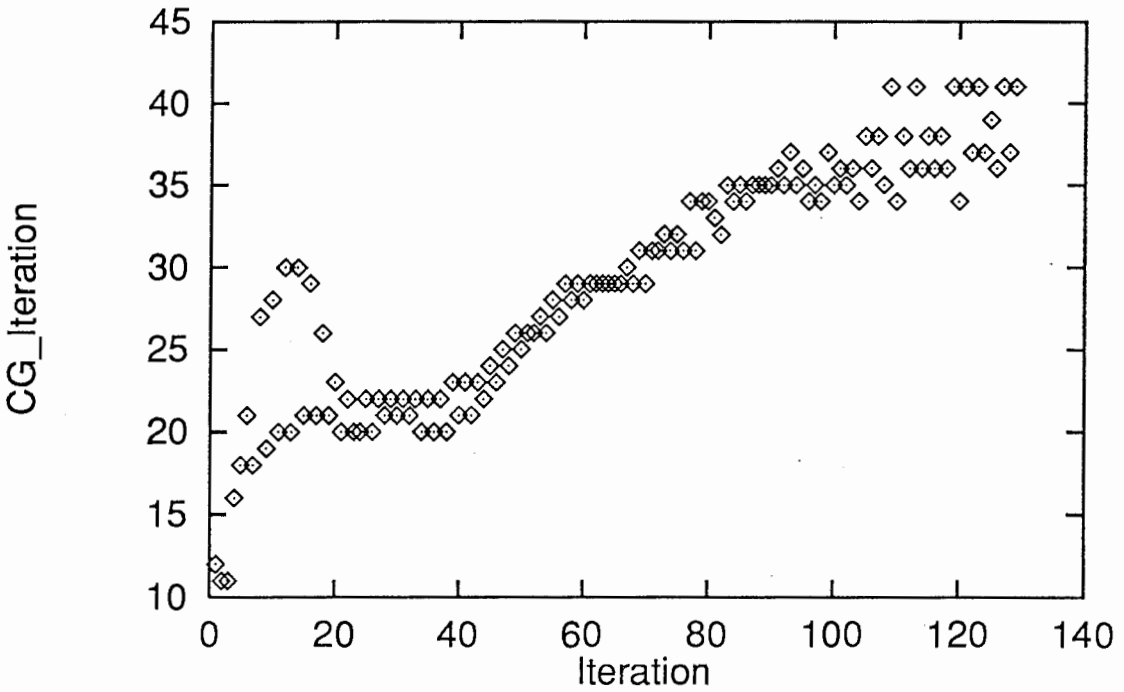

図 8: ＣＧの反復回数（前処理付，２００変数)

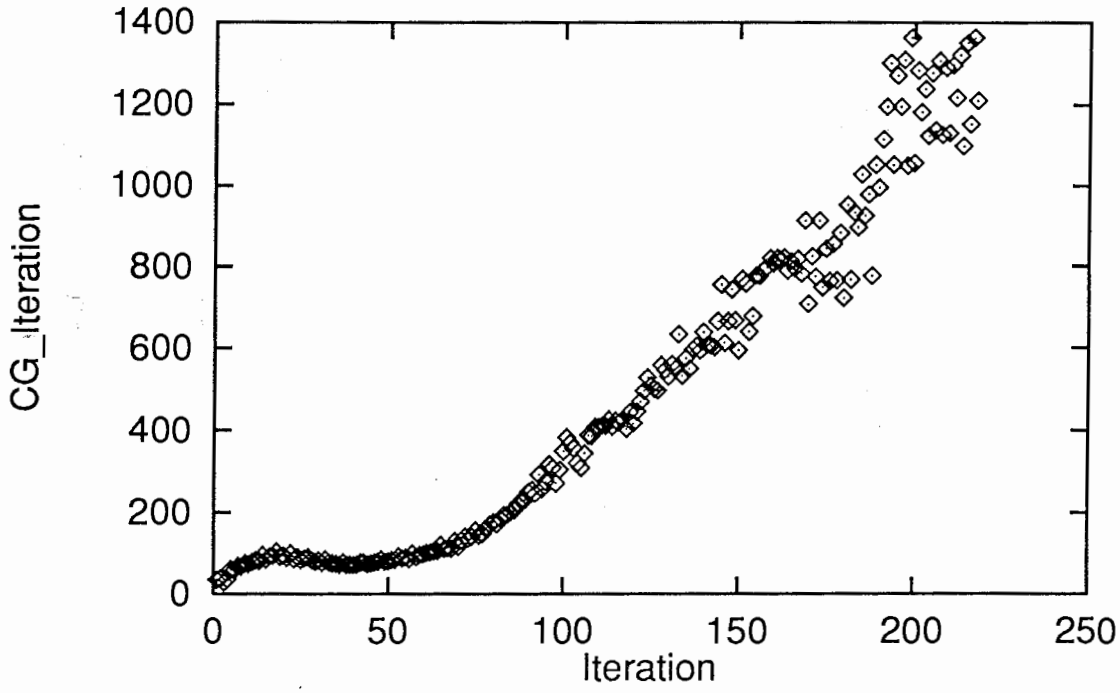

図 9: ＣＧの反復回数（前処理無し，４００変数）

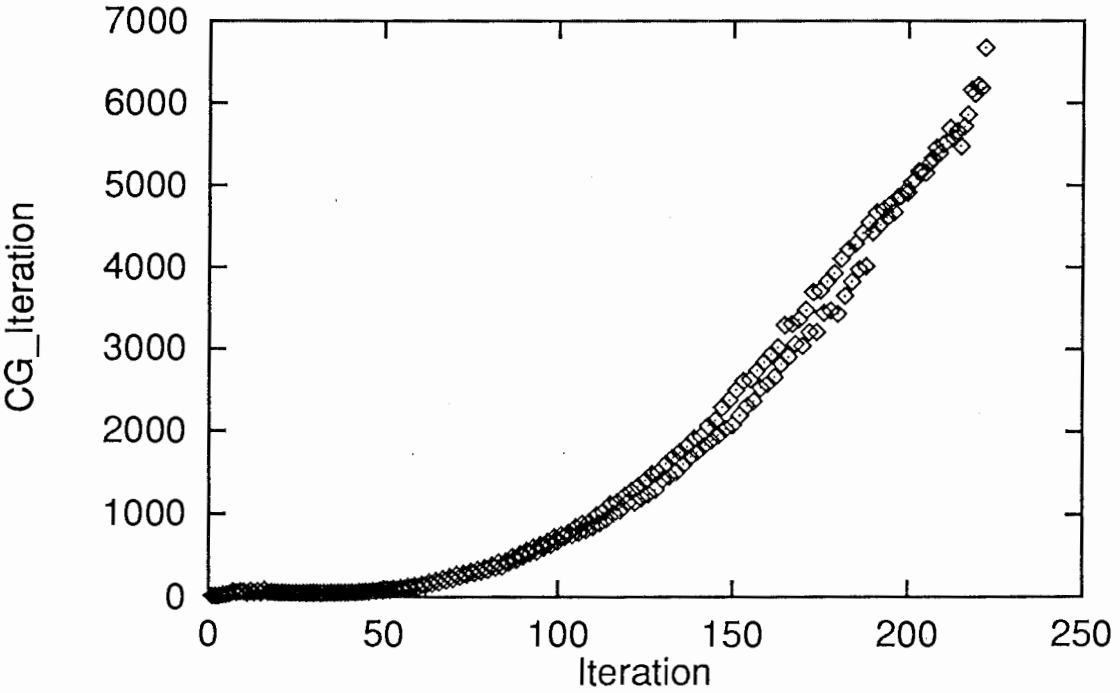

図 10: ＣＧの反復回数（前処理付，４００変数)