信頼領域を用いた大規模非線形計画問題に対する並列部分空間法

高須啓介

摘要

非線形最適化問題に対する一般的な解法では, 各反復で目的関数を線形もしくは二次のモデルに近似した部分問題を解くことで次の反復点を求める. 通常, 各反復における近似問題は元の問題と同じ次元の変数で定義される. そのため, 大規模な問題については, 各反復で大規模な近似問題を解く必要があり, 計算コストが膨大になる. そこで, 大規模な最適化問題を効率よく解く方法として並列部分空間法が提案されている. 並列部分空間法とは複数の低次元の部分空間を生成し, それぞれの空間内で並列に最適化問題を解き, それらの解の情報を利用し, 次の反復点を求める方法である.

本報告書では, 大規模な制約なし非線形最小化問題に対して並列部分空間法を適用し, 特に, 各反復に現れる部分問題に対して信頼領域法を用いるアルゴリズムを提案する. 信頼領域法は各反復で元の問題の目的関数を 2 次近似し, 信頼領域内で近似モデルの最小化問題を解くアルゴリズムである. 並列部分空間法の低次元部分問題に信頼領域法を適用することで各反復の計算量の減少が期待できる. また, いくつかのテスト問題に対して提案アルゴリズムを適用し有効性を検証する.

目 次

1 序論

本報告書では以下の制約なし最小化問題 (1) を考える.

$$\min_{x \in R^n} \ f(x) \tag{1}$$

ここで関数 $f : R^n \to R$ は 2 回連続的微分可能であり, 下に有界であるとする. 制約なし最小化問題を解く手法は, 最急降下法やニュートン法など数多く提案されている [6]. それらのアルゴリズムでは各反復で目的関数を線形もしくは二次のモデルに近似した部分問題を解き, 次の反復点を求める. 通常, 各反復における部分問題は元の問題と同じ次元の変数で定義される. そのため, 大規模な問題については各反復で大規模な近似問題を解く必要があり, 計算コストが大きくなる.

そこで, 大規模な問題を解く方法として, 部分空間法が提案されている [4]. 部分空間法は次の反復点を探索する空間を低次元の部分空間に限定する方法である. 各反復の部分問題が低次元になるため, 一回の反復における計算量を減らすことができる. 部分空間法に関して, Fukushima [2] は, 並列変数変換法 (Parallel Variable Transformation algorithm, 以下 PVT 法) を提案している. PVT 法は各反復で複数の部分空間を生成し, それぞれの空間内で並列に部分問題を解き, それらの情報を用いて次の反復点を求めるアルゴリズムである. PVT 法については, 各反復で現れる低次元部分問題に対して一般的な降下法を適用することで大域的収束性が保証されることが示されている.

また, 制約なし最小化問題を解くアルゴリズムの代表的な手法として信頼領域法が知られている. 信頼領域法は各反復において目的関数の 2 次モデルが妥当であると思われる領域 (信頼領域) 内でその 2 次モデルを最小化するステップを求める方法である. つまり, 各反復で制約つき最小化問題を解く必要がある. そのため, 元の問題が大規模な問題のときは, 各反復で大規模な制約つき最小化問題を解く必要があり, 計算コストがかかる. 大規模な問題に対して, Yuan [4] により部分空間法と信頼領域法を組み合わせたアルゴリズムが提案されている. その手法では反復ごとに低次元部分空間を選ぶが, 提案された部分空間の選び方は 2 次モデル関数について一定以上の減少を保証するものである. 部分空間の選び方の点から見ると PVT 法は複数の部分空間の中から目的関数値が最も減少するような空間を選ぶことができるため, 特に並列処理が可能な場合には有効であると期待できる.

本報告書では大規模な問題 (1) を PVT 法を用いて解くアルゴリズムを

提案する. 提案アルゴリズムでは, 低次元の部分問題に対して信頼領域法を用いる. また, いくつかのテスト問題に対して, 並列プロセッサ上でアルゴリズムを実装して数値実験を行い, アルゴリズムの有効性を検証する.

本報告書の構成は以下の通りである. 2 節では PVT 法の考え方とアルゴリズムについて説明し, その性質を紹介する. 3 節では信頼領域法の特徴や部分問題の解き方について述べる. 4 節では PVT 法と信頼領域法を用いたアルゴリズムを提案する. 5 節ではいくつかのテスト問題に対して数値実験を行い, 提案アルゴリズムの有効性を検証する. 最後に 6 節で結論を述べる.

2 PVT 法

この節では PVT 法 [2] を紹介する. PVT 法は複数の低次元部分空間において並列に探索を行い, それらの情報を元に次の反復点を求める方法である. PVT 法は並列アルゴリズムを設計するための一般的な枠組みの一つになっている. 以下では, まず PVT 法の考え方とそのアルゴリズムを説明し, 次に PVT 法の大域的収束性について述べる.

2.1 PVT 法とは

以下では制約なし最小化問題 (1) について考える. いま, k 回目の反復点 $x^{(k)}$ と p 個の部分空間 $S_l^{(k)}$ $(l = 1, \ldots, p)$ が与えられているとする. $S_l^{(k)}$ の次元は m_l で, その基底を $q_{l1}, q_{l2}, \ldots, q_{lm_l}$ とする. このとき, 部分空間 $S_l^{(k)}$ 上で目的関数 f を最小にする点を見つけるには次の最小化問題を解けばよい.

$$\min_{y \in R^{m_l}} \phi_l^{(k)}(y) \equiv f(x^{(k)} + Q_l^{(k)} y) \tag{2}$$

上式において, $Q_l^{(k)}$ は $S_l^{(k)}$ の基底を並べた $n \times m_l$ 行列であり, $Q_l^{(k)} = [q_{l1}, q_{l2}, \ldots, q_{lm_l}]$ で定義される. この部分問題の変数 y は $m_l (< n)$ 次元のベクトルである. PVT 法のアルゴリズムは以下のようになる. なお, p は並列プロセッサ数に対応する.

アルゴリズム 1(PVT 法)

Step 1(初期化). p 個の正の整数 $m_l\,(l=1,\ldots,p)$ を $m_1+\cdots+m_p \geq n$ となるように選ぶ. 初期点 $x^{(0)} \in R^n$ を選び, $k=0$ とする.

Step 2(並列). 各 l について行列 $Q_l^{(k)} \in R^{n\times m_l}$ を選び, 次の最小化問題の近似解 $y_l^{(k)} \in R^{m_l}$ を並列に求める.

$$\min_{y_l \in R^{m_l}} \ \phi_l^{(k)}(y_l) \equiv f(Q_l^{(k)} y_l + x^{(k)}) \tag{3}$$

全ての l について $\nabla\phi_l^{(k)}(0)=0$ ならば終了する.

Step 3(同期). 行列 $B^{(k)} \in R^{n\times(p+1)}$ を次のように定める

$$B^{(k)} = [x^{(k)}, Q_1^{(k)} y_1^{(k)} + x^{(k)}, \ldots, Q_p^{(k)} y_p^{(k)} + x^{(k)}] \tag{4}$$

次の最小化問題の近似解 $z^{(k)} \in R^{p+1}$ を求める.

$$\min_{z \in R^{p+1}} \ \psi^{(k)}(z) \equiv f(B^{(k)} z) \tag{5}$$

$x^{(k+1)} = B^{(k)} z^{(k)},\ k = k+1$ として Step2 へ戻る.

アルゴリズム 1 において, Step 1 で選ぶ $m_1,\ldots,m_p$ は各部分空間 $S_l^{(k)}\,(l=1,\ldots,p)$ の次元に対応する. Step 2 では各 l について部分問題 (3) を解き, p 個の近似解を得る. Step 3 では, それらの近似解と現在の点 $x^{(k)}$ の線形結合を考え, 目的関数 f をより減少させるような点を見つける.

2.2 PVT 法の大域的収束性

PVT 法は適当な仮定のもとで大域的収束性が保証されている. また, その仮定は Step 2 と Step 3 の部分問題 (3) と (5) において, 厳密な最適解を要求しない. 部分問題 (3) については, 次の式を満たす $y_l^{(k)}$ を求めればよい.

$$\phi_l^{(k)}(y_l^{(k)}) \leq \phi_l^{(k)}(0) - \sigma_l^{(k)}(\|\nabla\phi_l^{(k)}(0)\|) \tag{6}$$

(6) 式において, 関数 $\sigma_l^{(k)}:\ [0,\infty) \to [0,\infty)$ は $\phi_l^{(k)}$ に関する強制関数であり, 任意の非負数列 $\{t_k\} \subset [0,\infty)$ に対して, 以下が成り立つ.

$$\sigma(t_k) \to 0 \Longrightarrow t_k \to 0$$

関数 $\phi_l^{(k)}$ に対して $y_l = 0$ を初期点として適当な降下法を 1 反復適用すれば, (6) 式を満たす $y_l^{(k)}$ が求められることが分かっている [2].
問題 (5) については, 次の式を満たす $z^{(k)}$ を求めればよい.

$$\psi^{(k)}(z^{(k)}) \leq \min_{1 \leq l \leq p} \phi_l^{(k)}(y_l^{(k)}) \tag{7}$$

$\phi_l^{(k)}(y_l^{(k)})$ を最小にする l を l' とすると (7) 式は $z^{(k)} = e_{l'}$ とすれば満たされる. このことは次の反復点 $x^{(k+1)}$ を $x^{(k+1)} = Q_{l'}^{(k)} y_{l'}^{(k)} + x^{(k)}$ とすることにあたる. なお, $e_i \in R^n$ $(1 \leq i \leq n)$ は第 i 要素が 1 で他の要素が 0 の単位ベクトルを表す.

以下に PVT 法の大域的収束性を保証するための仮定を記す.
(仮定 1) 行列 $Q_1^{(k)}, \ldots, Q_p^{(k)}$ を横に並べた $n \times (m_1 + \cdots + m_p)$ 行列を $\hat{Q}^{(k)}$ とする. このとき, 任意の k と任意の $x \in R^n$ について次式を満たす定数 $\beta > 0$ が存在する.

$$\|\hat{Q}^{(k)T} x\| \geq \beta \|x\|$$

仮定 1 は部分空間 $S_1^{(k)}, \ldots, S_p^{(k)}$ によって n 次元空間全体が張られる, つまり, 任意の k について

$$S_1^{(k)} + \cdots + S_p^{(k)} = R^n$$

が成り立つことと同値である
(仮定 2) 各 l に対して, 式 (6) を満たす強制関数列 $\left\{\sigma_l^{(k)} | k = 0, 1, \ldots\right\}$ について次のことが成り立つ. 任意の $\epsilon > 0$ に対して, 任意の k に対して次の式を満たす, k に依存しないある $\delta > 0$ が存在する.

$$\sigma_l^{(k)}(t) < \delta \Longrightarrow t < \epsilon$$

仮定 2 については, より一般的な仮定で仮定 2 を満たす強制関数が存在することが証明されている [2].
PVT 法の大域的収束性について以下の定理が証明されている [2].
(定理 1) PVT 法の Step 2 と Step 3 において, それぞれ (6) 式と (7) 式を満足するような $y_l^{(k)}$ $(l = 1, \ldots, p)$ と $z^{(k)}$ が求められ, 仮定 1 と仮定 2 が満たされるとする. そのとき PVT 法で生成される点列 $\left\{x^{(k)}\right\}$ は

$$\lim_{k \to \infty} \nabla f(x^{(k)}) = 0$$

を満たし, $\left\{x^{(k)}\right\}$ の任意の集積点は問題 (1) の停留点となる.

以上の議論より, PVT 法の大域的収束性を保証する条件にはかなり柔軟性があることが分かる. 実際, PVT 法は並列アルゴリズムの一般的な枠組みを与えるものであり, 多様な実行方法を考えることができる.

3 信頼領域法

信頼領域法 [1] [5] は最急降下法や準ニュートン法等の直線探索法に比べ適用範囲が広く, 数値的に頑健であることが知られている. この節では, 信頼領域法の考え方と部分問題の解き方, その性質について述べる. なお, 以下における信頼領域法の説明は問題 (1) を対象としている.

3.1 信頼領域法の考え方

ニュートン法は目的関数 f のヘッセ行列 $\nabla^2 f(x)$ が常に正定値である場合を除いて, 降下方向を生成するとは限らないため, 直線探索法を直接適用するのは適切とはいえない. そこで, ニュートン法に大域的収束性をもたせるため, 信頼領域法がしばしば用いられる. 信頼領域法は, 各反復において目的関数の 2 次モデルが妥当であると思われる領域 (信頼領域) 内で 2 次モデルを最小化する. つまり各反復で次の部分問題 (8) を解く.

$$\begin{array}{ll} \min & F^{(k)}(d) \equiv f(x^{(k)}) + \nabla f(x^{(k)})^T d + \frac{1}{2} d^T \nabla^2 f(x^{(k)}) d \\ \text{s.t.} & \|d\| \leq \Delta^{(k)} \end{array} \tag{8}$$

ここで $\|d\|=\sqrt{d^T d}$ であり, $\Delta^{(k)}$ は信頼半径と呼ばれる定数である. 信頼領域は有界閉集合であるため,$\nabla^2 f(x^{(k)})$ が正定値でなくとも, 問題 (8) は最小解 $d^{(k)}$ をもつ. 信頼領域法では, モデル関数の減少量

$$\Delta F^{(k)} \equiv F^{(k)}(0) - F^{(k)}(d^{(k)})$$

と目的関数の減少量

$$\Delta f^{(k)} \equiv f(x^{(k)}) - f(x^{(k)} + d^{(k)})$$

を比較し, 反復点を更新するかどうかと信頼半径の増減を決める. 信頼領域法のアルゴリズムは以下の通りである.

アルゴリズム 2(信頼領域法)

Step 1. 定数 $0 < \mu_1 < \mu_2 < 1$, $0 < \gamma_1 < 1 < \gamma_2$ を選ぶ.
$x^{(0)} \in R^n$, $\Delta^{(0)} > 0$ を選び, $k = 0$ とする.
Step 2. 終了条件を満たせば停止する.
部分問題 (8) を解き, $d^{(k)}$ を求める.
Step 3. $r^{(k)} = \frac{\Delta f^{(k)}}{\Delta F^{(k)}}$ を計算する.
$r^{(k)} \geq \mu_1$ ならば, $x^{(k+1)} = x^{(k)} + d^{(k)}$ とする.
$r^{(k)} < \mu_1$ ならば, $x^{(k+1)} = x^{(k)}$ とする.
Step 4. $r^{(k)} \geq \mu_2$ ならば, $\Delta^{(k+1)} = \max\left\{\gamma_2 \|d^{(k)}\|, \Delta^{(k)}\right\}$ とする.
$\mu_1 \leq r^{(k)} < \mu_2$ ならば, $\Delta^{(k+1)} = \Delta^{(k)}$ とする.
$r^{(k)} < \mu_1$ ならば, $\Delta^{(k+1)} = \gamma_1 \Delta^{(k)}$ とする.
$k = k + 1$ として Step2 へ戻る.

信頼領域法の考え方を説明する. 部分問題 (8) の解 $d^{(k)}$ を採用したときに目的関数値が十分減少する ($r^{(k)} \geq \mu_1$) ならば, モデル関数が f の良い近似であると判断し, $x^{(k+1)} = x^{(k)} + d^{(k)}$ とする. 目的関数値の減少量が少ないかもしくは逆に増加する ($r^{(k)} < \mu_1$) ならば, モデル関数が f の良い近似ではないと判断し, 反復点は $x^{(k)}$ から移動しない. また, そのようなときは Step 4 で信頼半径を小さくし, 次の反復では目的関数値が減少する可能性を増やす. 一方, 目的関数とモデル関数のずれが小さいとき ($r^{(k)} \geq \mu_2$) は次の反復点でのステップが大きくなるように信頼半径を増加させる.

3.2 部分問題の解き方

信頼領域法のアルゴリズムの Step 2 において部分問題 (8) を解く方法について述べる. $d^{(k)}$ が問題 (8) の大域的最適解であるための必要十分条件は次の三つの条件を満たす $\lambda \geq 0$ が存在することである [1] [5].

$$(\nabla^2 f(x^{(k)}) + \lambda I) d^{(k)} = -\nabla f(x^{(k)}) \tag{9}$$

$$\|d^{(k)}\| \leq \Delta^{(k)}, \quad \lambda(\|d^{(k)}\| - \Delta^{(k)}) = 0 \tag{10}$$

$$\nabla^2 f(x^{(k)}) + \lambda I \text{ は半正定値} \tag{11}$$

(9)-(11) 式を満たす $d^{(k)}$ と $\lambda \geq 0$ を見つける方法として以下の方法がある. 固定した λ に対する次の連立一次方程式の解を $d(\lambda)$ と表す.

$$(\nabla^2 f(x^{(k)}) + \lambda I) d = -\nabla f(x^{(k)}) \tag{12}$$

まず $\lambda = 0$ として, (9)-(11) 式が満たされるかどうか, すなわち, $\nabla^2 f(x^{(k)})$ が半正定値であり, かつ $\|d(0)\| \leq \Delta^{(k)}$ が成り立つかを調べる. $\nabla^2 f(x^{(k)})$ の半正定値性は (12) を計算するのに, コレスキー分解を用いれば計算過程で判定することができる. (9)-(11) 式が成り立てば, $d^{(k)} = d(0)$ は問題 (8) の大域的最適解である. 成り立たなければ, 以下の手順を実行する.

まず, (11) 式を満たす $\lambda > 0$ を見つける. (11) 式は λ が十分大きいときには満たされる. 次に (11) 式が成り立つ $\lambda > 0$ の範囲で

$$\|d(\lambda)\| = \Delta^{(k)} \tag{13}$$

を満たすものを探す. このような λ を計算するには (13) 式を λ に関する非線形方程式とみなして解けばよい. しかし, λ が $\nabla^2 f(x^{(k)})$ の固有値に -1 をかけたものに等しいとき, $\|d(\lambda)\|$ は無限大に発散するので, (13) 式をそのまま解くと数値的に不安定になる. よって, 次の方程式を適当な反復法を用いて解く.

$$\frac{1}{\|d(\lambda)\|} - \frac{1}{\Delta^{(k)}} = 0 \tag{14}$$

(14) 式を解くのに, $\nabla^2 f(x^{(k)}) + \lambda I$ が半正定値かつ (14) の左辺が負である λ を初期点としてニュートン法を用いれば, (13) 式と (11) 式を満たす λ が求まることが証明されている [1].

3.3 信頼領域法の性質

信頼領域法は次に述べるような理論的に優れた収束性を持つことが知られている [1] [5].

(大域的収束性)

信頼領域法によって生成される点列 $\left\{x^{(k)}\right\}$ が有界のとき, 最適性の 2 次の必要条件である次の式を満たす $\left\{x^{(k)}\right\}$ の集積点 x^* が存在する.

$$\begin{aligned} \nabla f(x^*) &= 0 \\ y^T \nabla^2 f(x^*) y &\geq 0 \quad (y \in R^n) \end{aligned}$$

(2 次収束性)

さらに, x^* において, 最適性の 2 次の十分条件である次の式が成り立つとき, $\left\{x^{(k)}\right\}$ は x^* に 2 次収束する.

$$y^T \nabla^2 f(x^*) y > 0 \quad (y \in R^n,\ y \neq 0)$$

4　提案アルゴリズム

この節では問題 (1) に対する PVT 法において, 各反復の部分問題を信頼領域法を用いて解くアルゴリズムを提案する. いま, 2 節と同様に, k 回目の反復点 $x^{(k)}$ と部分空間 $S_l^{(k)}$ $(l = 1, \ldots, p)$, および, それらの基底を並べた行列 $Q_l^{(k)}$ $(l = 1, \ldots, p)$ が与えられているとする. なお, p はプロセッサ数である. PVT 法において (3) 式で定義される関数 $\phi_l^{(k)}(y_l)$ に対して, 2 次モデルを作成し, 次の信頼領域法の部分問題を解き, p 個の解 $d_l^{(k)}$ $(l = 1, \ldots, p)$ を得る.

$$\left.\begin{array}{ll} \min & \Phi_l^{(k)}(d) \equiv \phi_l(0) + \nabla\phi_l(0)^T d + \frac{1}{2} d^T \nabla^2 \phi_l(0) d \\ \text{s.t.} & \|d\| \leq \Delta^{(k)} \end{array}\right\} \tag{15}$$

$d_l^{(k)}$ $(l = 1, \ldots, p)$ の中で $\phi_l^{(k)}(d_l^{(k)})$ が最小となるものを選び, その $d^{(k)}$ に対し, 信頼領域法と同様に $\phi_l^{(k)}$ の減少量とモデル関数 $\Phi_l^{(k)}$ の減少量を比較し, 次の反復点と信頼半径を決定する. 提案するアルゴリズムでは, 部分空間 $S^{(k)}$ として, p 個の $\frac{n}{p}$ 次元部分空間 $S_1^{(k)}, S_2^{(k)}, \ldots, S_p^{(k)}$ を考える. ただし, p は n の約数とする. $S_l^{(k)}(l = 1, \ldots, p)$ の基底を並べた行列 $Q_l^{(k)}$ については, 互いに一次独立な n 本のベクトルを $q_1^{(k)}, q_2^{(k)}, \ldots, q_n^{(k)}$ とし, $Q_l^{(k)} = [q_{(l-1)n/p+1}^{(k)}, \ldots, q_{ln/p}^{(k)}]$ と定義する. このとき, $S_1^{(k)} + S_2^{(k)} + \cdots + S_p^{(k)} = R^n$ であり, PVT 法の大域的収束性の仮定 1 を満たす. 本報告書の数値実験では部分空間として, 座標軸方向のベクトルで張られる空間を用いる. つまり, $S_l^{(k)}$ の基底は $e_{(l-1)n/p+1}, \ldots, e_{ln/p}$ であり, $Q_l^{(k)}$ $(l = 1, \ldots .p)$ は次で定義される.

$$Q_l^{(k)} = [e_{(l-1)n/p+1}, \ldots, e_{ln/p}]$$

提案するアルゴリズムは以下の通りである.

アルゴリズム 3(提案アルゴリズム)

Step 1. 定数 $0 < \mu_1 < \mu_2 < 1$, $0 < \gamma_1 < 1 < \gamma_2$ を選ぶ.
$x^{(0)} \in R^n$, $\Delta^{(0)} > 0$ を選び, $k = 0$ とする.

Step 2. 終了条件を満たせば停止する.
各 $l \in \{1, \ldots, p\}$ について

$$\phi_l^{(k)}(y_l) \equiv f(x^{(k)} + Q_l^{(k)} y_l) \quad (l = 1, \ldots, p)$$

として, 部分問題 (15) を並列に解く.

Step 3. 得られた p 個の解 $d_l^{(k)}$ $(l = 1, \ldots, p)$ に対して,

$$d_{l'}^{(k)} \equiv \arg\min_{1 \le l \le p} \phi(d_l^{(k)})$$

を求める.

Step 4.

$$r^{(k)} = \frac{\phi_{l'}^{(k)}(0) - \phi_{l'}^{(k)}(d_{l'}^{(k)})}{\Phi_{l'}^{(k)}(0) - \Phi_{l'}^{(k)}(d_{l'}^{(k)})}$$

を計算し,
$r^{(k)} \ge \mu_1$ ならば, $x^{(k+1)} = x^{(k)} + Q_{l'}^{(k)} d_{l'}^{(k)}$ とする.
$r^{(k)} < \mu_1$ ならば, $x^{(k+1)} = x^{(k)}$ とする.

Step 5. $r^{(k)} \ge \mu_2$ ならば, $\Delta^{(k+1)} = \max\left\{\gamma_2 \|d^{(k)}\|, \Delta^{(k)}\right\}$ とする.
$\mu_1 \le r^{(k)} < \mu_2$ ならば, $\Delta^{(k+1)} = \Delta^{(k)}$ とする.
$r^{(k)} < \mu_1$ ならば, $\Delta^{(k+1)} = \gamma_1 \Delta^{(k)}$ とする.
$k = k + 1$ として Step2 へ戻る.

5 数値実験

この節では, 前節で提案したアルゴリズムをいくつかのテスト問題に適用した結果を記す. 実験は CPU が 1150MHz × 16, メモリが 32GB の京都大学学術情報メディアセンターのスーパーコンピュータを利用して実施した. アルゴリズムは Fortran で実装し, 並列処理は OpenMP を用いて行った.

数値実験で用いるテスト問題は以下の 5 つである [3].

$$\text{問題 } 1 \begin{cases} f(x) = 1 + \sum_{i=2}^{n} \left\{ 100(x_i - x_{i-1}^2)^2 + (1 - x_{i-1})^2 \right\} \\ x^{(0)} = (1/n, \ldots, 1/n) \\ f^* = 1 \\ x^* = (1, \ldots, 1) \end{cases}$$

$$\text{問題 } 2 \begin{cases} f(x) = \sum_{i=1}^{n-1} \left\{ (x_i^2 + x_n^2)^2 - 4x_i + 3 \right\} \\ x^{(0)} = (3, \ldots, 3) \\ f^* = 0 \\ x^* = (1, \ldots, 1, 0) \end{cases}$$

$$\text{問題 } 3 \begin{cases} f(x) = \sum_{i=2}^{n} \left\{ (x_{i-1}^2 + x_i^2)^2 - 4x_{i-1} + 3 \right\} \\ x^{(0)} = (2, \ldots, 2) \\ f^* \sim 1108 \quad (n = 1000) \end{cases}$$

$$\text{問題 } 4 \begin{cases} f(x) = \sum_{i=4}^{n-1} \left\{ (x_{i-3}^2 + 2x_{i-2}^2 + 3x_{i-1}^2 + 4x_i^2 + 5x_n^2)^2 - 4x_{i-3} + 3 \right\}^2 \\ x^{(0)} = (1, \ldots, 1) \\ f^* \sim 2342 \quad (n = 1000) \end{cases}$$

$$\text{問題 } 5 \begin{cases} f(x) = \sum_{i=1}^{n} \dfrac{(x_i - 1)^2}{10^5} + \left(\sum_{j=1}^{n} x_j^2 - \dfrac{1}{4} \right)^2 \\ x^{(0)} = (3, \ldots, 3) \\ f^* \sim 0.00968627 \quad (n = 1000) \end{cases}$$

これらのテスト問題に対して, 変数の次元 n を変化させて計算実験を行い, それらに対する計算時間を計測する.

5.1 実験結果

以下の実験結果において, 各 p の値に対して, T_1 は Step 2 で p 個の部分問題を 1 つの CPU で逐次計算したときの実行時間, T_p は Step 2 で p 個の部分問題を p 個のプロセッサで並列に計算したときの実行時間を表す.

$p = 1$ のときは, 通常の信頼領域法アルゴリズムを用いたことを意味する. なお, アルゴリズムの終了条件として, 以下の式を用いる.

$$\|\nabla f(x^{(k)})\| < 10^{-5}$$

また, 並列化効率とは以下の式で定義される値であり, この値が低いと, プロセッサの数 p に見合った並列効果が得られていないことを表す. 一般的に並列化効率が 0.5 以上だと並列化が効果的であるとみなせる.

$$\text{並列化効率} = \frac{T_1}{T_p} \times \frac{1}{p}$$

表 1: 問題 1 の結果 ($n = 400, 800, 1200$)

$n = 400$	$p = 1$	$p = 4$	$p = 8$	$p = 16$
$T_1(\text{sec})$	130.6	3.960	2.102	1.217
$T_p(\text{sec})$	-	2.410	0.640	0.400
反復回数	939	1123	1492	2200
並列化効率	-	0.411	0.410	0.190

$n = 800$	$p = 1$	$p = 4$	$p = 8$	$p = 16$
$T_1(\text{sec})$	-	43.18	11.88	5.621
$T_p(\text{sec})$	-	21.72	4.339	1.795
反復回数	-	2052	2332	3078
並列化効率	-	0.497	0.342	0.196

$n = 1200$	$p = 1$	$p = 4$	$p = 8$	$p = 16$
$T_1(\text{sec})$	-	232.8	44.91	15.98
$T_p(\text{sec})$	-	113.1	12.93	4.317
反復回数	-	2969	3198	3904
並列化効率	-	0.514	0.434	0.231

表 1 において, 問題 1 は他の問題に比べ反復回数が多く, 計算量が膨大になるため, $n = 800, 1200$ における $p = 1$ のときの結果は得られていない. 表 1 より, p の数が多いほど反復回数は増加しているが, プログラムの計算時間が短くなっているのが分かる. また, 次元数 n が大きくなるほど並列化効率は上がっているもののあまり良い結果とはいえない.

表 2: 問題 2 の結果 ($n = 400, 800, 1200$)

$n = 400$	$p = 1$	$p = 4$	$p = 8$	$p = 16$
T_1(sec)	2.153	0.135	0.070	0.057
T_p(sec)	-	0.041	0.027	0.027
反復回数	9	27	50	98
並列化効率	-	0.823	0.324	0.132

$n = 800$	$p = 1$	$p = 4$	$p = 8$	$p = 16$
T_1(sec)	30.45	1.017	0.440	0.265
T_p(sec)	-	0.268	0.080	0.053
反復回数	10	27	50	98
並列化効率	-	0.949	0.688	0.313

$n = 1200$	$p = 1$	$p = 4$	$p = 8$	$p = 16$
T_1(sec)	210.4	3.617	1.310	0.746
T_p(sec)	-	1.124	0.191	0.120
反復回数	10	28	50	98
並列化効率	-	0.804	0.857	0.389

表 2 より, 問題 2 の場合は, 変数の次元 n を変化させても反復回数がほとんど変わらないことが分かる. また, 問題 1 と同様に, p を増やすほど反復回数が増加するが計算時間の短縮が確認できる. n が大きくなるにつれ, $p = 8, 16$ における並列化効率が良くなることも分かる.

表 3: 問題 3 の結果 ($n = 1000$)

$n = 1000$	$p = 1$	$p = 4$	$p = 8$	$p = 10$
T_1(sec)	128.3	3.607	1.625	1.289
T_p(sec)	-	0.985	0.238	0.171
反復回数	16	59	107	129
並列化効率	-	0.915	0.853	0.754

表 4: 問題 4 の結果 ($n = 1000$)

$n = 1000$	$p = 1$	$p = 4$	$p = 8$	$p = 10$
T_1(sec)	79.17	2.315	1.148	0.941
T_p(sec)	-	0.693	0.176	0.136
反復回数	12	40	78	94
並列化効率	-	0.835	0.815	0.692

表 5: 問題 5 の結果 ($n = 1000$)

$n = 1000$	$p = 1$	$p = 4$	$p = 8$	$p = 10$
T_1(sec)	150.6	52.47	14.99	15.42
T_p(sec)	-	17.45	5.745	6.678
反復回数	21	1719	1499	2139
並列化効率	-	0.752	0.326	0.144

表 3 と表 4 より, 問題 3 と問題 4 においても p が大きくなるにつれ, 反復回数が増加するが, 計算時間の短縮が確認できる. また, 並列化効率は問題に依存することが分かる. 表 5 より, 問題 5 においては p を増やすことで計算時間は短くなっているものの, 反復回数の増加が著しく, 他の問題に比べ, 計算時間が短縮できていないことが分かる.

5.2 収束率

通常の信頼領域法では 3.3 節で述べたように 2 次収束性が保証されている. この節では数値実験を通して, 生成される点列の挙動を調べ, 提案アルゴリズムの収束性を見る. 実験方法として, 各反復で生成される点 $x^{(k)}$ と最適解 x^* との距離 $\|x^{(k)} - x^*\|$ を計算し, グラフに表す. 以下では, 問題 1 と問題 2 の $n = 200$ の場合について調べる. 以下の図において, 横軸は反復回数 k, 縦軸は $\log_{10} \|x^{(k)} - x^*\|$ を表す.

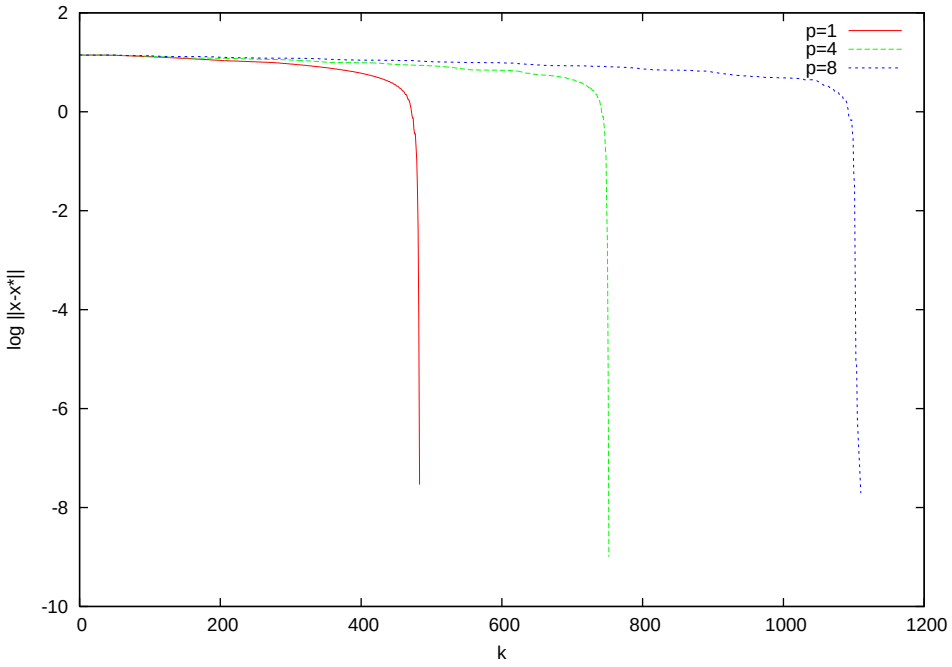

図 1: 問題 1 における反復点と最適解の距離

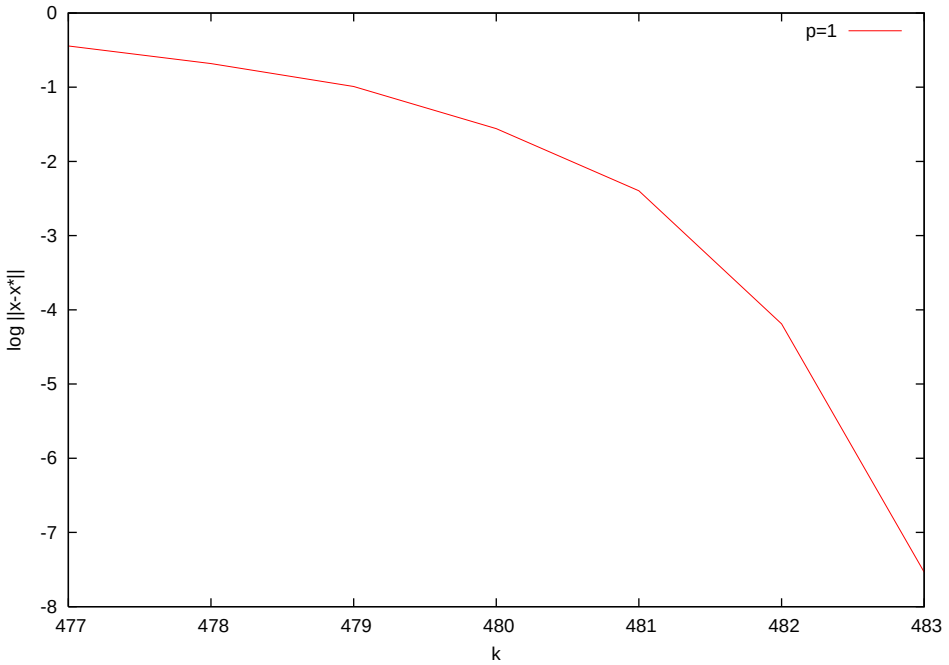

図 2: 問題 1 に対する $p = 1$ の場合の収束の最終段階の挙動

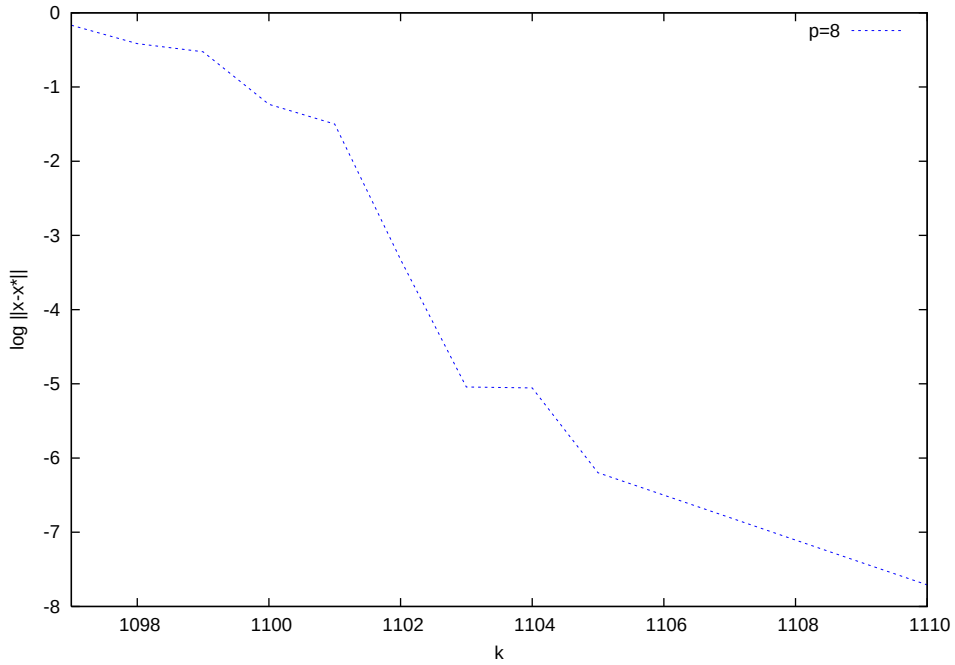

図 3: 問題 1 に対する $p = 8$ の場合の収束の最終段階の挙動

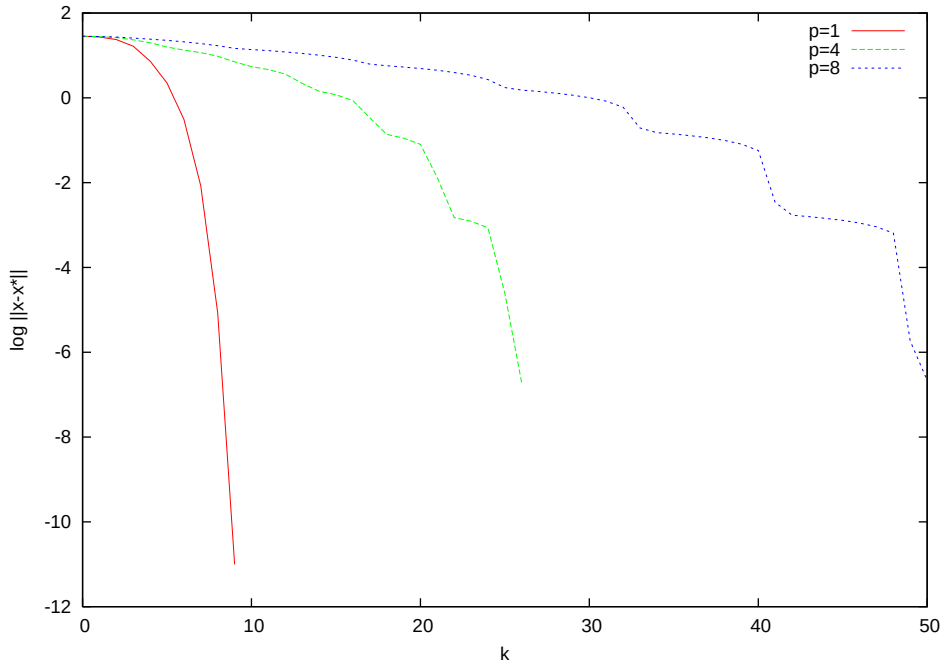

図 4: 問題 2 における反復点と最適解の距離

図 1 において, 赤, 緑, 青はそれぞれ $p = 1$, $p = 4$, $p = 8$ の場合の点列の挙動を表す. 各 p に対し, 反復の最終段階で点列が最適解に急速に近づくのが分かる. さらに, 収束の様子をより詳細に見るため, 図 2, 図 3 で $p = 1$ と $p = 8$ のそれぞれの場合に対して, 反復の最終段階の挙動を拡大したものを示す. $p = 1$ では反復ごとに傾きが急になっており, 2 次収束の特徴を表している. 一方, $p = 8$ ではそういった特徴は見られない. 図 4 においても p が増えるにつれ収束率が劣化しているのが分かる.

5.3 考察

提案アルゴリズムでは, 通常の信頼領域法に比べ反復回数が増加した. それは次の反復点を低次元部分空間で探すため, 反復点の動きが制限されたためと思われる. しかし, 一回の反復における計算量が少ないため, 全体での実行時間は短縮された. 問題 1-問題 4 において, 次の反復点の探索を低次元空間に限定しているにもかかわらず, 反復回数は比較的増加していない. このことから, 計算量を減らすという点において, 提案アルゴリズムの効果が期待できる. また, 問題 5 は他の問題とは違いヘッセ行列が密構造となっている. このような問題に対して, 提案アルゴリズムのように座標方向のベクトルで探索する部分空間を定めると, 元の問題のヘッセ行列の情報の損失が大きい. そのため, 他の問題に比べ, 反復回数が増え, 計算時間の短縮の効果が得られにくいと考えられる. 部分空間の選び方については今後の重要な課題である.

次に, 並列化効率について考える. 変数の次元 n が大きくなるにつれ, 並列化効率が向上したのは提案アルゴリズムにおける並列部分 (Step 2) の計算量が同期部分 (Step 3〜Step 5) の計算量に比べ大きくなったためと考えられる. また, 問題によって並列化効率が異なる原因として, 並列部分において各プロセッサの計算量にばらつきがあることが挙げられる. 3.2 節で述べたように信頼領域法の部分問題を解く手順は 2 段階ある. そのため, あるプロセッサは 1 段階目で解が求まり計算が終了するが他のプロセッサで 2 段階目を計算しているような場合が考えられる. このような状況を作らないために, あるプロセッサが 1 段階目で計算が終わると他のプロセッサにおいても 2 段階目の計算を打ち切る等の工夫が考えられる.

提案アルゴリズムでは, 目的関数値を最も減らす空間上に次の反復点を定めるため, 次の反復点の選択が制限されている. PVT 法において Step 3 の部分問題 (5) を近似的に解き, 次の反復点を $p + 1$ 次元空間で探索する

ことができれば, 収束の速さが改善できると期待できる.

6　結論と今後の課題

本報告書では大規模な問題に対し, PVT 法を適用した. その際, 各反復に現れる部分問題を信頼領域法を用いて解くアルゴリズムを提案した. また, 提案アルゴリズムを実装し, 数値実験を通してアルゴリズムの有効性を検証した. その結果, 通常の信頼領域法よりも計算時間を短縮できることが確認された.

今後の課題としては, 5.2 節で述べたようなアルゴリズムの効率化や既存のソルバーとの比較などが挙げられる. また, 本報告書の数値実験ではプログラムを作成する際に, 大規模問題においてしばしば生じる行列の疎構造を考慮していない. そういったプログラムの実装上の改善も今後の重要な課題である.

謝辞

日頃から御教授くださり, 本研究に対しても適切なご指摘と熱心なご指導を賜った福嶋雅夫教授に深く感謝の意を表します. また，日頃からお世話になっている山下信雄准教授, 林俊介助教ならびに福嶋研究室の皆様に厚く御礼申し上げます.

参考文献

[1] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. *Trust-Region Methods*. SIAM, 2000.

[2] M. Fukushima. Parallel variable transformation in unconstrained optimization. *SIAM J.Optim.*, Vol. 8, pp. 658–672, 1998.

[3] E. Yamakawa and M. Fukushima. Testing parallel variable transformation. *Comput. Optim. Appl.*, Vol. 13, pp. 253–274, 1999.

[4] Y. X. Yuan. Subspace method for large scale nonlinear equations and nonlinear least squares. *Optim. Eng.*, Vol. 10, pp. 207–218, 2009.

[5] 茨木俊秀・福島雅夫. 最適化の手法. 共立出版, 1993.

[6] 矢部博. 工学基礎 最適化とその応用. 数理工学社, 2006.