跡見学園女子大学マネジメント学部紀要　第12号　（2011年10月14日）

動的計画法複合アルゴリズムのメカニズムと最適制御問題への応用

The Mechanism of The Hybrid Dynamic Programming Algorithm and Its Application to Optimal Control Problems

花　岡　照　明
Teruaki HANAOKA

要　旨

本論文では，先に開発した動的計画法と分枝限定法を併用した動的計画法複合アルゴリズムの特性を従来の動的計画法および最大原理の手法と比較し，その長所及び欠点について考察している．このアルゴリズムは基本的には微分可能性を仮定できないような状態の下でも適用が可能であり，複雑な非線形最適制御問題に対し，統一的に適用することができる．問題毎に技巧的な工夫を必要としない．この複合アルゴリズムでは，分枝限定操作を取り入れた従来の複合アルゴリズムに対し，計算上の工夫として，優先順位計算，代表点，およびコストの下界値を精度よく推定する繰り返し論理を組み入れることによって数値計算上の負荷（計算数とサイズ）を大幅に削減することに成功している．繰り返し論理では，計算数を左右する下界値の推定精度を向上させることによって高精度解を得ることができる．本論文では，この拡張した複合アルゴリズムのメカニズムとその特性を明らかにし，連続時間最適制御問題および離散時間最適制御問題に適用した結果が示されている．また，状態制約がある問題に対し，制約条件を完全に満たす最適軌道の生成が可能であることが示されている．

キーワード： 動的計画法，ブランチ・アンド・バウンド，最適制御問題，状態制約

1　はじめに

今日，航空宇宙をはじめ，多くの分野で，システムに対する要求は複雑かつ多様化してきている．一方，これらの要求を含んだ複雑な問題を容易に扱うことのできる計算法の開発が強く望まれている．本稿では，複雑な非線形最適制御問題を統一的に解くことが可能な動的計画法

複合アルゴリズムの構造を明らかにし，連続時間および離散時間の最適制御問題への適用特性，ならびに状態制約最適制御問題への適用性について考察する．

動的計画法[1)]は，基本的には，どのような複雑なシステムに対しても数値的に取り扱うことのできる手段としてよく知られている．さらに，動的計画法を用いると，大域的最適解（以下，大域解と表記）やフィードバック構造をもつ解が得られること，制約条件や非線形の取扱いが容易であること，状態の遷移やコスト関数の表現が表関数であってもよいなど，数多くの望ましい性質を利用できる．しかしながら，動的計画法は，「Bellman の次元の呪い」の問題[1)]，すなわち，状態変数の次元数の増加に伴って計算数が指数関数的に増加し，特別な問題を除き，計算実行上，求解が困難，という致命的な欠陥を持つため，次元の高い問題への適用は，きわめて難しいとされてきた．本稿では，動的計画法の応用で最も障害となっている「過大な計算数」を低減する手法について考察する．

動的計画法の計算数を削減するのに，分枝限定法の限定操作を併用する方法がある[2)~5)]．この方法が効果を発揮するかどうかの最大要因は，分枝限定法の限定操作で使用する強力な下界値の作り方にある．Hanaoka and Tanabe[4)] および Hanaoka[5)] では，再突入飛行体や航空機の経路最適化などの実際的な連続系に対して，動的計画法と分枝限定法を併用した，動的計画法複合アルゴリズム（基本形複合アルゴリズム）を適用し，計算数を大幅に削減させることに成功している．その中では，アルゴリズムの実行で必要となるコストの下界値を，システムの特殊性を利用して計算する方法が提案されている．しかしながら，本稿ではシステムの特殊性を用いなくても限定操作を効果的に行い得ることを明らかにする．また，より最適な経路を先に計算させるという優先順位の概念を導入することにより，コスト関数の大小比較の手間を大幅に削減している．一方，代表点と呼ばれる実軌道（トラジェクトリー）上にコスト関数の評価点を取ることにより，従来，動的計画法の適用において煩わしい問題であった内挿計算や外挿計算を一切省くことに成功している．さらに，この複合アルゴリズムでは，基本形複合アルゴリズムに対し，計算上の工夫として，コストの下界値を精度よく推定する繰り返し論理を組み入れた算法（繰り返し型複合アルゴリズム）を導入することによって，数値計算上の負荷（計算数とサイズ）を大幅に削減し，同時に解の精度を大幅に向上させることに成功している．この動的計画法複合アルゴリズムは，ダーウィン進化，すなわち，淘汰，増殖，突然変異の３つの繰り返し過程，の一部をまねたような算法であり，従来型動的計画法[1)]とは異なった構成法に基づいている．淘汰過程は，本法の「ブロックを使った優先順位計算」に，増殖過程は，量子化した制御の適用による「経路群の繰り返し生成」に対応する．突然変異の組入れは，本稿では扱わない．本稿で述べる複合アルゴリズムの計算メカニズムは，分散的に，ばらばらに存在する経路群を，最適な経路へと集約させるもので，その理論的根拠を前向き動的計画法[4)~6)]に置いている．

本稿では，上述の分枝限定操作，優先順位計算，代表点，および逐次近似のこれら４つの概念を併用したメカニズムを用いることにより，初めて，従来の動的計画法の適用範囲を大幅に拡大することが可能であることを明らかにする．

2　最適制御問題とその解法

いま，次の最適制御問題を考える．

$$\text{Minimize} \quad J = \sum_{k=0}^{N-1} L_k(\boldsymbol{x}_k, \boldsymbol{u}_k) + \Phi_N(\boldsymbol{x}_N) \tag{1}$$

$$\text{Subject to} \quad \boldsymbol{x}_{k+1} = \boldsymbol{g}_k(\boldsymbol{x}_k, \boldsymbol{u}_k)(k = 0, \ldots, N-1) \tag{2}$$

$$\boldsymbol{x}_0 = \boldsymbol{c}_0 \tag{3}$$

$$\boldsymbol{x}_N \in \Omega_F \tag{4}$$

$$\boldsymbol{x}_k \in X_k \quad (k = 1, \ldots, N-1) \tag{5}$$

$$\boldsymbol{u}_k \in U(\boldsymbol{x}_k) \quad (k = 0, \ldots, N-1) \tag{6}$$

ここで，$\boldsymbol{x}_k$ と $\boldsymbol{u}_k$ を，それぞれ，p 次元状態ベクトルおよび q 次元制御ベクトルとする．また，k を段変数の添え字，$\boldsymbol{g}_k$ を p 次元ベクトル値関数とする．$L_k(\boldsymbol{x}_k, \boldsymbol{u}_k)(k = 0, 1, \ldots, N-1)$ は k 段のコスト関数，$\Phi_N(\boldsymbol{x}_N)$ は最終段のコスト関数である．また，X_k と $U(\boldsymbol{x}_k)$ は，それぞれ k 段での許容状態集合と許容制御集合である．(3) と (4) 式は，それぞれ，初期条件と終端条件であり，$\Omega_F(\Omega_F \subset X_N)$ は終端条件を満たす集合である．

上述の最適制御問題を解くための基本的な手法として動的計画法と最大原理[8] からのアプローチがある．また，動的計画法は，大域的最適解が得られることや制約条件の取り扱いの容易さなどの枚挙法[7] (pp.14–16) の特徴を保存しながら組み合わせ爆発的な過度な計算量を削減するために，最適性の原理を用いコスト関数の値を各段に対して埋め込んでいる．しかし動的計画法ではコスト関数や制御変数の内挿手続きを用いる必要があるため，枚挙法の素晴らしい特徴である解の精度を放棄している．一方，最大原理では解の精度は高いが制約条件の取り扱いの複雑さや計算実行上の不安定な問題がある．各手法の特徴を表１のように纏めることができる．本稿で述べる複合アルゴリズムは，枚挙法の利点を最大限保存しながら動的計画法の欠点を改善し，最大原理の高精度解の利点を取り込んだアルゴリズムである．

3　複合アルゴリズムの概観

3．1　複合アルゴリズムの手続き

複合アルゴリズムには，基本形複合アルゴリズムと繰り返し複合アルゴリズムがある．それらは以下の３つの手続きで構成される．

表１　３つの計算法の比較

項目	枚挙法	動的計画法	最大原理
必要計算機メモリー	×（爆発的）	×	○
計算量・計算時間	×（爆発的）	×	○
コスト関数の内挿	○（不要）	×（必要）	○（不要）
制御変数の内挿	○（不要）	×（必要）	○（不要）
解の精度	○（高い）	×（低い）	○（高い）
大域的最適解	○	○	×（局所解）
計算実行の安定性	○	○	×
制約条件の取扱い	○（容易）	○（容易）	△（複雑）
境界条件の取扱い	○（容易）	○（容易）	△（複雑）

○：要求を満たす，　△：問題点を含む，　×：要求を満たさない

- 前向き動的計画法
- 分枝限定法の限定操作
- 逐次近似

基本形複合アルゴリズムは最初の２つの手続きを組み合わせたものであり，繰り返し複合アルゴリズムは基本形アルゴリズムにさらに逐次近似の手続きを加えたものである．ただし，ここで述べる前向き動的計画法は，従来の前向き動的計画法[7)(pp.170–177)]とは異なる手続きを用いており，後に述べる優先順位計算と代表点の概念に基づいて構成される．

３．２　代表点と前向き動的計画法の概念

代表点の概念図を図１に示す．まず，前向き動的計画法を適用するために問題の定義域を「ブロック」と呼ぶ単位に量子化する．前向き動的計画法の計算は初期点から木が枝を伸ばすように前向きに進む (図中の矢印)．各段のブロック内で初期点からのコストが最小となる到達点に高々１個,「代表点」(図中の○印) をとる．もし，同一ブロックにそれ以外のより大きなコストを持つ到着経路がある場合はそれらを削除する．代表点の位置は，格子点上でなくてもよい．この計算は経路が最終段 N に到達するまで行う．図中，A, B, C, D, F, G が代表点である．しかし，D 点と同じブロック内にある E 点は初期点からのコストが $f_2 = 6$ であり，このブロックには既により小さなコスト値 $f_2 = 3$ を持つ D 点が代表点として取れれるため削除される．

３．３　優先順位計算の概念

優先順位計算の概念図を図２に示す．図中，各矢印の先端，すなわち代表点でコスト関数が

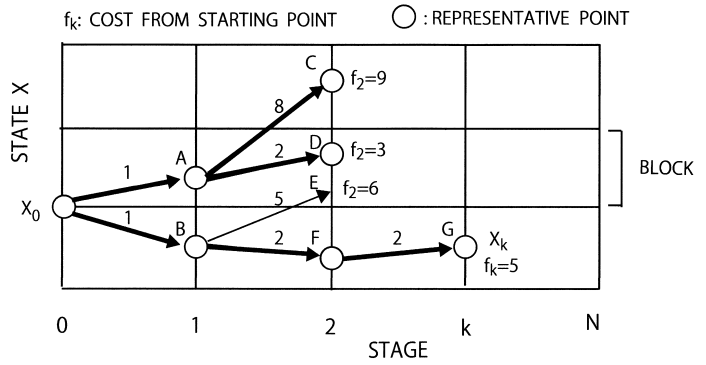


図 1　ブロックの代表点で定義された前向き動的計画法

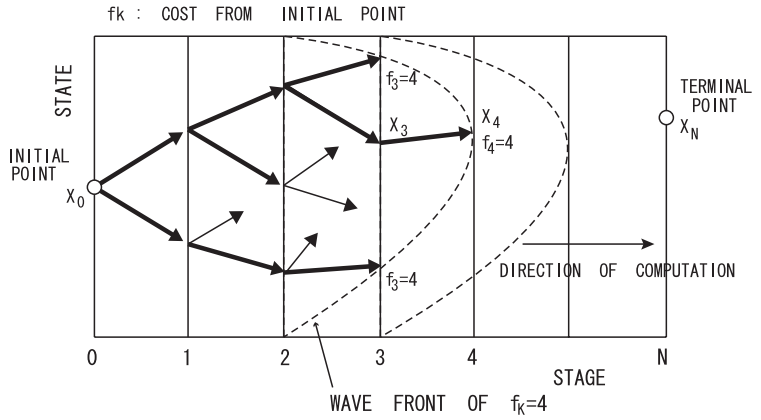


図 2　優先順位計算によって作られた等コスト面の wave front

計算される．コスト関数を計算する順序は，最初に初期点，以後，初期点から矢印先端までの経路に対応するコスト関数の値の昇べきの順である．このような順序に基づく計算を優先順位計算と呼ぶことにする．図中，矢印に接している Wave front と表記された曲線は，等しいコスト関数の値をもつ代表点を連ねた曲線であり，前向き動的計画法の計算が進行していく最先端を表している．wave front が終端点に到達したとき，前向き動的計画法の計算は終了する．

3．4　限定操作の概念

限定操作の概念図を図 3 に示す．限定操作は，前向き動的計画法の計算過程において，最適経路の候補とならない経路を削除する操作である．すべての経路は代表点において，最適経路の候補になることができるかどうかのチェックを受ける．図中，初期点 x_0 から D 点までのコスト値は f_k=5 であり，D 点から最終段 N までのコストの下界値は M_k=4 である．これらの値を削除の条件式と呼ばれる式，すなわち，$f_k + M_k > I$ に代入すると，$f_k + M_k = 5 + 4 = 9 > 8 = I$

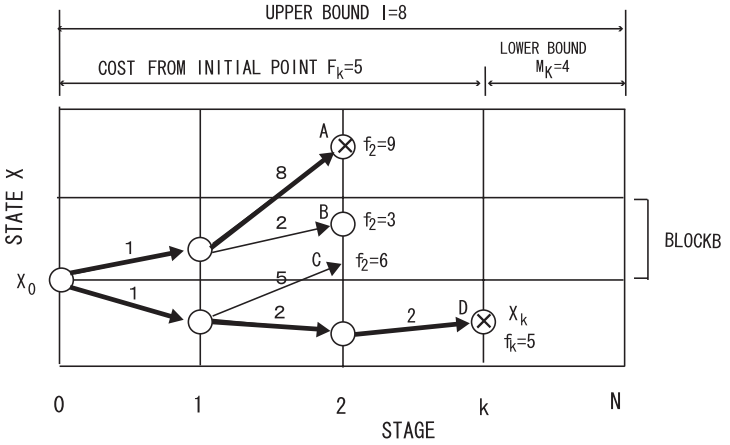


図３　限定操作によって削除された代表点とそのブロック（× 印）

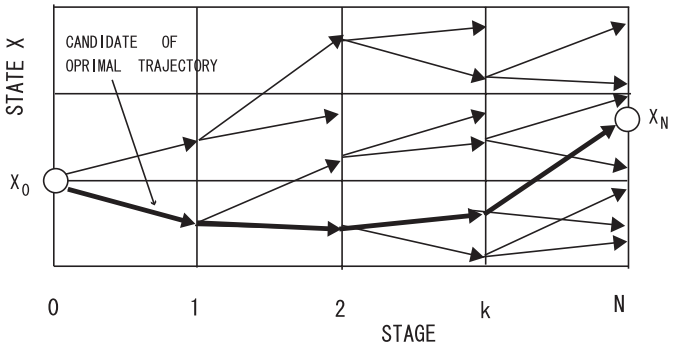


図４　逐次近似によって計算領域が狭まっていく経路群（基本形複合アルゴリズム $i = 1$）

となり，削除条件式を満たすため D 点および D 点の属するブロックは削除される（× 印）．ただし，I は最適コストの上界値である．また，初期点から A 点までのコスト値は $f_2 = 9$ であり，下界値 M_2 の値に関わらず削除条件式を満たすため，A 点および A 点の属するブロックは削除される（× 印）．一方，初期点から B 点までのコストは $f_2 = 3$ であり削除条件式を満たさないから，この時点では最適経路の候補であり，この代表点とブロックは削除されない．

３．５　逐次近似の概念

逐次近似の概念図を図 4 と図 5 に示す．図 4 は基本複合アルゴリズムの実行の様子である．初期点からの経路は，粗い量子化の下で定義域全体に広がっている．太い実線はこのアルゴリズムで得られた最適解の近似解である．図 5 は繰り返し複合アルゴリズムの実行の様子である．定義域と制御入力は，図 4 の基本形複合アルゴリズムよりもさらに細分化され，初期点からの経路は基本複合アルゴリズムの最適解の近似解の周りに集中している．これらの集中した経路

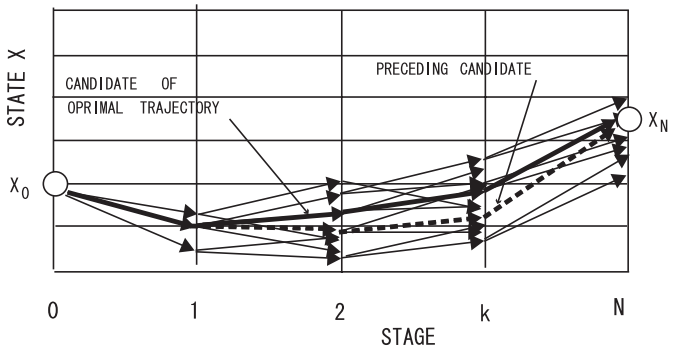


図 5　逐次近似によって計算領域が狭まっていく経路群（繰り返し複合アルゴリズム $i = 2, 3, \ldots$）

の中から繰り返し複合アルゴリズムの解が選ばれることになり，基本形複合アルゴリズムの解はより精度を改善した最適解の近似解へと逐次近似される．図中，破線は基本形複合アルゴリズムによって得られた最適解の近似解である．

4　複合アルゴリズム

Bellman の動的計画法[1] は，状態の遷移方向，通常は時間の増加方向に関して，後向きに定式化されることが多い．本章で提案する基本形複合アルゴリズムは前向き最適性の原理を用いており，初期点から前向きに定式化する．

いま，量子化した許容制御 $\boldsymbol{u}_k \in U(\boldsymbol{x}_k)$ を初期点 $\boldsymbol{x}_0$ から，各段 k で繰り返し適用してできる初期点からの実行可能経路の集合を，$\{X_k^\circ\} = \bigcup_{i=0}^{k} X_i^\circ$ とおく．ただし，X_k° は，再帰的に定義され，$X_0^\circ = \{\boldsymbol{x}_0\}, X_{k+1}^\circ = \{\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_{k+1} = \boldsymbol{g}_k(\boldsymbol{x}_k, \boldsymbol{u}_k), \boldsymbol{x}_k \in X_k^\circ, \boldsymbol{u}_k \in U(\boldsymbol{x}_k)\}(k = 0, \ldots, N-1)$ である．また，$\boldsymbol{g}_k$ は状態点 $\boldsymbol{x}_k$ を新たな状態点 $\boldsymbol{x}_{k+1}$ に移す状態遷移関数である．ここで，計算上の工夫として，前向き動的計画法における計算点を，押し出し計算によって，実行可能経路の集合 $\{X_k^\circ\}$ 上にとる．さらに，コスト関数の計算と比較を，正確に経路上のコスト値を用いて行う．したがって，この処理では，コスト関数の比較を行う計算点が実行可能経路上にない場合に必要となるコスト関数の内挿計算を含まない．しかし，このままでは，段数 N が大きいとき，実行可能経路数の巨大化を招くため，量子化の手続きを行う．

4．1　代表点と前向き動的計画法

複合アルゴリズムでは，前向き動的計画法を適用するために，許容状態集合 X_k を適当な部分集合 $X_{k1}, X_{k2}, \ldots, X_{kn_k}$ に量子化する．ただし，$X_k = \bigcup_{i=1}^{n_k} X_{ki}, X_{ki} \cap X_{kj} = \emptyset (i \neq j)$ とする．以後，この部分集合 X_{ki} を「ブロック」と呼ぶ．各ブロック X_{ki} に対して，初期点 $\boldsymbol{x}_0$ から以下に示す代表点だけを辿って到達できる経路が存在する場合のみ，一個ずつそのブロッ

クの代表点 $\boldsymbol{x}_{ki}(\in X_{ki})$ とそのコスト関数値を以下のように定義する．

$$f_k(\boldsymbol{x}_{ki}) = \min_{\boldsymbol{x}_k}\{\tau(\boldsymbol{x}_k) \mid \boldsymbol{x}_k \in X_{ki}\}$$
$$(i = 1, \ldots, m_k,\ k = 0, \ldots, N) \tag{7}$$

ただし，$m_k(m_k \leq n_k)$ は，k 段の代表点の数である．$\tau(\boldsymbol{x}_k)$ は，前段の代表点 $\boldsymbol{x}_{k-1i}(\boldsymbol{x}_{k-1i} \in X_{k-1i})$ に対して，量子化した各 $\boldsymbol{u}_{k-1} \in U(\boldsymbol{x}_{k-1i})$ を適用し，以下の式を用い，押し出し計算によって求めた値とする．

$$\begin{aligned}
&\tau(\boldsymbol{x}_0) = 0\\
&\tau(\boldsymbol{x}_k) = f_{k-1}(\boldsymbol{x}_{k-1i}) + L_{k-1}(\boldsymbol{x}_{k-1i}, \boldsymbol{u}_{k-1})\\
&\boldsymbol{x}_k = \boldsymbol{g}_{k-1}(\boldsymbol{x}_{k-1i}, \boldsymbol{u}_{k-1})\\
&\qquad (i = 1, \ldots, m_k, k = 1, \ldots, N)
\end{aligned} \tag{8}$$

ここで，$U(\boldsymbol{x}_{k-1i})$ は許容制御であり，L_k は 1 段当りのコストである．ただし，最終段 N でのコスト $\Phi_N(\boldsymbol{x}_N)$ を，$N-1$ 段でのコスト L_{N-1} に含めるものとする．この代表点 $\boldsymbol{x}_{ki}$ は，初期点 $\boldsymbol{x}_0(\boldsymbol{x}_0 = \boldsymbol{x}_{01})$ から k 段の各ブロック上での到達点までの経路の中で，最小のコスト関数の値を与える点である．この点は初期点から再帰的に計算できる．以降では代表点におけるコスト関数を最小コスト関数と呼び，代表点 $\boldsymbol{x}_k$ の関数として $f_k(\boldsymbol{x}_k)$ と書く．このとき，有限個の代表点 $X_k^\circ = \{\boldsymbol{x}_{k1}, \boldsymbol{x}_{k2}, \ldots, \boldsymbol{x}_{km_k}\}(k = 0, \ldots, N)$ を考慮した前向き動的計画法を構成することができる．すなわち，

$$\begin{aligned}
&f_0(\boldsymbol{x}_{01}) = 0\\
&f_{k+1}(\boldsymbol{x}_{k+1i}) = \min_{\boldsymbol{x}_{ki}, \boldsymbol{u}_k}\{f_k(\boldsymbol{x}_{ki}) + L_k(\boldsymbol{x}_{ki}, \boldsymbol{u}_k) \mid\\
&\qquad \boldsymbol{x}_{ki} \in X_k^\circ, \boldsymbol{u}_k \in U(\boldsymbol{x}_{ki})\} \quad (i = 1, \ldots, m_k)\\
&\boldsymbol{x}_{k+1i} = \boldsymbol{g}_k(\boldsymbol{x}_{ki}, \boldsymbol{u}_k) \qquad (k = 0, \ldots, N-1)
\end{aligned} \tag{9}$$

である．一方，代表点以外の各ブロック内の状態点の最小コスト関数値は，代表点の値で近似し，

$$\text{もし}\boldsymbol{x}_k \in X_{ki}\text{ならば，}\ f_k(\boldsymbol{x}_k) = f_k(\boldsymbol{x}_{ki})$$
$$(i = 1, 2, \ldots, m_k,\ k = 0, 1, \ldots, N-1)$$

とする．

4.2 優先順位計算

基本形複合アルゴリズムの計算は，初期点から前向きに進行する．初期点および各段の代表点では，許容制御を量子化した制御が適用され，最適経路候補群が繰り返し生成される．このとき，(8) 式によって計算されるコスト関数値 $\tau(\boldsymbol{x}_k)$ の小さい順に，各経路がブロックに到着するように処理する．以後，この処理を優先順位計算と呼ぶ．この処理によって，代表点の定義により，ブロックに最初に到着した経路がそのブロックの代表点となる．そのため，それ以後に到着するいかなる経路も代表点となることはないため，それらを削除できる．

このアルゴリズムは，初期点からの経路のいずれかが，終端条件を満たす状態集合 Ω_F に最初に到着したとき終了する．このとき，この先着経路が最適経路となる．この先着経路に対応する最適コストを

$$f^*_{0,N} = \min_{\boldsymbol{x}_N}\{f_N(\boldsymbol{x}_N) \mid \boldsymbol{x}_N \in \Omega_F\} \tag{10}$$

と定義する．

4.3 限定操作とクリアランス

複合アルゴリズムでは，前向き動的計画法の計算量を削減するために，分枝限定法の限定操作を応用する．

いま，後向き動的計画法[1] の最小コスト関数 $J_k(\boldsymbol{x}_k)$ の下界値を $M_k(\boldsymbol{x}_k)$，原問題 (1)〜(6) 式下での最適解を $f^*_{0,N}$ とする．また，最適値 $f^*_{0,N}$ の上界値を I とする．分枝限定法の上界値（実行可能解に対応する目的関数値）は通常，計算の進行に伴って得られた上界値の最小の値で改良するが，提案するアルゴリズムの上界値は，計算終了まで更新しない．それらの下界値と上界値が満たすべき条件は，それぞれ

$$\begin{aligned} M_k(\boldsymbol{x}_k) &\le J_k(\boldsymbol{x}_k), \qquad \boldsymbol{x}_k \in X_k \\ &= \min_{\boldsymbol{u}_k \in U(\boldsymbol{x}_k), \boldsymbol{u}_{k+1} \in U(\boldsymbol{x}_{k+1}), \ldots, \boldsymbol{u}_{N-1} \in U(\boldsymbol{x}_{N-1})} \{\sum_{i=k}^{N-1} L_i(\boldsymbol{x}_i, \boldsymbol{u}_i) + \Phi_N(\boldsymbol{x}_N)\} \\ &\qquad (k = 0, \ldots, N-1) \end{aligned} \tag{11}$$

$$I \ge f^*_{0,N} \tag{12}$$

である．ここで，$J_k(\boldsymbol{x}_k)$ は後向き動的計画法での最小コスト関数の値である．もし，任意の状態 $\boldsymbol{x}_k$ を通る経路が

$$f_k(\boldsymbol{x}_k) + M_k(\boldsymbol{x}_k) > I \qquad (k = 0, \ldots, N) \tag{13}$$

を満たすならば，

$$f_k(\boldsymbol{x}_k) + J_k(\boldsymbol{x}_k) \geq f_k(\boldsymbol{x}_k) + M_k(\boldsymbol{x}_k) > I \geq f^*_{0,N}$$

なので，代表点 $\boldsymbol{x}_k$ は最適経路の一部になれない．よって代表点 $\boldsymbol{x}_k$ は代表点 X°_k の中から削除する．ただし，$f_0(\boldsymbol{x}_0)$ は，(8) 式より $f_0(\boldsymbol{x}_0) = \tau(\boldsymbol{x}_0) = 0$ である．また，ここでは，最終段 N でのコスト $\Phi_N(\boldsymbol{x}_N)$ を，$N-1$ 段でのコスト L_{N-1} に含めているので，形式的に，$J_N(\boldsymbol{x}_N) = 0$ とする．したがって，$M_N(\boldsymbol{x}_N) = 0$ である．削除できる代表点 $\boldsymbol{x}_k$ の数を増やすためには，(13) 式から明らかなように，より小さな上界値 I を，また，より大きな下界値 $M_k(\boldsymbol{x}_k)$ を求めればよい．

ここで，上界値と下界値の弱さを表わす量として，それぞれ，Δa および Δb_k を導入し，

$$\Delta a = I - f^*_{0,N}, \quad \Delta b_k = J_k(\boldsymbol{x}_k) - M_k(\boldsymbol{x}_k) \tag{14}$$

とおく．(14) 式の I と $M_k(\boldsymbol{x}_k)$ を (13) 式に代入すると

$$f_k(\boldsymbol{x}_k) + J_k(\boldsymbol{x}_k) > f^*_{0,N} + \Delta a + \Delta b_k \tag{15}$$

となる．(15) 式より，基本形複合アルゴリズムの計算量は，Δa と Δb_k の和 $\Delta\epsilon_k (\Delta\epsilon_k = \Delta a + \Delta b_k)$ に依存するとみなせる．以後，この和 $\Delta\epsilon_k$ をクリアランスと呼ぶ．基本形複合アルゴリズムを用いて大域解を得るためには，クリアランス条件，すなわち，

$$\Delta\epsilon_k = \Delta a + \Delta b_k \geq 0 \tag{16}$$

を満足する代表点を求めればよい．ここで，複合アルゴリズムの計算量が最小となるのは，$\Delta\epsilon_k = \Delta a + \Delta b_k = 0$ のときであることは明らかである．

4.4　基本形複合アルゴリズム

基本形複合アルゴリズムは，つぎの 10 ステップに要約できる．

1. (境界条件設定)：初期条件 $\boldsymbol{x}_0 = \boldsymbol{c}_0$ と終端条件 $\boldsymbol{x}_N \in \Omega_F$ を設定する．

2. (状態集合の量子化)：各段の状態集合 X_k を，n_k 個のブロック $X_{k1}, X_{k2}, \ldots, X_{kn_k}$ に分割する．ただし,$X_k = \bigcup_{i=1}^{n_k} X_{ki}, X_{ki} \cap X_{kj} = \emptyset\ (i \neq j)$ である．

3. (上界，下界の設定)：各 $k = 0, \ldots, N$ に対し，適当な $\boldsymbol{x}_k \in X_k$ に対する下界値 $M_k(\boldsymbol{x}_k)$ を計算する．ただし，有効な下界値を計算できないときは，$M_k(\boldsymbol{x}_k) = 0$ とし，I を $\Delta\epsilon_k \geq 0$ となるように設定する．また，一つの上界値 I を設定する．

4. (初期化)：$\Omega \leftarrow \{\boldsymbol{x}_0\}$, $\tau(\boldsymbol{x}_0) = 0$, $\Re \leftarrow \emptyset$ を行う．ただし，$\Re$ は，そこまでの最小コスト経路が確定したブロックの集合を表す．また，$\leftarrow$ は代入操作である．

5.（停止）：もし，$\Omega = \emptyset$ なら停止せよ．

6.（前向き動的計画法）：Ω の中から $\tau(\boldsymbol{x}^*)$ の値が最小である $\boldsymbol{x}^*$ を選び，$\Omega \leftarrow \Omega \backslash \{\boldsymbol{x}^*\}$ とせよ．ただし $\backslash$ は差集合の演算子である．$\boldsymbol{x}^*$ が属する段の値を k にセットし，$\boldsymbol{x}_k^* \leftarrow \boldsymbol{x}^*$ とせよ．

7.（終端テスト）：もし $\boldsymbol{x}_k^*$ が終端条件 $\boldsymbol{x}_N \in \Omega_F$ を満たすなら，停止せよ．この段階で最適解が得られる．

8.（代表点テスト）：もし $[x_k^*] \in \Re$ ならばステップ 5 へ行け．それ以外は $\Re \leftarrow \Re \cup \{[\boldsymbol{x}_k^*]\}$ とし，ステップ 9 へ行け．ただし $[y]$ は状態 y を含むブロックを示す．この段階で，$\boldsymbol{x}_k^*$ に到達させる最適制御 $\boldsymbol{u}_{k-1}^*$ が確定する．また，最小コスト関数の候補値 $\tau(\boldsymbol{x}_k^*)$ は，真の最小コスト関数値 $f_k(\boldsymbol{x}_k^*)$ となる．

9.（分枝限定操作）：量子化した各 $\boldsymbol{u}_k \in U(\boldsymbol{x}_k^*)$ に対して，次段の状態 $\boldsymbol{x}_{k+1}$ および最小コスト関数の候補値 $\tau(\boldsymbol{x}_{k+1})$ を，$\boldsymbol{x}_{k+1} = \boldsymbol{g}_k(\boldsymbol{x}_k^*, \boldsymbol{u}_k)$, $\tau(\boldsymbol{x}_{k+1}) = \tau(\boldsymbol{x}_k^*) + L_k(\boldsymbol{x}_k^*, \boldsymbol{u}_k)$ とする．そして，もし各 $\boldsymbol{x}_{k+1}$ に対し，$\tau(\boldsymbol{x}_{k+1}) + M_{k+1}(\boldsymbol{x}_{k+1}) \leq I$ ならば，$\Omega \leftarrow \Omega \cup \{\boldsymbol{x}_{k+1}\}$ とせよ．

10. ステップ 5 へ行け．

ステップ 6 で $\boldsymbol{x}^*$ を選択するとき，$\boldsymbol{x}$ をそれらに対応する最小コスト関数の候補値 $\tau(\boldsymbol{x}^*)$ のサイズの順序に整列しておくと，探索の手間を削減できる．また，ステップ 9 で $\tau(\boldsymbol{x}_{k+1}) + M_{k+1}(\boldsymbol{x}_{k+1}) \leq I$ の代わりに $\tau(\boldsymbol{x}_{k+1}) + M_{k+1}(\boldsymbol{x}_{k+1}) \leq I$ かつ $[\boldsymbol{x}_{k+1}] \in \Re$ とすると，Ω に格納する状態点の数を減らすことができ，データの記憶容量や探索の手間を削減できる．しかし，この操作は $[\boldsymbol{x}_{k+1}] \in \Re$ であるかどうかの判定回数を増加させるため，実際には両者のトレードオフとなる．

4.5 繰り返し複合アルゴリズム

ここでは，基本複合アルゴリズムの改良を行う．この改良型の複合アルゴリズムを繰り返し複合アルゴリズムと呼び，経路群を繰り返し生成することによりコストの上下界値の推定精度の向上を行う．

(13) 式による限定操作を強化し，より多くの計算数を削減するため，下界値を精度良く推定する繰り返し論理を考える．この目的のため，複合アルゴリズムの一連の計算を，より細かく量子化した状態集合と許容制御の下で繰り返す．すなわち，$i (i = 2, 3, \ldots)$ 回目の逐次計算では，状態集合 X_k を，前回 $(i-1)$ のブロックよりも，より小さなブロックに分割する．たとえ

ば，各状態変数に対し，量子化幅を前回の半分とし，したがって，量子化レベルを２倍に増やす．一方，許容制御 $u_k^{l,i}$ の範囲を

$$u_{kmin}^{l,i} \leq u_k^{l,i} \leq u_{kmax}^{l,i} \tag{17}$$

ここで，

$$u_{kmin}^{l,i} = u_k^{*l,i-1} - \Delta u_k^i, u_{kmax}^{l,i} = u_k^{*l,i-1} + \Delta u_k^i$$

$$(k = 0, 1, \ldots, N-1\ , l = 1, \ldots m\ ,\ i = 1, 2, \ldots)$$

によって計算する．ここで，$u_k^{*l,i-1}$ は $i-1$ 回目の繰り返し計算で得た最適制御であり，l を制御変数のベクトル成分の添え字とする．また，Δu_k^i は，通常，$\Delta u_k^i \leq \Delta u_k^{i-1}$ となるように設定し，制御入力の分割数を変えずに量子化幅を前回よりも細かくする．たとえば，大域解を得る保証を放棄し，$\Delta u_k^i = 0.5\Delta u_k^{i-1}$ とする．また，i 回目の許容制御 $u_k^{l,i}$ をつくる際，$u_k^{*l,i-1}$ を含めるように量子化し，かつ，前回の最適経路上の代表点を i 回目の代表点に加えると，最悪でも，前回の最適コスト値を保証できる．

しかし，状態変数や許容制御のこのような量子化レベルの増加の下では，複合アルゴリズムの計算数を指数関数的に増大させる可能性がある．この増大を抑制するため，上界値 I^i と下界値 $M_k^i(\boldsymbol{x}_k)(\boldsymbol{x}_k \in X_k)$ を，それぞれ，前回 $(i-1)$ の繰り返し計算で得た，最終最適コスト $f*_{0,N}^{i-1}$ と最適経路上のコスト値を用い，それぞれ，

$$I^i = f*_{0,N}^{i-1} \qquad (i = 2, 3, \ldots) \tag{18}$$

$$\begin{aligned} M_k^i(\boldsymbol{x}_k) &= J_k^{i-1}(\boldsymbol{x}_k^{*i-1}) \\ &= \sum_{l=k}^{N-1} L_l(\boldsymbol{x}_l^{*i-1}, \boldsymbol{u}_l^{*i-1}) + \Phi_N(\boldsymbol{x}_N^{*i-1}) \\ &\quad (k = 0, 1, \ldots, N-1,\ i = 2, 3, \ldots) \end{aligned} \tag{19}$$

によって強化する．ここで，$\boldsymbol{x}_l^{*i-1}$ と $\boldsymbol{u}_l^{*i-1}$ は，それぞれ，$i-1$ 回目の繰り返し計算で得た最適経路上の状態量と制御量である．(18) 式による I^i は良好な上界値を与える．また，(19) 式による M_k^i は，前回の最適経路の近傍で，ぴったりとした下界値を与える．

一方，この繰り返し複合アルゴリズムの現実的な停止条件を

$$|f*_{0,N}^{i} - f*_{0,N}^{i-1}| \ll \epsilon_1 \tag{20}$$

で与える．ただし，$\epsilon_1 \ll 1$ とする．

このように，繰り返し複合アルゴリズムでは，上界値と下界値を逐次改良する過程で，同時に，初期点からの最小コストを，より精密な値に逐次近似している．

5　複合アルゴリズムの特性

繰り返し複合アルゴリズムは，初回の計算で基本形複合アルゴリズムを実行し，その計算結果から得られた下界値と上界値を２回目以後の繰り返し計算に反映させる算法である．ここでは，繰り返し複合アルゴリズムの特性を調べるため，繰り返し複合アルゴリズムをいくつかの代表的な問題に適用し，解の収束性，解の精度，計算量を調べる．また，他の計算法との比較を行う．それらの結果より，基本形複合アルゴリズムでは得られない，繰り返し複合アルゴリズムの特徴を浮き彫りにし，応用上，どのような点が有益なのかを明らかにする．

5．1　連続時間最適制御問題への適用例

まず，高非線形性を持つ連続時間最適制御問題への適用結果を示す．この例題の状態方程式，評価関数，および境界条件は以下の通りである．

問題：連続-1[Sirisena, 1979]

$$\text{Minimize} \qquad J = \int_0^{0.5} (10x(t)^2 + u(t)^2)dt + 10x(0.5)^2 \tag{21}$$

$$\text{Subject to} \qquad \dot{x}(t) = -0.2x(t) + 10\tanh u(t) \quad t \in [0, 0.5] \tag{22}$$

$$x(0) = 5 \tag{23}$$

この問題に対し，基本形複合アルゴリズムであるイテレーション 0 の最初の計算では状態変数 x は $0 \leq x \leq 6$ の範囲を 256 分割，t の定義域を 20 分割した．制御変数 u は $-2 \leq u \leq 2$ の範囲を 17 分割とした．以後のイテレーションでは，量子化を徐々に細かくし，繰り返し複合アルゴリズムの 11 回目のイテレーションでは x の定義域は 16,384 まで細分化した．制御量の分割はすべてのイテレーションを通じ 17 分割に固定した．

結果を表 2 に示す．この表で，繰り返し複合アルゴリズムのイテレーション 0 における数値は，基本形複合アルゴリズムによるコスト値を表している．このコスト値は，他の計算法よりも最適解の近似解に近い値を示している．繰り返し複合アルゴリズムの数回のイテレーションで，コスト値は急速に低下している．繰り返し複合アルゴリズムの収束性が速いことがわかる．得られたコスト値は共役傾斜法と最急降下法の中間に位置している．また，収束の速さは，共役傾斜法には及ばないが，最急降下法よりも速い．繰り返し複合アルゴリズムの 12 回の適用によって，コスト値の改善は見られなくなった．

もし，状態変数と制御変数の量子化の度合いを同じとし，従来の動的計画法を適用した場合には，最適解を得るまでに評価関数を 87,040 回評価し，最小コスト関数を 5,120 個計算する必要がある．

基本形複合アルゴリズムを適用した場合には，評価関数を 7,089 回評価し，最小コスト関数

表 2　連続時間最適制御問題における複合アルゴリズムと他の計算法のコストの収束

イテレーション	共役傾斜法	最急降下法	繰り返し複合アルゴリズム
0	123.4413	123.44	42.1478
1	41.7625	53.02	42.8410
2	41.6066	52.48	41.8374
3	41.5960	51.96	41.7423
4	41.5954	51.96	41.7191
5	41.5953	50.98	41.6538
6	41.5953	·	41.6526
7	41.5953	·	41.6175
8	41.5953	·	41.6170
9	41.5953	·	41.6138
10	41.5953	·	41.6135
11	41.5953	·	41.6127
12	41.5953	48.10	41.6127
·		·	
·		·	
50		41.64	

を 537 個計算することで最適解の近似解を得た．したがって，状態変数が 1 個の本例では，基本形複合アルゴリズムの計算量は従来の動的計画法よりも 1 桁小さい．

繰り返し複合アルゴリズムを適用した場合には，11 回目のイテレーションにおいて評価関数を 3,348 回評価し，最小コスト関数を 507 個計算することで表 2 の最適解の近似解を得た．コストの収束の様子を図 6（連続-1）に示す．もし，従来の動的計画法を繰り返し複合アルゴリズムの 11 回目のイテレーションと同程度の量子化の下で用いたとすると，繰り返し複合アルゴリズムの計算量は従来の動的計画法よりも 3 桁～ 4 桁小さい．

つぎに，限定操作によって削除された経路の割合を考察する．基本複合アルゴリズムでは，7,089 個の評価関数の評価を行い，その内 6,032 個が最適解の候補になることができず削除された．削除の割合は 85.1%であった．繰り返しアルゴリズムの 11 回目のイテレーションでは，3,347 個の評価関数の評価を行い，その内 2,839 個が最適解の候補になることができず削除された．削除の割合は 85.0%であった．従来の動的計画法は限定操作の概念を用いていないため，削除されるものはない．

他の 3 つの連続最適制御問題，すなわち

問題：連続-2[Bryson ら, 1964]

$$\text{Minimize} \qquad J = \frac{1}{2}\int_0^1 u(t)^2 dt \tag{24}$$

$$\text{Subject to} \qquad \dot{x}_1(t) = x_2(t) \quad t \in [0,1] \tag{25}$$

$$\dot{x}_2(t) = u(t) \tag{26}$$

$$x_1(t) \leq 0.2 \tag{27}$$

$$x_1(0) = x_1(1) = 0 \tag{28}$$

$$x_2(0) = -x_2(1) = 1 \tag{29}$$

問題：連続-3[Merriam, 1964]

$$\text{Minimize} \qquad J = \frac{1}{2}\int_0^5 (x_1(t)^2 + x_2(t)^2 + u(t)^2) dt \tag{30}$$

$$\text{Subject to} \qquad \dot{x}_1(t) = x_2(t) \quad t \in [0,5] \tag{31}$$

$$\dot{x}_2(t) = -x_1(t) + (1 - x_1(t)^2)x_2(t) + u(t) \tag{32}$$

$$x_1(0) = 1 \tag{33}$$

$$x_2(0) = 0 \tag{34}$$

問題：連続-4[Bryson ら, 1969]

$$\text{Minimize} \qquad J = t_f \tag{35}$$

$$\text{Subject to} \qquad \dot{x}_1(t) = (2gx_2(t))^{0.5}\cos u(t) \quad t \in [0, t_f] \tag{36}$$

$$\dot{x}_2(t) = (2gx_2(t))^{0.5}\sin u(t) \tag{37}$$

$$x_2(t) \leq x_1 \tan\theta + h \tag{38}$$

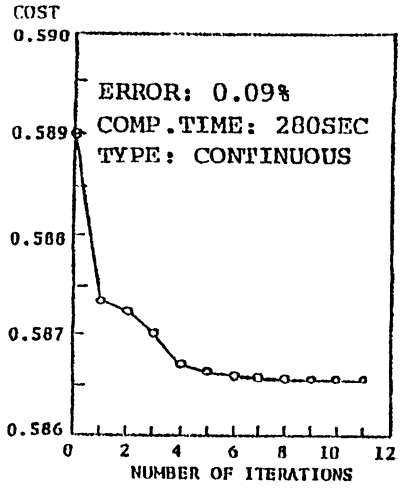
$$x_1(0) = 0 \tag{39}$$

$$x_2(0) = 0 \tag{40}$$

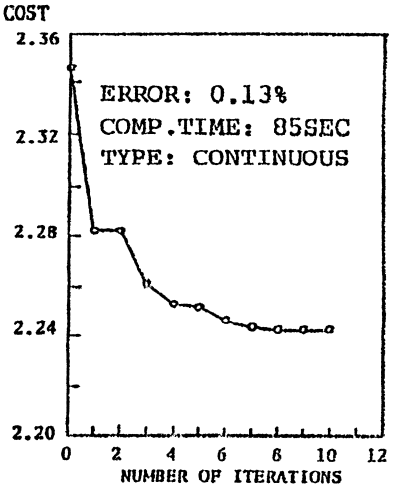
$$x_1(t_f) = 1 \tag{41}$$

$$\text{ただし，} t_f \text{ は終端時刻，} \theta = 26.6°, h = 0.1 \tag{42}$$

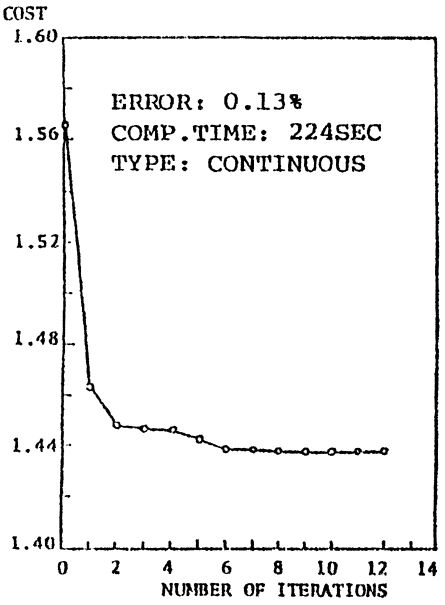
に対し，J を最小とする制御変数 $u(t)$ の値を求める問題を考える．これらの問題におけるコストの収束の様子を図 6（連続-2，連続-3，連続-4）に示す．いずれの数値例でも，数回のイテレーションで最適解の近似解に近くなり，10 回のイテレーションではコストの改善は見られなくなっている．



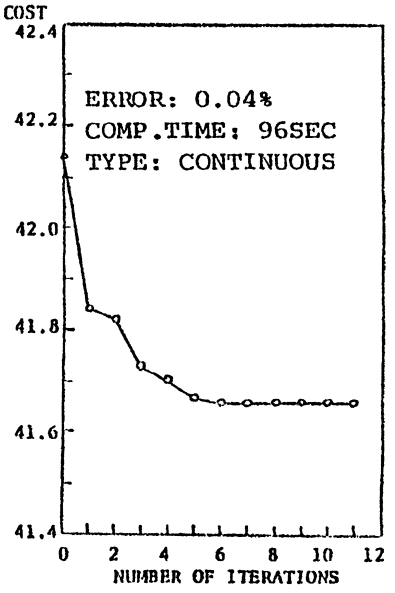
連続-1



連続-2



連続-3



連続-4

図６　４つの連続時間型最適制御問題におけるコストの収束

５．２　離散時間最適制御問題への適用例

高非線形性をもつ離散時間最適制御問題への適用結果を示す．この例題の状態方程式，評価関数，および境界条件は以下の通りである．

問題：離散-1[Noton, 1972]

$$\text{Minimize} \qquad J = \frac{0.5}{11}\sum_{k=0}^{9}(10x(k)^2 + u(k)^2) + (10 + \frac{5}{11})x(10)^2 \tag{43}$$

$$\text{Subject to} \qquad x(k+1) = 0.99x(k) + 0.5\tan u(k) \quad k \in [0, 9] \tag{44}$$

$$x(0) = 5 \tag{45}$$

この問題に対して，基本形複合アルゴリズムのイテレーション 0 の計算では，状態変数 x は $0 \leq x \leq 6$ の範囲を 256 分割した．段変数 k は問題より 10 分割であり，制御変数 u は $-3 \leq u \leq 1$ の範囲を 17 分割とした．以後のイテレーションでは量子化を徐々に細かくし，繰り返し複合アルゴリズムの 11 回目のイテレーションでは x の定義域は 16,384 まで細分化した．制御量の分割はすべてのイテレーションを通じ 17 分割に固定した．結果を表 3 に示す．表 3 で，繰り返し複合アルゴリズムのイテレーション 0 における数値は，基本形複合アルゴリズムによるコスト値を表している．このコスト値は，他の計算法よりも最適解の近似解に近い値を示している．繰り返し複合アルゴリズムの数回のイテレーションで，コスト値は急速に低下している．収束の速度は共役傾斜法や微分動的計画法よりも速くなっている．繰り返し複合アルゴリズムの 12 回の適用によって，コスト値の改善は見られなくなった．繰り返し複合アルゴリズムによって得られた最適解の近似解は共役傾斜法による最適解，あるいは微分動的計画法による最適解よりもよい解となっている．

基本形複合アルゴリズムを適用した場合には，評価関数を 23,970 回評価し，最小コスト関数を 1,529 個計算することで最適解の近似解を得た．

繰り返し複合アルゴリズムを適用した場合には，11 回目のイテレーションにおいて評価関数を 1,692 回評価し，最小コスト関数を 418 個計算することで表 3 の最適解の近似解を得た．この問題におけるコストの収束の様子を図 7（離散-1）に示す．もし，従来の動的計画法を繰り返し複合アルゴリズムの 11 回目のイテレーションと同程度の量子化の下で用いたとすると，繰り返し複合アルゴリズムの計算量は従来の動的計画法よりも 2 桁～3 桁小さい．すなわち，繰り返し複合アルゴリズムの計算量は従来の動的計画法よりも 2 桁～3 桁小さい．本例では，繰り返し複合アルゴリズムの 11 回目のイテレーションの計算量は基本複合アルゴリズムの計算量の 1/10 前後であった．

つぎに，限定操作によって削除された経路の割合を考察する．基本複合アルゴリズムでは，23,970 個の評価関数の評価を行い，その内 10,234 個が最適解の候補になることができず削除された．削除の割合は 42.7%であった．繰り返しアルゴリズムの 11 回目のイテレーションでは，1,692 個の評価関数の評価を行い，その内 1,269 個が最適解の候補になることができず削除された．削除の割合は 75.0%であった．

表 3　離散時間最適制御問題における複合アルゴリズムと他の計算法のコストの収束

イテレーション	共役傾斜法	微分動的計画法 (DDP)	繰り返し複合アルゴリズム
0	86.682	86.692	43.625
1	43.966	50.911	43.541
2	43.551	45.088	43.525
3	43.542	43.830	43.509
4	43.5426	43.670	43.509
5	43.5422	43.517	43.506
6	43.5422	43.506	43.503
7	43.4322	43.503	43.503
8	43.5421	43.502	43.5005
9	43.5421	43.501	43.5005
10	43.5421	43.501	43.5004
11	43.5421	43.501	43.5003
12	43.5420	43.5006	43.5003
.	.	.	
.	.	.	
20	43.5417	43.5004	

他の 3 つの離散最適制御問題，すなわち

問題：離散-2[Noton, 1972]

$$\text{Minimize} \qquad J = \frac{1}{11}\sum_{0}^{9}(x_1(k)^2 + u(k)^2) + x_1(10)^2 \tag{46}$$

$$\text{Subject to} \qquad x_1(k+1) = x_1(k) + 0.1x_2(k) \quad k \in [0,9] \tag{47}$$

$$x_2(k+1) = 1.14x_2(k) + 0.1(4u(k) - x_1(k) - 0.14x_2(k)^3) \tag{48}$$

$$x_1 = 0 \tag{49}$$

$$x_2 = -5 \tag{50}$$

問題：離散-3[Larson, 1968]

$$\text{Minimize} \qquad J = \sum_{k=0}^{4}(x_1(k)^2 + x_2(k)^2 + u_1(k)^2) + u_2(k)^2) \tag{51}$$

$$+2.5(x_1(5) - 2)^2 + 2.5(x_2(5) - 1)^2 \tag{52}$$

$$\text{Subject to} \quad x_1(k+1) = x_1(k) + x_2(k) + u_1(k) \quad k \in [0,4] \tag{53}$$

$$x_2(k+1) = x_2(k) + u_2(k) \tag{54}$$

$$x_1(0) = 2 \tag{55}$$

$$x_2(0) = 1 \tag{56}$$

$$0 \leq x_1(k) \leq 2 \tag{57}$$

$$-1 \leq x_2(k) \leq 1 \tag{58}$$

$$-1 \leq u_1(k) \leq 1 \tag{59}$$

$$-1 \leq u_2(k) \leq 1 \tag{60}$$

問題：離散-4[Larson, 1968]

$$\text{Minimize} \quad J = \sum_{k=0}^{4}(x_1(k)^2 + x_2(k)^2 + u_1(k)^2) + u_2(k)^2) + x_1(10)^2 \tag{61}$$

$$\text{Subject to} \quad x_1(k+1) = 0.625x_1(k) + 0.25x_2(k) + u_1(k) \quad k \in [0,4] \tag{62}$$

$$x_2(k+1) = -0.1875x_1(k) + 0.125x_2(k) + u_2(k) \tag{63}$$

$$x_1(0) = 5 \tag{64}$$

$$x_2(0) = 5 \tag{65}$$

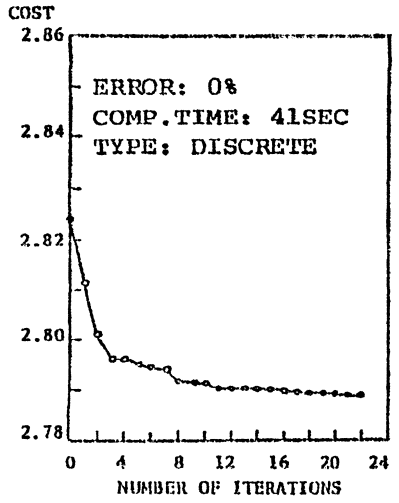
$$-5 \leq x_1(k) \leq 5 \tag{66}$$

$$-5 \leq x_2(k) \leq 5 \tag{67}$$

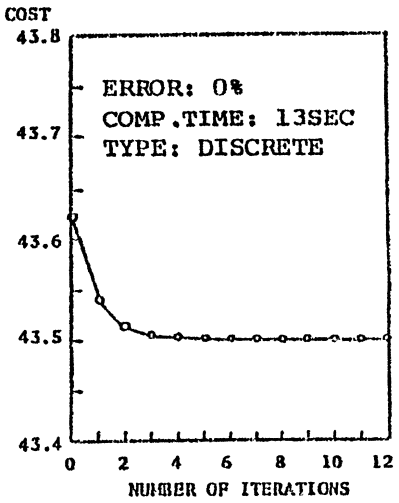
$$-2 \leq u_1(k) \leq 2 \tag{68}$$

$$-2 \leq u_2(k) \leq 2 \tag{69}$$

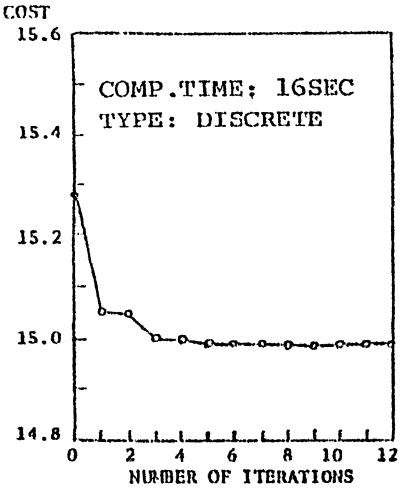
に対し，J を最小とする制御変数 $u(k)$ あるいは $u_1(k)$ と $u_2(k)$ の値を求める問題を考える．これらの問題におけるコストの収束の様子を図 7（離散-2，離散-3，離散-4）に示す．本数値例では 22 回の繰り返しでもコストは僅かではあるが下がり続けている．その他の数値例では，数回のイテレーションでコストの改善が見られなくなり，最適解の近似解に達していることがわかる．



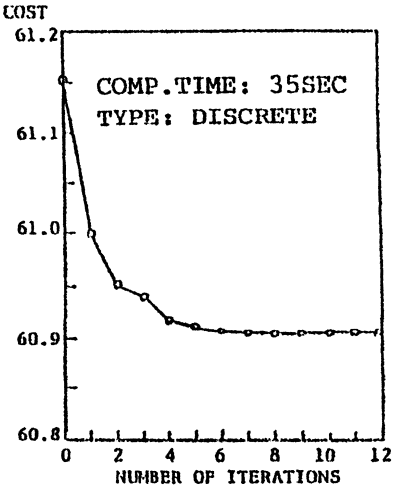
離散-1



離散-2



離散-3



離散-4

図７　４つの離散時間型最適制御問題におけるコストの収束

５．３　状態制約最適制御問題への適用例

繰り返し並列複合アルゴリズムによる解の精度とその収束性を微分動的計画法 (Differential Dynamic Programming, DDP と略記) [9] と比較するため，以下の，状態変数制約最適制御問題

$$\text{Minimize} \qquad J = \int_0^1 ((x_1(t))^2 + (x_2(t))^2 + 0.005u(t)^2)dt \tag{70}$$

$$\text{Subject to} \qquad \dot{x_1}(t) = x_2(t) \qquad x_1(0) = 0 \tag{71}$$

$$\dot{x_2}(t) = -x_2(t) + u(t) \quad x_2(0) = -1 \tag{72}$$

$$x_2(t) \leq 8(t - 0.5)^2 - 0.5 \tag{73}$$

を取り上げる．全てのイテレーションを通じ，独立変数 t の定義域を 20 分割し，段 k と段 k における t の値，すなわち t_k を割り付ける．初期点からの実行可能経路を正確に計算するため，状態遷移の計算とコスト関数の計算には，4 次のルンゲクッタ法を単精度で用い，ステップサイズを $\Delta t = 0.01$ とした．また，各区間 $t_k \leq t \leq t_{k+1}$ での制御入力を $u_k(t) = u(u_k^*, u_{k+1}, t)(u_{k+1} \in U)$ とし，この区間を直線近似した．ただし，u_k^* を，代表点での最適制御の値とする．

繰り返し並列複合アルゴリズムでの初期値となる初回 $(i = 1)$ の計算においては，初期解が劣悪の場合を調べるために粗い量子化を採用し，状態変数の定義域を 8 分割とした．一方，制御変数の定義域を 17 分割した．以後，イテレーションのたびに，量子化レベルを増加させ，最終のイテレーション $(i = 12)$ では，状態変数について 320 分割した．また，$i = 2$ 回目以降の制御変数のレンジを $\Delta u = 2.0$ $(i = 2)$ から 0.03125 $(i = 12)$ に徐々に狭めた．クリアランス $\Delta\epsilon_k$ は，全イテレーションを通じ，上界値，すなわち，粗い量子化の下での基本形複合アルゴリズムでの最適コストの 1% とした．コストの収束の様子を図 8 示す．コスト関数値は，数回のイテレーションで収束値付近に達し，0.2127 $(i = 1)$ から 0.1707 $(i = 12)$ へ収束した．$i = 12$ でのコストの変化は 1×10^{-6} 以下となった．一方，経路 x_2 と制御量 u の収束の様子を，それぞれ，図 9 と図 10 に示す．計算時間は 20MIPS の計算機で 168 sec であった．以後，計算時間の「秒」を sec で表す．

Ohno[9] は，この問題にニュートン法を用いた DDP を適用し，10 回のイテレーションで，コスト 0.1748 を得ている．

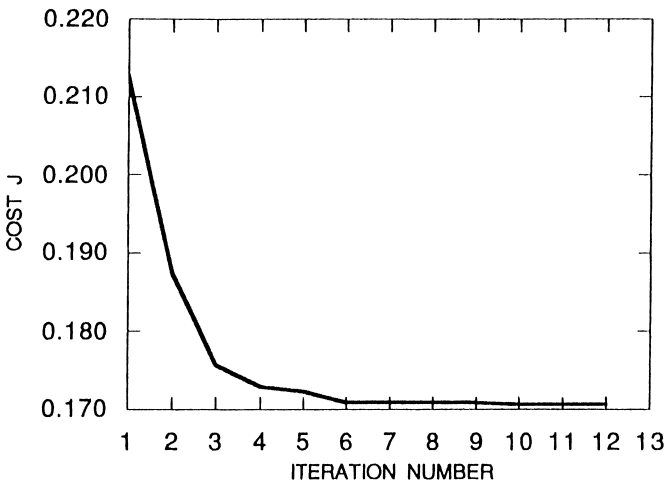


図 8　イテレーションに対するコストの収束

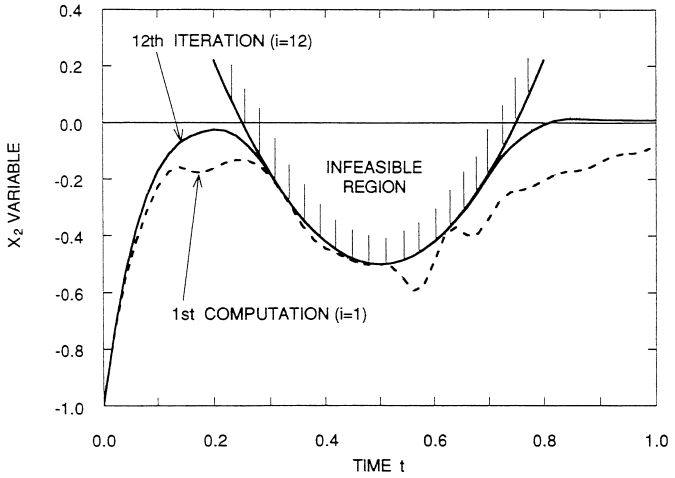


図 9　経路 x_2 の収束

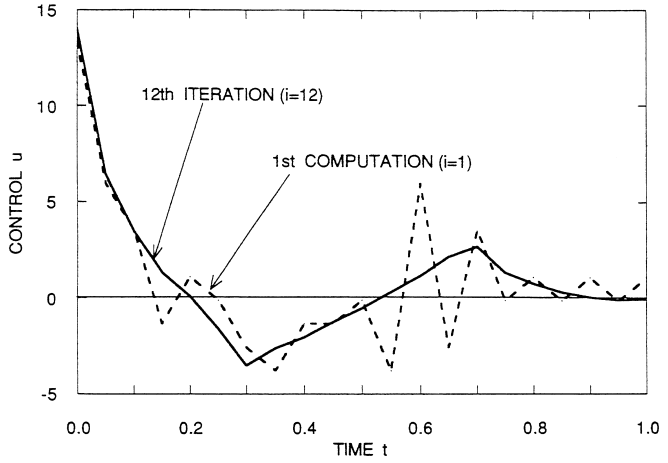


図 10　制御入力の収束

つぎに，繰り返し並列複合アルゴリズムによる解の精度を推定する．ただし本数値例の解析解を得ることができないため，ここでは解析解が得られる問題，すなわち，本数値例から (73) 式の制約条件を取り除いた問題に対して解の精度を推定した．この新たな問題の解析解では，最適値は $J = 0.06936$ となる．一方，繰り返し並列複合アルゴリズムでは，7 回のイテレーションで最適値は J=0.06945 の実行可能解を得た．この値は解析解からの誤差が 0.13% である．一方，制約条件付きの本数値例では，繰り返し並列複合アルゴリズムによる数値解のコスト $J = 0.1707$ は DDP での解のそれよりもより小さなコスト値を得ることができ，DDP による解に対するコストの改善率は 0.24%であった．

繰り返し複合アルゴリズムによるこれらの結果は，初期解が，たとえ劣悪であっても，最適

解に収束することを示している．この主要因として，以下の３つが挙げられる．

第１に，本法の第 i 回目の繰り返し計算では，前回 $(i-1)$ の最適経路の上下界値のみを参照し，経路そのものを直接参照していないこと．換言すれば，劣悪な初期解は参照経路としては使えないが，大域解を得るための計算領域の大きさ，すなわちクリアランスのサイズを決めることに利用できる．

第２に，本法でのコスト関数の計算と比較は内挿計算を用いずに，正確に，実行可能経路上に沿って行われるため，粗い量子化の下での解も，相対的に誤差が小さい最適解の候補経路を与えることである．

他方，DDP による解は局所解であり，また，解軌道付近の初期推定軌道を必要とする．さらに，DDP の微係数計算における計算負荷を考慮すると，繰り返し並列複合アルゴリズムは DDP との比較において，状態変数制約最適制御問題の解法に対して，優れた算法だと考えられる．

５．４　繰り返し複合アルゴリズムの特性

基本形複合アルゴリズムと繰り返し複合アルゴリズムの特徴は以下のようにまとめることができる．

- 基本形複合アルゴリズムによる初期解は他の計算法よりも良好な近似解を与える．その要因は，基本形が内挿計算を用いず，代表点の概念を用いていることに起因する．同様なことは，繰り返し複合アルゴリズムにおいても成立する．繰り返し複合アルゴリズムは，共役傾斜法や最急降下法，微分動的計画法に匹敵する精度で最適解の近似解を生成する．

- 基本形複合アルゴリズムの実行でも，下界値を必要とする．この下界値は適当な方法によって推定する必要がある．効果的な下界値ほど，推定の手間を必要とする．

- 基本形複合アルゴリズムの実行で得られた解は，繰り返し複合アルゴリズムの下界値として用いることが可能である．

- 基本形複合アルゴリズムの適用により，問題のコスト構造を抽出することができる．また，粗い量子化の下で大域的最適解の近似解を得ることができる．

- 従来の動的計画法に対する繰り返し複合アルゴリズムの計算量の割合（イテレーション 11 回）は 1/10,000（1 次元）～1/60,000（2 次元問題）である．一方，基本形では 1/2（1 次元問題）～1/40（2 次元問題）である．割合が異なる原因は，基本形と繰り返しの役割の違いにある．前者は問題のコスト構造を抽出するために用いられ，後者は下界値を精度よく推定するために用いられる．

- 計算量の削減の度合いは高次元の問題ほど大きい．一方，従来の動的計画法では，高次元の問題に対する初期解を得ることが難しい．故に，基本形複合アルゴリズムでは，高次元問題ほど削減の度合いが大きいことを利用して初期解をつくることができる．

- 効果的な下界値は繰り返し複合アルゴリズムの繰り返し適用によって生成できる．

これらを総合すると，基本形複合アルゴリズムだけで近似精度の高い最適解を得ることは難しい．精度の高い解を得るには，基本形複合アルゴリズムの実行で得られた解を初期値とし，繰り返し複合アルゴリズムを適用する必要がある．まず，基本形複合アルゴリズムの適用によって得られた解より下界値を算出し，その下界値を繰り返し複合アルゴリズムの実行で用いることにより計算量を削減する．

一方，繰り返し複合アルゴリズムを用いると，共役傾斜法や最急降下法，微分動的計画法に匹敵する解を得ることができる．これら従来の算法は，アルゴリズムの最初の計算で初期推定解と呼ばれる最適解の候補経路を必要とするため，境界条件が異なるたびに初期推定解を準備しなくてはならない[9)]．また，この初期推定解は最適解の近傍にとる必要がある．一方，繰り返し複合アルゴリズムは，初期推定解は不要であるため，境界条件を新たに設定し直した問題に対しても，アルゴリズムをシステマティックに適用することができる．

さらに，従来の方法は，状態変数の制約がある問題に対して，適用し難いことが知られている[9)]．一方，繰り返し複合アルゴリズムは，制約があるほど複合アルゴリズムの計算量が少なくてすむ可能性が高い．

これらの考察により，繰り返し複合アルゴリズムは

- 状態変数制約付き最適制御問題

- 時間制約付き最適制御問題（時間点制約，時間枠制約）

- 高度の非線形性をもつシステムの最適化

に対して有効であると考えられる．このような問題の応用分野としては，

- 低推力宇宙機の軌道計画問題への応用

- エネルギー近似を用いた飛行体の最適制御上昇問題への応用

- 飛行体の再突入最適軌道生成への応用

- スペースプレーン最適経路問題への応用

- 制約のある航空路の設計問題への応用

表 4　最適制御問題に対する計算法の比較

評価項目	従来の手法		複合アルゴリズム	
	最大原理	動的計画法	基本形	繰り返し
必要計算機メモリー	○	×	△	△
計算量・計算時間	○	×	△	△
解の精度	○	×	△	○
計算実行の安定性	△	○	○	○
制約条件の取り扱い	△	○	○	○
大域的最適解	×	○	○	△

○：要求を満たす，△：問題点は工夫次第で克服可能，×：要求を満たさない

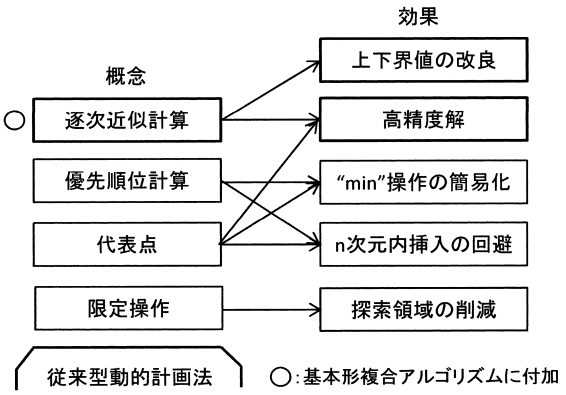


図 11　繰り返し複合アルゴリズムの概念と効果の関係

- 非線形制御系の設計への応用

等があげられる．

種々の最適制御問題に繰り返し複合アルゴリズムを適用した結果，表 4 に示す評価が可能となる．この表より，繰り返し複合アルゴリズムは，計算量や必要計算機メモリーにおいて最大原理によるアプローチには及ばないが，従来の動的計画法の「Bellman の次元の呪い」の問題を工夫次第で克服可能であることがわかる．解の精度の観点においては最大原理によるアプローチと同程度の性能を示している．一方，大域的最適解に関しては，動的計画法からのアプローチである基本形複合アルゴリズムが優れている．基本形複合アルゴリズムは，最適制御問題のコスト構造を抽出するために繰り返し複合アルゴリズムの初回の計算ルーティンの中に組み込まれている．

図 11 は，繰り返し複合アルゴリズムの 4 つの概念と，それらの概念によって得られた効果の関係である．これらの概念は従来の動的計画法に付加された概念であり，これらの概念の導入によって動的計画法の計算数と解の精度は極めて改善されることが判明した．

6　まとめ

最適制御問題を解く基本形複合アルゴリズムおよび繰り返し複合アルゴリズムの算法を提案した．提案したアルゴリズムは動的計画法の長所を継承し，動的計画法の短所である計算量を削減している．そのために必要となる最適解の下界値と上界値はアルゴリズム自身で生成する．基本形複合アルゴリズムは対象問題の全体のコスト構造を抽出し，大域的最適解の近似解を得ることができる．一方，繰り返し複合アルゴリズムは，自身でコストの下界値と上界値を精度よく推定することによって数値計算上の計算量やメモリー容量のサイズといった負荷を大幅に削減することが可能である．解の精度は共役傾斜法や微分動的計画法に匹敵する．また，基本形複合アルゴリズムは最適解の近似解に近い精度の解を生成する．繰り返し複合アルゴリズムの収束は速く，多くの事例では数回程度で，最適解の近似解に至る．

このような特性を持つ基本形/繰り返し複合アルゴリズムは，複雑な制約条件下で，最適経路をシステマティックに生成することが可能である．

参考文献

1) R.E. Bellman: *Dynamic Programming*, Princeton University Press, Princeton, N.J. (1957)

2) O.G. Alekseev and I.F. Volodos: “Combined Use of Dynamic Programming and Branch-and-Bound Methods in Discrete-Programming Problems”, *Automation and Remote Control*, **37**, 557/565 (1976)

3) T.L. Morin and R.E. Marsten: “Branch-and-Bound Strategies for Dynamic Programming”, *Opns. Res.*, **24**, 611/627 (1976)

4) T. Hanaoka and T. Tanabe: “A New Dynamic Programming Algorithm and its Application to Optimal Reentry Problems”, *Proc. of 13th Int'l Symp. Space Tech. Sci.*, 1031/1036 (1982)

5) T. Hanaoka: “A Lower Bound Computation Method Using Energy-State Approximation and Its Application to Supersonic Aircraft Shortest Path Problems”, *Journal of the Operations Research of Japan*, **41**-2, 289/310 (1998)

6) 花岡照明: “エネルギー近似を用いた下界値計算法と超音速航空機最短経路問題への応用”, *Journal of the Operations Research Society of Japan*, Vol.41, No.2 (1998)

7) R.E. Larson: *State Increment Dynamic Programming*, American Elsevior Publishing Company, New York (1968)

8) M. Athans and P.L. Falb: *Optimal Control*, McGraw-Hill, New York (1966)

9) K. Ohno: “A New Approach to Differential Dynamic Programming for Discrete Systems”, *IEEE Trans. Automat. Contr.*, AC-**23**, 37/47 (1978)