動的計画法複合アルゴリズムを用いた
非線形制御系の設計法とその応用

Nonlinear Control System Design by Hybrid Dynamic Programming Algorithm and Its Applications

花　岡　照　明
Teruaki HANAOKA

跡見学園女子大学マネジメント学部紀要　第 7 号　(2009 年 3 月 15 日)

跡見学園女子大学マネジメント学部紀要　第７号　（2009 年３月 15 日）

動的計画法複合アルゴリズムを用いた非線形制御系の設計法とその応用

Nonlinear Control System Design by Hybrid Dynamic Programming Algorithm and Its Applications

花　岡　照　明

要　旨

本論文では，先に開発した動的計画法複合アルゴリズムを用いて時間領域における非線形制御系の新しい設計法を提案する．この手法はプラントのアクチュエータあるいはプラント自身に種々の非線形要素を含む非線形制御系に対し，最小時間制御，最小燃料消費制御，あるいは，より汎用的な評価基準のもとで試行錯誤を含まない設計手段を与える．本法によって得られる表形式の制御器は１入力-1 出力系だけでなく多入力- 多出力系の非線形制御システムに適用可能である．この制御器をプラントのアナログシミュレーションに組み入れて試験した結果，極めて効果的に機能することが確かめられた．この方法は，従来の記述関数設計法の適用上の難しさを克服する可能性を示唆している．

キーワード：非線形制御系，動的計画法，最適レギュレータ，状態空間法

1.　はじめに

非線形性には，望ましくない非線形性の補償や線形要素を用いるよりも更に良い性能を得るために故意にシステムの中に組み込まれる場合がある．ここではプラントのアクチュエータあるいはプラント自身，またはその双方に非線形要素を含む非線形制御系の設計問題を考察する．この問題の取り扱いは大変複雑であり，従来，設計には多大な時間を必要とした．多くは試行錯誤による繰り返し手順を含む．長い間，非線形制御系の設計問題には記述関数法が用いられてきた．この方法は非線形系のリミットサイクルの解析には有効であるが，古典的な線形制御理論における過渡応答に対しては余り有効ではない．記述関数法は前もって基本信号の形を仮定する必要がある．この方法の使用上の限界は，満足な精度が得られないこと，および多入力-多出力系への拡張が難しいとされることである．

これらの難しさを克服する効果的な方法は，実際の性能を測定できるように与えた評価基準の下で最適制御問題を解くことである．しかしながら，最適制御問題を解くための基本的な２つの手法の一方である最小原理は完全な理論であるが，バックラッシュやヒステリシス等の要素をもつ線形または非線形プラントのような複雑なシステムに対しては計算実行上問題がある．他方，動的計画法は，基本的にはどのような複雑さを持つ問題に対しても数値的に取り扱う手段を与えるほか，大域的最適解やクローズドフォーム解，制約条件や非線形性の取り扱いの容易さ，安定した計算実行が可能なこと，状態遷移とコスト関数の評価に対し表関数を許すなど，数多くの長所を持つ．このような多くの長所を持つ動的計画法は，しかしながら，状態変数の次数が増加すると計算数が指数関数的に増大するという，いわゆる，Bellman（1957）の“次元の呪い”の問題を持つこと，およびコスト関数の多次元空間での煩わしい内挿計算を必要とする問題を持っている．

これらの問題を克服する種々の方法が開発されてきた．Alekseev（1976），Morin and Marsten（1976）は離散動的計画法で最適解の一部になることができない状態を削除するために分枝限定手法を用いた．連続変数問題に対する指数関数的な必要計算数を減らす有望な方法はLarson（1968）によって扱われたstate increment dynamic programmingとJacobson and Mayne（1970）によって与えられたdifferential dynamic programmingである．Hanaoka and Tanabe（1982）では地球周回軌道の揚力体を地球に帰還させる際に加速度と空力加熱を最小化する実際の６変数再突入最適軌道問題に対し分枝限定操作を併用した前向き動的計画法アルゴリズム（基本形複合アルゴリズム）を適用し，その有効性を示した．Hanaoka（1989）では，低推力で航行する太陽観測衛星を地球周回軌道上から，途上で金星重力エネルギーをスイングバイによって獲得しながら最短時間で太陽近傍に到達させる最適低推力軌道生成問題に分枝限定操作を併用した繰り返し複合アルゴリズムを適用し，その有効性を示している．その中で用いられた複合アルゴリズムは基本形複合アルゴリズムの繰り返し適用によって，最適解を求めるまでの計算数を削減し，かつ最適解の精度を逐次向上させている．基本形複合アルゴリズムは，分枝限定操作の概念，コスト関数評価の優先順位計算の概念，コスト関数の代表点の概念の３つの概念に基づいて構成されている．他方，繰り返し複合アルゴリズムでは，上述の３つの概念に加え，新たに逐次近似の概念が導入されている．以後，基本形複合アルゴリズムと繰り返し複合アルゴリズムを２つ併せた呼称を複合アルゴリズムと呼ぶことにする．

複合アルゴリズムは連続変数最適制御問題に従来の動的計画法を適用した際に起こる”次元の呪い”の問題を回避している．この有望なアルゴリズムの効果は約30個の数値例によって支持されている（Hanaoka and Tanabe（1982），Hanaoka（1989），他）．典型的な３変数問題に本法を適用した数値例では，コスト関数の計算数は従来の動的計画法の概算1/1,000以下である．削減比率は変数の次数が高いほど大きい．非線形制御系の設計では，目標点または目標領

域から基本形複合アルゴリズムをバックワードに適用することによって最適コスト関数の等コスト面（超平面）を順次作成し最適制御信号と解軌道を得ている．この手法は，非線形性を含むシステムに対し難しくなく統一的に適用することができる．

この論文の目的は，制御に制約を持ち，プラントのアクチュエータあるいはプラント自身に種々の非線形要素を含む非線形制御系に対し，動的計画法複合アルゴリズムを適用することによって，最小時間制御，最小燃料消費制御，あるいは，より汎用的な評価基準のもとで試行錯誤を含まない設計手段を提案することであり，さらに本法によって作成された制御器の有効性を明らかにすることである．この論文では，動的計画法複合アルゴリズムによる新しいタイプの最適レギュレータが詳細に調べられている．

2. 最適制御問題と動的計画法

いま，最も一般的な離散時間システムは，

$$\boldsymbol{x}_{k+1} = \boldsymbol{g}_k(\boldsymbol{x}_k, \boldsymbol{u}_k) \qquad (k = 0, 1, 2, ..., N) \tag{1}$$

で記述されるものとする．ここで，x_k と u_k を，それぞれ，n 次元状態ベクトルおよび m 次元制御ベクトルとする．また，k を段変数の添え字，g_k を n 次元ベクトル値関数とする．このとき，最小化したい評価関数を

$$J = \sum_{i=0}^{N-1} L_i(\boldsymbol{x}_i, \boldsymbol{u}_i) + \Phi_N(\boldsymbol{x}_N) \tag{2}$$

とすれば，最適制御問題は

$$\text{Minimize} \quad J = \sum_{i=0}^{N-1} L_i(\boldsymbol{x}_i, \boldsymbol{u}_i) + \Phi_N(\boldsymbol{x}_N) \tag{3}$$

$$\text{Subject to} \quad \boldsymbol{x}_{k+1} = \boldsymbol{g}_k(\boldsymbol{x}_k, \boldsymbol{u}_k)(k = 0, ..., N-1) \tag{4}$$

$$\boldsymbol{x}_0 = \boldsymbol{c}_0 \tag{5}$$

$$\boldsymbol{x}_N \in \Omega_F \tag{6}$$

$$\boldsymbol{x}_k \in X_k \quad (k = 1, ..., N-1) \tag{7}$$

$$\boldsymbol{u}_k \in U(\boldsymbol{x}_k) \quad (k = 0, ..., N-1) \tag{8}$$

となる．ただし，$L_k(x_k, u_k)$（k = 0,1, ..., N − 1）を 1 段当りのコスト関数，$\Phi_N(x_N)$ を最終段のコスト関数とする．また，X_k と $U(x_k)$ を，それぞれ k 段での許容状態集合と許容制御集合とする．(5) と (6) 式は，それぞれ，初期条件と終端条件であり，$\Omega_F(\Omega_F \subset X_N)$ を終端条件を満たす集合とする．

上記の問題を従来型（後向き）動的計画法[1)]を用いて解くには，各段 $k(k$ = 0,1, ..., $N)$ での

すべての許容状態 $x_k \in X_k$ に対して動的計画法の再帰関数方程式を適用し，それらの状態 x_k の最適制御 u_k^* を計算する．この再帰関数方程式は，k 段以後の最小コスト関数を

$$J_k(\boldsymbol{x}_k) = \min_{\boldsymbol{u}_k, \boldsymbol{u}_{k+1}, ..., \boldsymbol{u}_{N-1}} \{ \sum_{i=k}^{N-1} L_i(\boldsymbol{x}_i, \boldsymbol{u}_i) + \Phi_N(\boldsymbol{x}_N) \}$$

と定義すると，Bellman の最適性の原理[1] から，

$$\begin{aligned} J_N(\boldsymbol{x}_N) &= \Phi_N(\boldsymbol{x}_N) \\ J_k(\boldsymbol{x}_k) &= \min_{\boldsymbol{u}_k \in U(\boldsymbol{x}_k)} \{ L_k(\boldsymbol{x}_k, \boldsymbol{u}_k) + J_{k+1}(\boldsymbol{g}_k(\boldsymbol{x}_k, \boldsymbol{u}_k)) \} \\ & \qquad (k = 0, ..., N-1) \end{aligned} \tag{9}$$

となる．この (9) 式から最適方策を解析的に求めることは多くの場合難しいので，通常，数値的に解くことになる．

(9)式を解くには，通常，各状態変数 $x_k^i (i = 1, 2, ..., n)$ を l_x レベルに，また，各制御変数 $u_k^j (j = 1, 2, ..., n)$ を l_u レベルに量子化する．そして，状態ベクトルの n 次元格子の各点において量子化した制御を適用し，(9)式を用いて格子点での $J_k(x_k)$ と最適制御 u_k^* を求める．もし，この計算で，次段の状態 $g_k(x_k, u_k)$ が格子点上に無い場合は，格子点での $J_{k+1}(x_{k+1})$ を用い，内挿によって $J_{k+1}(g_k(x_k, u_k))$ を計算する．状態変数の次元数 n が大きいとき，この内挿計算は相当煩わしい．(9) 式の計算は $k = N - 1$ から始め，後向きに，$k = 0$ まで再帰的に行う．

従来型動的計画法を適用する上で最も難しいことは，状態変数 x_k の次元数の増加に伴ってメモリーサイズ (データの格納領域) と計算数が飛躍的に増大することである．これは，状態ベルトルを量子化することに起因する．たとえば，問題の定義域の次元数が $n = 5$ で，段数が $N = 10$ の問題に対し，x_k の各要素当り $l_x = 20$ レベルに分割したとき，(9) 式の右辺を評価するために計算機メモリーに格納する必要がある $J_{k+1}(x_{k+1})$ の格子点の数は $l_x^n = 20^5 = 3{,}200{,}000$ 個である．また，$J_k(x_k)$ と u_k^* の計算を必要とする格子点の総数 (計算領域) は $l_x^n \cdot N = 20^5 \times 10 = 32{,}000{,}000$ 個となる．量子化レベルをさらに増やしたとき，従来型動的計画法によるアプローチでは事実上，計算が実行不可能に陥る．

3.　基本形複合アルゴリズム

従来型動的計画法は，状態の遷移方向 (通常は，時間の増加方向) に関し，後向きに定式化している．これは，「後向き」最適性の原理[1] を用いたことによる．一方，基本形複合アルゴリズムで用いる前向き動的計画法は，この原理を，時間に関して前後を逆にした「前向き」最適性の原理に基づき，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 $u_k \in U(x_k)$ を初期点 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集合を考え，$\{X_k^\circ\} = \cup_{i=0}^k X_i^\circ$ とおく．ただし，X_k° は，再帰的に定義さ

れ，$X_0^\circ = \{x_0\}, X_{k+1}^\circ = \{x_{k+1} \mid x_{k+1} = g_k(x_k, u_k), x_k \in X_k^\circ, u_k \in U(x_k)\}$ $(k = 0, ..., N-1)$ とする．また，g_k を状態遷移関数とする．この経路群は初期点からの枝構造をもつ．このような，状態点の決め方，すなわち，k 段の状態点 $x_k \in X_k^\circ$, を押し出すことによって，$k+1$ 段の状態点 $x_{k+1} \in X_{k+1}^\circ$ を確定させる計算を，「押し出し計算」と呼ぶことにする．ここで，計算上の工夫として，前向き動的計画法でのすべての計算点を，押し出し計算によって，この実行可能経路 $\{X_k^\circ\}$ 上にとり，かつ，コスト関数の計算と比較を，正確にこの経路上のコスト値を用いて行うことを考える．このような処理では，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．この巨大化を避けるため，次節では，量子化について考察する．

3.1 量子化手続きと前向き動的計画法

基本形複合アルゴリズムでは，前向き動的計画法を適用するのに，許容状態集合 X_k を，$X_k = \cup_{i=1}^{n_k} X_{ki}$, $X_{ki} \cap X_{kj} = \Phi(i \neq j)$ となるように，適当な部分集合 $X_{k1}, X_{k2}, ..., X_{kn_k}$ に量子化する．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，初期点 x_0 からの経路が存在する場合のみ，一個ずつ代表点 $x_{ki} (\in X_{ki})$ とそのコスト関数の値を以下のように定義する．すなわち，

$$\begin{aligned} f_k(\boldsymbol{x}_{ki}) &= \min_{\boldsymbol{x}_k}\{T(\boldsymbol{x}_k) \mid \boldsymbol{x}_k \in X_{ki}\} \\ &\quad (i = 1, ..., m_k,\ k = 0, ..., N) \end{aligned} \tag{10}$$

とする．ここで，$T(x_k)$ は，前段の代表点 $x_{k-1i} (x_{k-1i} \in X_{k-1i})$ に，量子化した各 $u_{k-1} \in U(x_{k-1i})$ を適用し，式，

$$\begin{aligned} &T(\boldsymbol{x}_0) = 0 \\ &T(\boldsymbol{x}_k) = f_{k-1}(\boldsymbol{x}_{k-1i}) + L_{k-1}(\boldsymbol{x}_{k-1i}, \boldsymbol{u}_{k-1}) \\ &\boldsymbol{x}_k = \boldsymbol{g}_{k-1}(\boldsymbol{x}_{k-1i}, \boldsymbol{u}_{k-1}) \\ &\qquad (i = 1, ..., m_k, k = 1, ..., N) \end{aligned} \tag{11}$$

を用い，押し出し計算によって計算する．ここで，L_k を 1 段当りのコスト関数とする．ただし，便利さのため，最終段 N でのコスト $\Phi_N(x_N)$ を，$N-1$ 段でのコスト L_{N-1} に含めて考える．また，$m_k (m_k \leq n_k)$ を，k 段の代表点の数とする．この代表点 x_{ki} が，初期点 $x_0 (x_0 = x_{01})$ から，k 段の各ブロック上での到達点までの経路の中で，対応するコスト関数の値が（各ブロックの中で）最小となる点であり，初期点から再帰的に計算できる．このような代表点におけるコスト関数の最小値を最小コスト関数と呼び，代表点 x_k の関数として $f_k(x_k)$ とおく．

このとき，有限個の代表点 $X_k^\circ = \{x_{k1}, x_{k2}, ..., x_{km_k}\}$ $(k = 0, ..., N)$ 上で考慮した前向き動的計画法が成立する．すなわち，

$$
\begin{aligned}
&f_0(\boldsymbol{x}_{01}) = 0 \\
&f_{k+1}(\boldsymbol{x}_{k+1i}) = \min_{\boldsymbol{x}_{ki},\boldsymbol{u}_k} \{f_k(\boldsymbol{x}_{ki}) + L_k(\boldsymbol{x}_{ki},\boldsymbol{u}_k) \mid \\
&\qquad\qquad \boldsymbol{x}_{ki} \in X_k^{\circ}, \boldsymbol{u}_k \in U(\boldsymbol{x}_{ki})\} \quad (i = 1, ...m_k) \\
&\boldsymbol{x}_{k+1i} = \boldsymbol{g}_k(\boldsymbol{x}_{ki},\boldsymbol{u}_k) \qquad (k = 0, ..., N-1)
\end{aligned} \tag{12}
$$

とする．

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の値は，代表点の値で近似し，

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

とする．

3.2　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，量子化した制御を適用し，枝構造をもった最適経路候補群を繰り返し生成させる．この際，(11)式によって計算されるコスト関数値 $T(x_k)$ 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とにする．このようにすると，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．それ以後に到着するいかなる経路も代表点となることはできないから，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

$$
f^*_{0,N} = \min_{\boldsymbol{x}_N}\{f_N(\boldsymbol{x}_N) \mid \boldsymbol{x}_N \in \Omega_F\} \tag{13}
$$

と定義することにする．

3.3　限定操作とクリアランス

基本形複合アルゴリズムでは，前向き動的計画法の計算数を削減するために，分枝限定法の限定操作を使う．

いま，(9) 式の最小コスト関数 $J_k(x_k)$ の下界値を $M_k(x_k)$ とし，また，原問題 (3) ～ (8) 式下での最終最適解 $f^*_{0,N}$(13) 式の上界値を I(暫定解) とする．これらの下界値と上界値が満たすべき条件は，それぞれ

$$
\begin{aligned}
M_k(\boldsymbol{x}_k) &\leq J_k(\boldsymbol{x}_k), \qquad \boldsymbol{x}_k \in X_k \\
&= \min_{\boldsymbol{u}_k,\boldsymbol{u}_{k+1},...,\boldsymbol{u}_{N-1}} \{\sum_{i=k}^{N-1} L_i(\boldsymbol{x}_i,\boldsymbol{u}_i) + \Phi_N(\boldsymbol{x}_N)\} \\
&\qquad\qquad (k = 0, ..., N-1)
\end{aligned} \tag{14}
$$

$$
I \geq f^*_{0,N} \tag{15}
$$

で与えられる．よく知られているように，もし，

$$f_k(\boldsymbol{x}_k) + M_k(\boldsymbol{x}_k) > I \qquad (k = 0, ..., N) \tag{16}$$

ならば，状態 x_k を通る経路について，

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

が成り立つ. よって, 代表点 x_k は最適経路の一部になれないから削除してよい. ただし, $f_0(x_0)$ は，(11) 式より $f_0(x_0) = T(x_0) = 0$ である．また，ここでは，最終段 N でのコスト $\Phi_N(x_N)$ を，$N-1$ 段でのコスト L_{N-1} に含めて考えたから，形式的に，$J_N(x_N) = 0$ とする．したがって，$M_N(x_N) = 0$ とおく．削除できる代表点 x_k の数を増やすには，(16) 式から明らかなように, 上界値 I をより小さな値へ，一方，下界値 $M_k(x_k)$ をより大きな値へと強化すればよい．ここで，上界値と下界値の強さを表わす量として，それぞれ，Δa および Δb_k を導入し，

$$\Delta a = I - f^*_{0,N}, \quad \Delta b_k = J_k(\boldsymbol{x}_k) - M_k(\boldsymbol{x}_k) \tag{17}$$

とおく．(17) 式の I と $M_k(x_k)$ を (16) 式に代入すると

$$f_k(\boldsymbol{x}_k) + J_k(\boldsymbol{x}_k) > f^*_{0,N} + \Delta a + \Delta b_k \tag{18}$$

となる．(18) 式より，基本形複合アルゴリズムの計算数は，Δa や Δb_k の個別の値というよりも，それらの和 $\Delta\epsilon$ ($\Delta\epsilon = \Delta a + \Delta b_k$) に依存する．以後，この和 $\Delta\epsilon$ をクリアランスと呼ぶことにする．基本形複合アルゴリズムを用いて大域解を得るには，(14) ,(15) 式の制約を満たす有効な下界値と上界値を計算できなくても，より緩和された，大域解を得るためのクリアランスの条件，すなわち，

$$\Delta\epsilon = \Delta a + \Delta b_k \geq 0 \tag{19}$$

を満足すればよい．ここで，この複合アルゴリズムの計算数が最小となるのは，$\Delta\epsilon = \Delta a + \Delta b_k = 0$ のときである．

3.4 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 $x_0 = c_0$ と終端条件 $x_N \in \Omega_F$ 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 $X_{k1}, X_{k2}, ..., X_{kn_k}$ に分割する（ただし，$X_k = \cup_{i=1}^{n_k} X_{ki}$, $X_{ki} \cap X_{kj} = \Phi (i \neq j)$）．

3.(クリアランスの設定)：各 $k = 0, ..., N$ に対し , 適当な $x_k \in X_k$ に対する下界値 $M_k(x_k)$

を計算する．また，一つの上界値Ｉを設定する．（ただし，有効な下界値を計算できないときは，$M_k(x_k) = 0$ とし，I を $\Delta\epsilon \geq 0$ となるように設定する）．

4. (初期化)：$\Omega \leftarrow \{x_0\}, T(x_0) = 0, \Re \leftarrow \emptyset$を行う（$\Re$ 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，また，←は代入操作を表わす）．
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 $T(x^*)$ の値が最小である x^*を選び，$\Omega \leftarrow \Omega \backslash \{x^*\}$ とせよ．そして，x^* が属する段の値を k にセットし，$x_k^* \leftarrow x^*$ とせよ（$\backslash$ は差集合の演算子）．
7. (終端テスト)：もし x_k^* が終端条件 $x_N \in \Omega_F$ 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$\Re \leftarrow \Re \cup \{[x_k^*]\}$ とし，ステップ９へ行け（[y] は状態 y を含むブロックを示す．この段階で，x_k^* に到達させる最適制御 u_{k-1}^* が確定する．また，最小コスト関数の候補値 $T(x_k^*)$ は，真の最小コスト関数値 $f_k(x_k^*)$ となる．）
9. (分枝限定操作)：量子化した各 $u_k \in U(x_k^*)$ に対して，次段の状態 x_{k+1} および最小コスト関数の候補値 $T(x_{k+1})$ を，$x_{k+1} = g_k(x_k^*, u_k), T(x_{k+1}) = T(x_k^*) + L_k(x_k^*, u_k)$ とする．そして，もし各 x_{k+1} に対し，$T(x_{k+1}) + M_{k+1}(x_{k+1}) \leq I$ ならば，$\Omega \leftarrow \Omega \cup \{x_{k+1}\}$ とせよ．
10. ステップ５へ行け．

ステップ６で x^*を選択するとき，x をそれらに対応する最小コスト関数の候補値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．

3.5　繰り返し複合アルゴリズム

本節で述べる繰り返し複合アルゴリズムは，上述の基本形複合アルゴリズムを改良したものである．(16) 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（$i = 2, 3, ...$）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

$$u_{kmin}^{l,i} \leq u_k^{l,i} \leq u_{kmax}^{l,i} \tag{20}$$

ここで，

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

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

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 $M_k^i(x_k)(x_k \in X_k)$ 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，

$$I^i = f*_{0,N}^{i-1} \quad (i=2,3,...) \tag{21}$$

$$\begin{aligned} M_k^i(\boldsymbol{x}_k) &= J_k^{i-1}(\boldsymbol{x}_k^{*i-1}) \\ &= \sum_{l=k}^{N-1} L_l(\boldsymbol{x}_l^{*i-1}, \boldsymbol{u}_l^{*i-1}) + \Phi_N(\boldsymbol{x}_N^{*i-1}) \\ &\quad (k=0,1,...,N-1,\ i=2,3,...) \end{aligned} \tag{22}$$

によって強化する．ここで，x_l^{*i-1} と 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．(21) 式による I^i は良好な上界値を与える．また，(22) 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．しかしながら，最適経路の近傍以外では，(22) 式は大域解を得るための下界の必要条件，すなわち，(14) 式の $M_k^i(x_k) \leq J_k(x_k)(k=0,...,N-1)$ を常に満たすとは限らない．また，この判定には $J_k(x_k)$ の評価を必要とするが，正確な $J_k(x_k)$ の値を知ることは難しい．そこで，次節では，(19) 式の，大域解を得るためのクリアランスの条件 $\Delta\epsilon \geq 0$ を用いて，大域解を得ることを考える．

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

$$|f*_{0,N}^{i} - f*_{0,N}^{i-1}| \ll \epsilon_1 \tag{23}$$

で与える．ただし，$\epsilon_1 \ll 1$ とする．

一般に，状態量や制御量の量子化単位が粗い時点では，経路群が終端条件を満たせなくなって，計算が停止してしまう可能性がある．この現象に対処するため，終端条件が満たされないほど終端段のコスト値が大きくなるようなペナルティー関数 $P(x_N)$ を導入する．そして，元の評価関数にこのペナルティー関数項を加えた新しい評価関数

$$J = \sum_{i=0}^{N-1} L_i(\boldsymbol{x}_i, \boldsymbol{u}_i) + \mu P(\boldsymbol{x}_N) \tag{24}$$

$$\text{ここで，} P(\boldsymbol{x}_N) = \Psi(\boldsymbol{x}_N)^T \Psi(\boldsymbol{x}_N) \tag{25}$$

の下で，終端条件のない最小化問題を解く．ここで，$\Psi(x_N) = 0$ は原問題の終端条件であり，μ は十分大きな正定数とする．このとき，量子化単位を細かくするほど，したがって，イテレーション回数の増加とともに，最適経路は終端条件を満たすように改善される．

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．

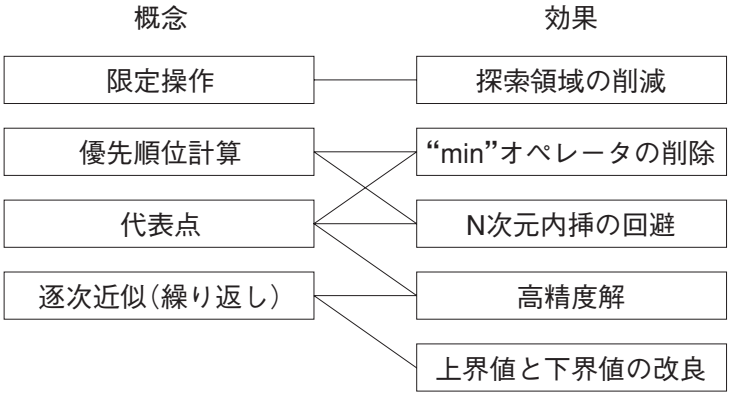

図１：４つの概念と効果

図１は複合アルゴリズムで用いられた４つの概念と効果の間の関係を示している．両者間の実線がそれらの概念と効果の対応である．

4.　非線形制御系設計への応用

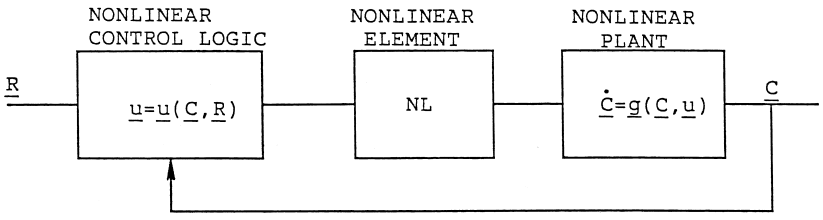

図２：非線形制御系

ここで取り扱う基本システムは，図２に示されたようなプラントのアクチュエータあるいはプラント自身に種々の非線形要素を含む非線形制御系である．目的は，ある精度以内に，出力信号 c を与えられた参照信号レベル r になるように，制御入力 u を決定することである．現在の状態の関数として制御入力を得るために，基本形複合アルゴリズムを参照信号レベルに対応

する状態点(終端点)から状態遷移関数の逆写像として許容制御入力を用い時間に関して正確にバックワードに，区分された時間の 1 区間ずつ適用する．このようにすると，終端点を原点とする木構造の枝が時々刻々生成される．基本形複合アルゴリズムを用いると，各区間の状態遷移関数の逆写像の生成順序は，終端点から順次生成された経路に対応するコスト関数値の昇べき順序に正確に一致するため，この適用によって，終端点からの最小コスト関数の等コスト面（超平面）が時々刻々生成される．結果として，表形式の最適制御のマップが生成される．制御器のこのような計算過程に対しては，逐次近似の概念は使用されていない．

以下では，1 入力-1 出力系（Single-Input-Single-Output system，通常 SISO 系と表記）への適用例を示す．

4.1 最小時間制御系の設計

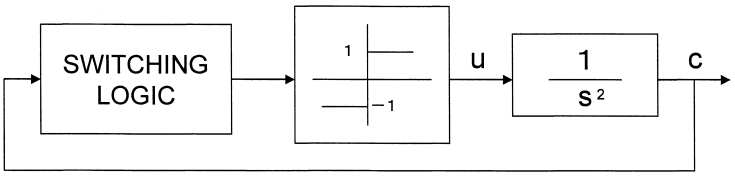

図 3：最小時間制御系

時間不変システムのクローズドフォーム解を得るために，図 3 に示されているような，制御入力に制約をもつ $1/s^2$ プラントを最短時間で原点に移動させる問題 (Larson, 1968) を考える．この問題は，

$$\text{Minimize} \quad J = \int_{t_0}^{t_f} dt = t_f - t_0 \tag{26}$$

$$\text{Subject to} \quad \dot{x}_1 = x_2(t) \tag{27}$$

$$\dot{x}_2 = u(t) \tag{28}$$

$$|u(t)| \leq 1. \tag{29}$$

と記述することができる．ここで，t_f は $x_1(t_f) = x_2(t_f) = 0$ によって決定される．目的は，この問題に対するクローズドフォーム解を基本形複合アルゴリズムを適用することによって求めることである．コンピュータ実行を行うため，問題の定義域をブロックサイズ $\Delta x_1 = \Delta x_2 = 0.025$ で量子化し，当該問題に対する最小原理による解の事前情報を用い，許容制御を $U = \{-1, +1\}$ とした．計算領域は $-2 \leq x_1 \leq 2$ および $-2 \leq x_2 \leq 2$ とし，また，$t \in [0, 4]$ とした．計算は原点 $(x_1, x_2) = (0, 0)$ から逆写像を作り出すように，ステップサイズ $\Delta t = 0.025$ の刻み幅で正確に基本形複合アルゴリズムをバックワードに実行させ，原点からの最小コスト関数の等コスト面を時々刻々生成させた．

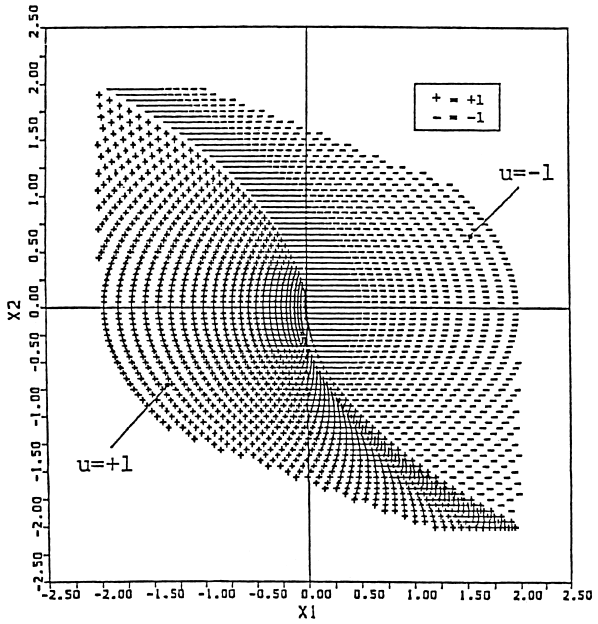

図４：最小時間制御のスイッチング線

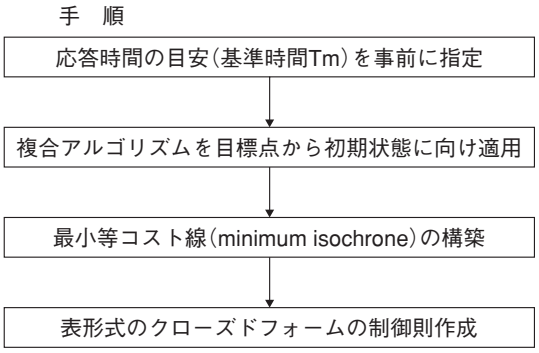

図５：非線形制御系の設計手順

この実行で得られた最適制御のマップを図４に示す．この図は，基本形複合アルゴリズムを適用することによって，最適制御集合 $U \in \{-1, +1\}$ を－１と＋１に区分するよく知られた滑らかなスイッチング曲線をコスト関数の煩わしい内挿計算を必要としないで得ることができることを示している．したがって，得られた最適制御は正確に実軌道上の制御である．計算時間は約４秒（100MIPS コンピュータに換算）であった．基本形複合アルゴリズムで計算した最小コストの値は厳密解と極めてよく一致した．定義域の全空間における平均誤差は約 0.12%であり，従来の動的計画法の精度に比べ本法は極めて高精度な最適制御と最小コストを計算できる

ことを示している．この計算で得られた最適制御則が最適制御器として使われることになる．非線形制御系の設計手順を図５に示す．

4.2 飽和非線形要素を含む制御系の設計

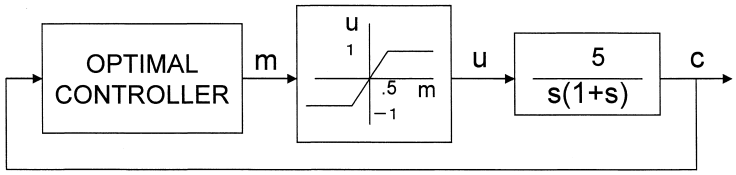

図６：飽和要素を持つ非線形制御系

図６に示されているような，プラントの一部に飽和非線形要素を含む次数２の非線形制御系を考える．このようなモデルは，フェーズが２の A-C サーボモータ等に見られる．目的は，与えられた評価関数と運動方程式の下で非線形制御系を設計することである．この問題は，

$$\text{Minimize} \quad J = \int_{t_0}^{t_f} (10x_1^2(t) + u^2(t))dt \tag{30}$$

$$\text{Subject to} \quad \dot{x}_1 = x_2(t) \tag{31}$$

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

$$x_1(0) = c_1 \tag{33}$$

$$x_2(0) = c_2 \tag{34}$$

$$|u(t)| \leq 1. \tag{35}$$

と記述することができる．

コンピュータ実行を行うため，問題の定義域をブロックサイズ $\Delta x_1 = \Delta x_2 = 0.025$ で量子化し，許容制御集合を $U = \{-1, 0.5, 0, +0.5, +1\}$ とした．計算領域を $-2 \leq x_1 \leq 2$ および $-4 \leq x_2 \leq 4$ とした．この計算で得られた最適制御器を図７に示す．小容量のコンピュータ主メモリーを用いる場合を考慮して，最適制御則は１次元あたり 80 データの表に格納された．計算時間は約 10 秒（100MIPS コンピュータに換算）であった．図８に，本法で設計した最適制御器による過渡応答と記述関数法で設計した進み補償器 $G(s) = (1 + 0.5s)/(1 + 0.125s)$ による過渡応答を比較した結果を示す．図中，１が本法，２が記述関数法，３が補償を行わない場合の過渡応答である．基本形複合アルゴリズムに基づいて設計した最適制御器を用いた場合の時間応答が，記述関数法のそれよりもより速い好ましい応答であることがわかる．

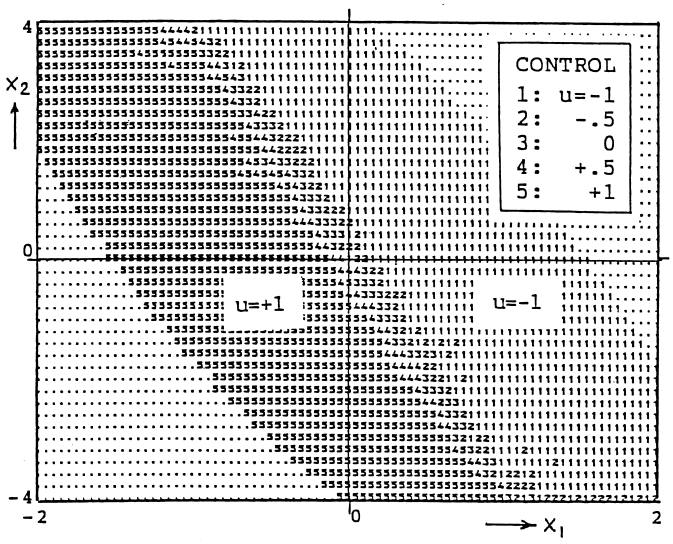

図７：飽和非線形要素を含む系の制御器

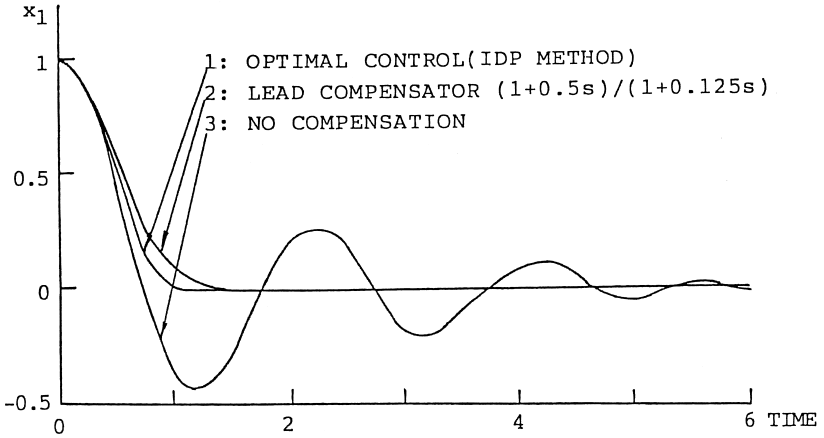

図８：種々の制御器による過渡応答の比較

多段レベル制御　過渡応答の原点付近のスイッチング境界での不正確な動作による振動を減少させるため，多段レベル制御が用いられた．これは，原点付近では制御入力の増幅度を低減させた低レベル制御のモードを用いることによって，制御入力のチャタリングレベルを減少させる．ここでは，原点付近で制御入力レベルを 1/10 に減少させた２段レベル制御を用いた．１段レベル制御モードは，原点付近での大きなエラーに対してアクティブであり，他方，２段レベル制御モードは，原点付近での小さなエラーに対してアクティブである．後者の制御器は，より低レベルの制御入力の下で，また，より細かく量子化された状態空間の下で，前述と同一の

最適制御問題に基本形複合アルゴリズムを適用して作成される．図 9 に，2 段レベル制御を用いない場合の最適制御のヒストリーを，また，図 10 に，低レベル制御モードの許容制御集合と状態空間をそれぞれ $|u(t)| \leq 0.1$ および $-0.2 \leq x_1 \leq 0.2$, $-0.2 \leq x_2 \leq 0.2$ とした場合の 2 レベル制御の最適制御のヒストリーを示す．これらの結果より，2 レベル制御モードでは，原点付近の制御入力の振動がかなり減少していることがわかる．

4.3 最小燃料消費制御系の設計

図 11 に示されたような 1 軸姿勢制御系を考える．ここで，制御トルク $L(t)$ は推力レベルの上限値をもつガスジェットの適当な配置によって供給されるものとする．$L(t)$ の飽和レベルを L_0 とし，制御入力 $u(t)$ を $u(t)=L(t)/L_0$ と定義する．このとき，姿勢誤差 x_1 とその微係数 x_2 の値を，上限と下限に制約のある制御入力を用いて最小燃料消費の評価指標の下で 0 に移行させる問題は，

$$\text{Minimize} \quad J = \int_0^T |u(t)|dt, \tag{36}$$

$$\text{Subject to} \quad \dot{x}_1 = x_2(t) \tag{37}$$

$$\dot{x}_2 = \frac{L_0(t)}{I}u(t) \tag{38}$$

$$x_1(0) = c_1 \tag{39}$$

$$x_2(0) = c_2 \tag{40}$$

$$|u(t)| \leq 1. \tag{41}$$

$$t \in [0, T] \tag{42}$$

と記述することができる．

ここで，I は対象物体の慣性モーメントであり，T は前もって固定されているか，または $0 \leq T \leq T_e$ のように固定時間 T_e によって制約されている．制約された応答時間 T に対して，その目安として基準時間 T_m, $0 \leq T_m \leq T_e$ を導入した．最初に，各基準時間 T_m に対する燃料最適制御則のマップが基本形複合アルゴリズムによって計算された．計算実行は逆写像のように原点から正確にバックワードに進められた．図 12 は $T_m=15$ に対して得られたマップである．図中，一部分に時間最適制御も見られる．もし必要なら時間最適制御のマップもまた基本形複合アルゴリズムによって計算することができ，2 つのマップは同じ原点で重ね合わせられる．この図で，基準時間 $T_m=15$ に対する典型的な燃料最適経路は，経路 $a-b-c-d$ である．その内，経路 $a-b$ は時間最適であり，経路 $b-c-0$ は真の燃料最適経路である．もし，各基準時間 T_mに対する制御則が瞬時に用いられたなら，固定時間 T に対する燃料最適制御則が得られる．

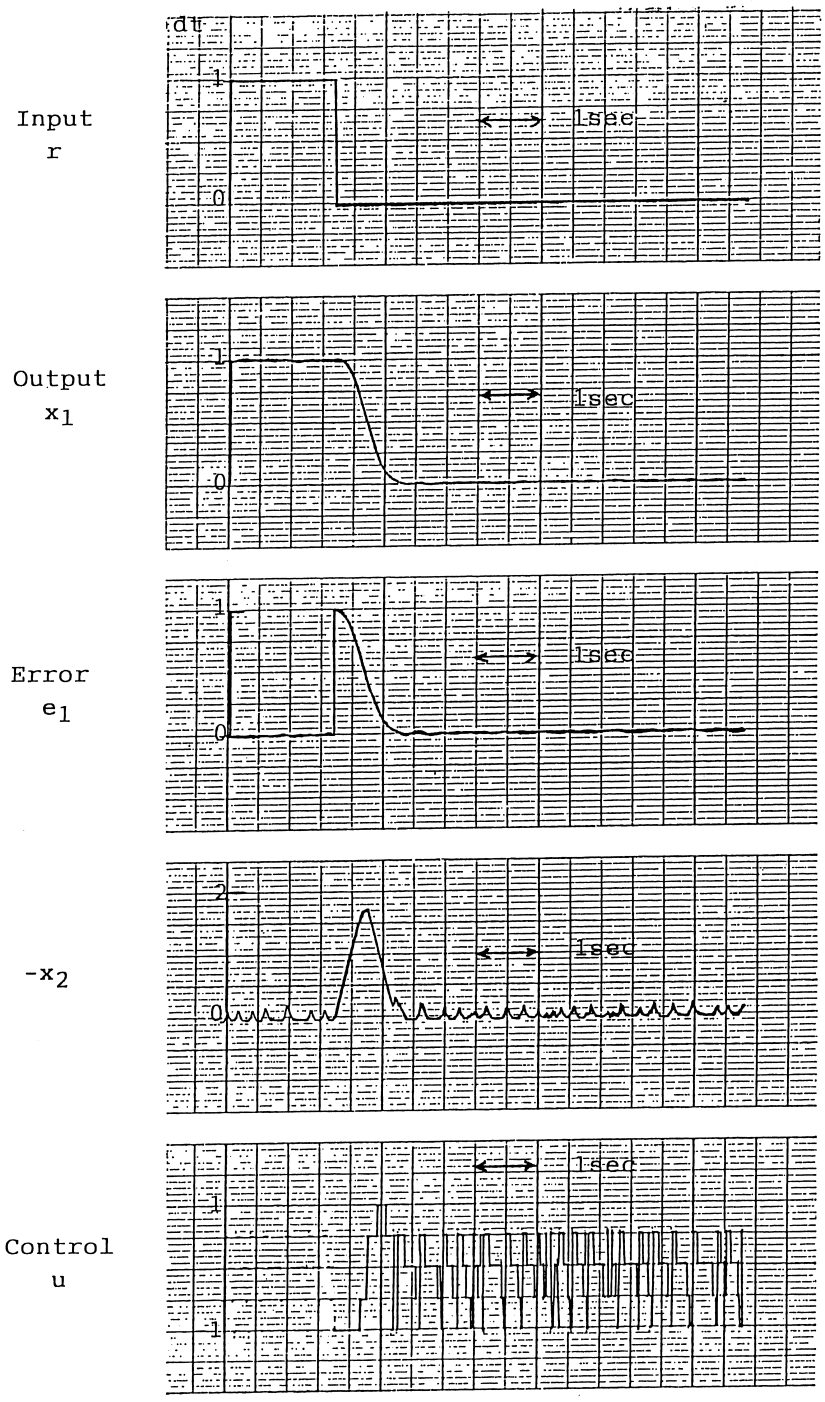

図９：２段レベル制御を行わない場合の応答

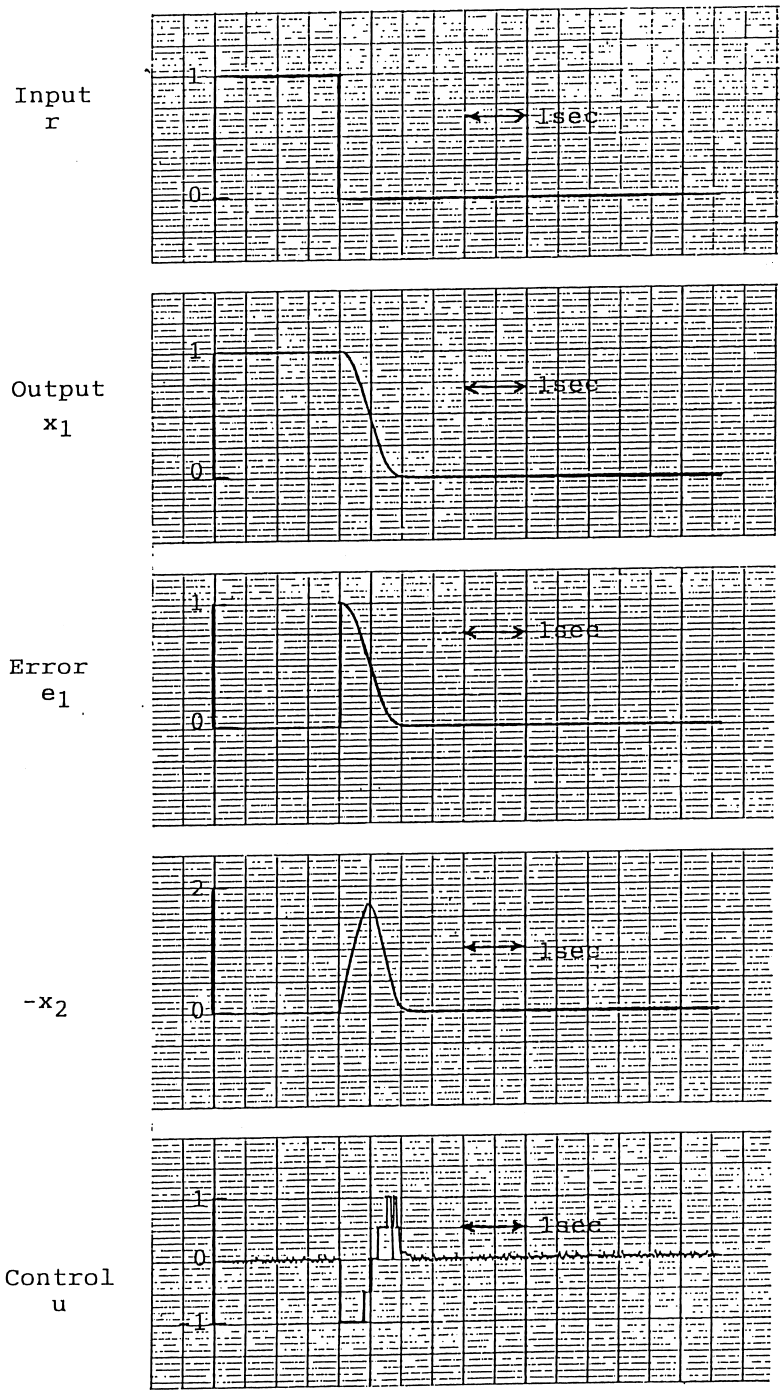

図10：2段レベル制御を行った場合の応答

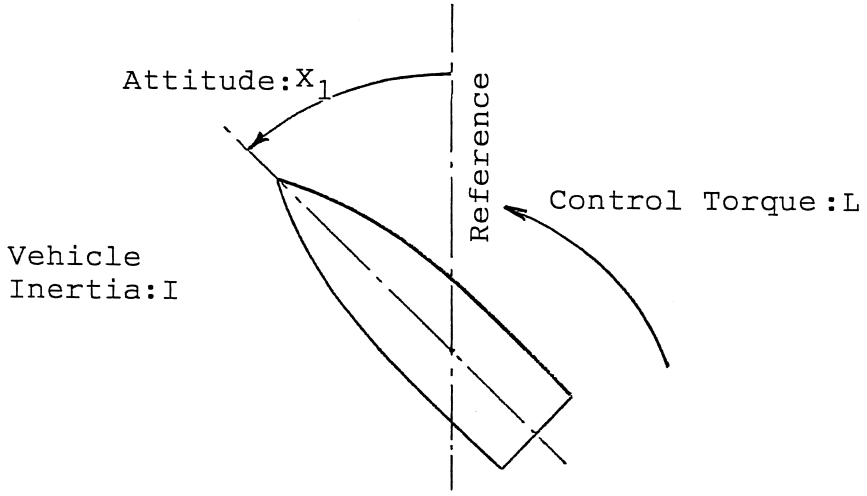

図 11：最小燃料消費系

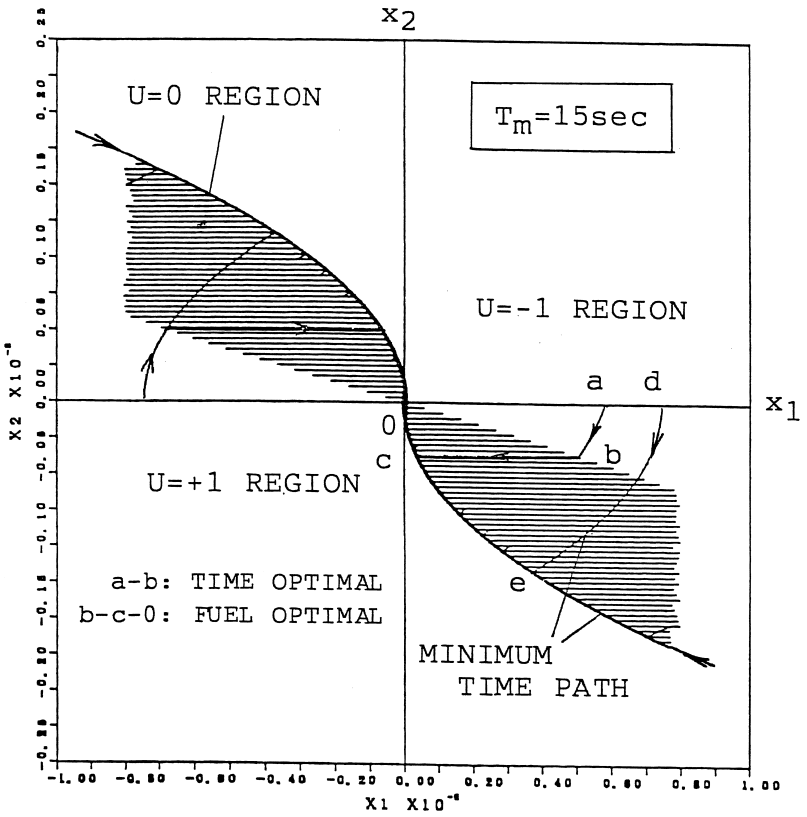

図 12：最小燃料消費系の制御器

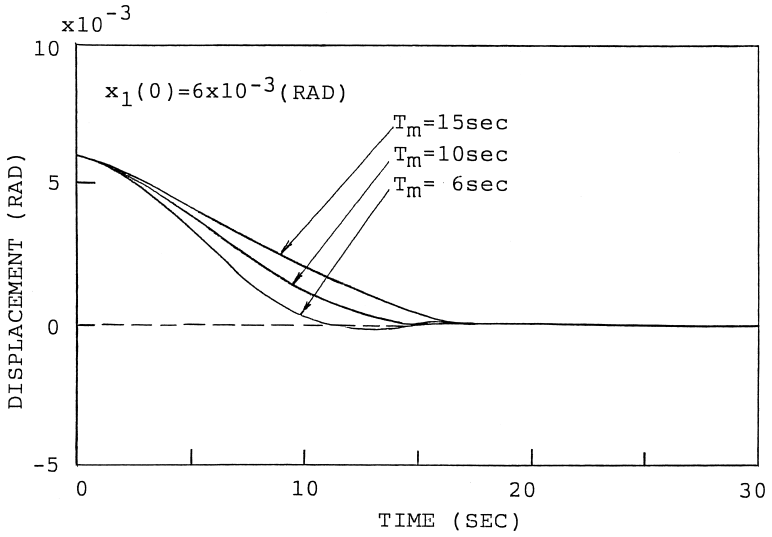

図 13：最小燃料消費系の過渡応答

図 13 は種々の T_m に対する過渡応答である.

5. MIMO 制御システム設計への応用

この章では，多入力- 多出力系（Multiple-Input-Multiple-Output system, 通常 MIMO と表記）の非線形制御システムの設計に対し,動的計画法複合アルゴリズムを適用した場合について議論する.

5.1 宇宙機の最小時間速度制御系の設計

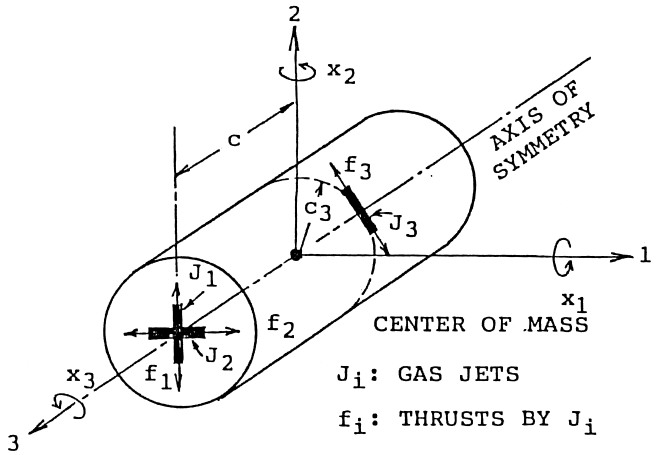

図 14：軸対象の宇宙機

図 14 に示すような，第３軸に関し軸対称の回転している宇宙機の速度を制御する問題（Athans and Falb, 1966）を考える．目的は，x_3 の回転速度を一定値 c_3 に保持しながら角速度 x_1, と x_2 を最小時間で 0 に減少させる推力 $f_1(t)$ と $f_2(t)$ を決定することである．

この問題は，次のような問題，

$$\text{Minimize} \quad J = \int dt \tag{43}$$

$$\text{Subject to} \quad \dot{x}_1(t) = \omega x_2(t) + k u_1(t) \tag{44}$$

$$\dot{x}_2(t) = -\omega x_1(t) + k U_2(t) \tag{45}$$

$$x_1(0) = c_1 \tag{46}$$

$$x_2(0) = c_2 \tag{47}$$

$$|u_1(t)| \leq 1 \tag{48}$$

$$|u_2(t)| \leq 1. \tag{49}$$

と等価である．ここで，w=1, k=1である．このとき問題は，どのような初期状態 $x_1(0)$ = $c_1, x_2(0)$ = c_2 をも原点 $(0, 0)$ に持っていく制御入力 $u_1(t)$ と $u_2(t)$ を見つけることである．

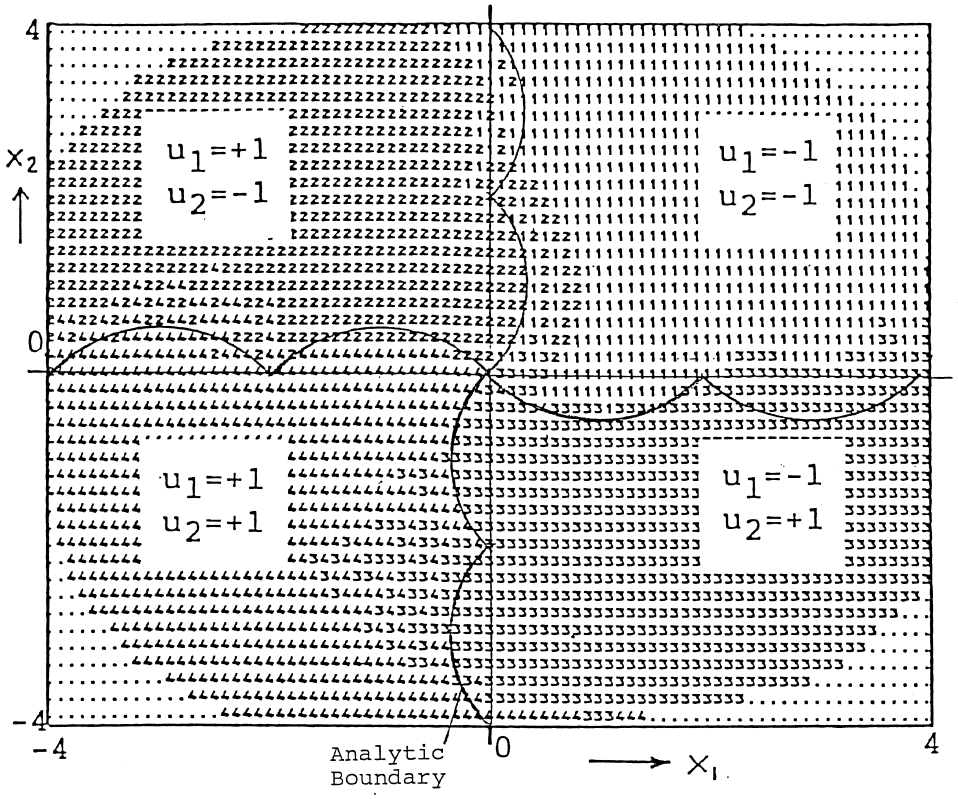

図 15：MIMO 宇宙機の制御器

この問題に対して，まずこのシステムのクローズドフォーム解を計算するため基本形複合アルゴリズムを適用した．状態空間は各状態変数をそれぞれ 320 レベルに量子化した．量子化された各ブロックには代表点が最大 1 つ格納される．コンピュータ実行を行うため，問題の定義域をブロックサイズ $\Delta x_1 = \Delta x_2 = 0.025$ で量子化し，当該問題に対する最小原理解の事前情報を用い許容制御を $U = \{-1, +1\}$ とした．計算領域は $-4 \leq x_1 \leq 4$ および $-4 \leq x_2 \leq 4$ の定義域とし，また，$t \in [0, 3.6]$ とした．計算は原点 $(x_1, x_2) = (0, 0)$ から逆写像を作り出すように，ステップサイズ $\Delta t = 0.03$ の刻み幅で正確にバックワードに実行させた．得られた最適制御則を図 15 に示す．図中，1, 2, 3, 4 の数字は，それぞれ $1 \equiv (u_1 = -1, u_2 = -1)$, $2 \equiv (u_1 = +1, u_2 = -1)$, $3 \equiv (u_1 = -1, u_2 = +1)$, $4 \equiv (u_1 = +1, u_2 = +1)$ の制御則を表す．制御のスイッチング線は，ほぼ厳密解に一致している．計算時間は約 26 秒（100MIPS コンピュータに換算）であった．次に，得られた制御器を用いて，この MIMO 系の過渡応答が調べられた．小さなメモリーサイズの制御器の使用を考慮して，制御器に格納したデータ数は 1 次元当たり 80 個に低減させた．図 16 と図 17 は，それぞれ当該制御器を用いたときの，初期条件 $(x_1(0), x_2(0)) = (3, 3)$ からの過渡応答における状態変数と制御変数の履歴であり，同一のステップサイズ $\Delta t = 0.025$ の刻み幅で最小原理を用いて計算した厳密解の制御則と比較してある．両者は極めてよく一致している．

6. まとめ

先に開発した動的計画法複合アルゴリズムを用いて非線形制御系を設計する新しい設計法を提案した．この方法は，種々の性能評価関数の最適制御方策を求める枠組みの下で，プラントのアクチュエータあるいはプラント自身，またはその双方に非線形要素を含む種々の非線形システムに対し，記述関数法における煩わしい試行錯誤の手続きを含まないで統一的に精度の高い制御器を設計することができる．また，本法は単入力- 単出力系だけでなく多入力- 多出力系への拡張が容易である．提案した方法は，クローズドフォーム制御器の現実の設計に組み入れることができる．このアプローチは，次数の高いシステムの制御器を得るには大きな計算数を必要とするが，現時点においては，4 次元システムに対しも適応できることを確認している．本研究の結果は，非線形制御系の設計に本法を適用すれば，従来の記述関数法の欠点を克服できることを示している．

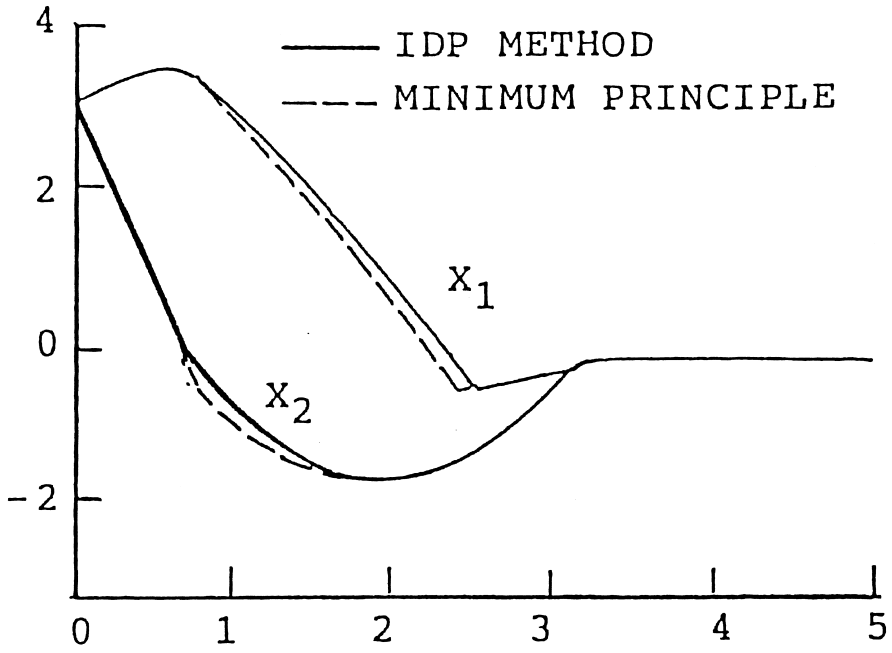

図 16：宇宙機の過渡応答における状態変数の履歴

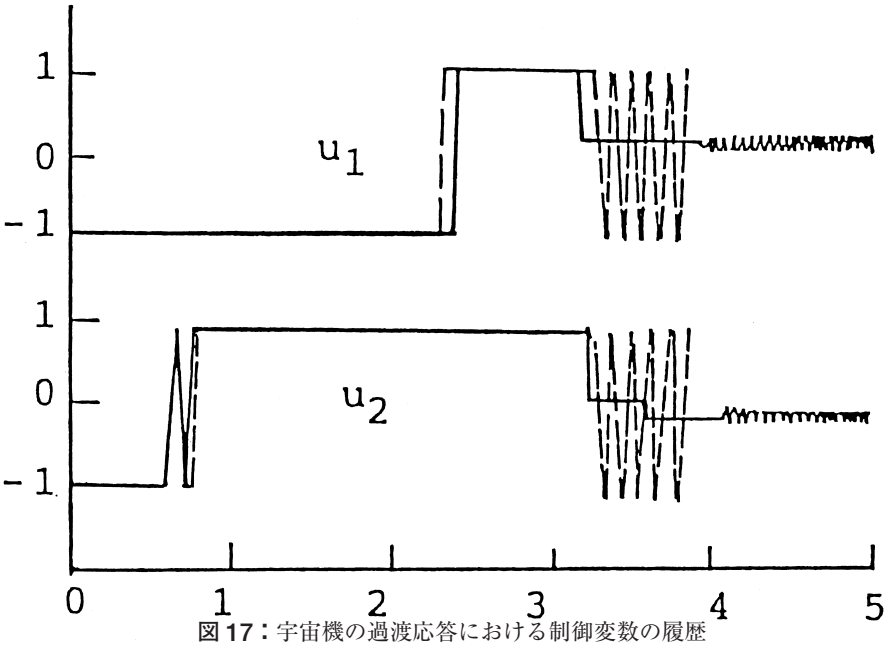

図 17：宇宙機の過渡応答における制御変数の履歴

参考文献

［1］Alekseev, O.G., and I.F. Volodos (1976), *Combined use of dynamic programming and branch-and-bound methods in discrete programming problem*, Automation and Remote Control, 37, 557-565.

［2］Athans, M., and P.L. Falb (1966), *Optimal Control*, Mcgraw-Hill Book Company, New York, Academic Press, New York.

［3］Bellman, R.E. (1957), *Dynamic Programming*, Princeton University Press, New Jersey.

［4］Hanaoka, T., and T. Tanabe (1982), *A new dynamic programming algorithm and its application to optimal reentry problem*, Proc. of 13th Int'l Symp. Space Tech. Sci., 1031-1036.

［5］Hanaoka, T. (1989), *A novel dynamic programming algorithm and its application to optimal low-thrust trajectory generation for space mission,* Preprint of 11th IFAC Symp. on Automatic Control in Space, Tukuba, 267-272.

［6］Jacobson, D.H., and D.Q. Mayne (1970), *Differential Dynamic Programming*, American Elsevier, New York.

［7］Larson, R.E. (1968), *State Increment Dynamic Programming*, American Elsevier, New York.

［8］Morin, T.L. and R.E. Marsten (1976), *Branch-and-bound strategies for dynamic programming*, Opns. Res., 24, 611-627.