跡見学園女子大学マネジメント学部紀要　第 19 号　（2015 年 1 月 25 日）

最適スイングバイ-低推力ミッション軌道生成への動的計画法複合アルゴリズムの適用

An Application to Optimal Swing-by Low Thrust Mission Trajectory Generation Using the Hybrid Dynamic Programming Algorithm

花　岡　照　明
Teruaki HANAOKA

要　旨

本論文では、太陽観測のために宇宙機を太陽に可能な限り接近させるための最適ミッション軌道を生成する手法が提案されている。もし必要なら、地球の周回軌道を出発した宇宙機は、途中、金星の重力場を利用してスイングバイを行い軌道変更を行うことによって近日点距離を最小化する場合を考える。今回用いた手法は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であり、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を最適スイングバイ-低推力ミッション軌道生成問題に適用して得られた最適解は典型的な推力計画 (latus rectum、circumferential、tangential) と比較されている。その結果、提案した複合アルゴリズムは太陽会合ミッションのような複雑な制約条件の下での大規模問題に対して極めて効果的であることが明らかになった。

キーワード：低推力、スイングバイ、最適軌道、動的計画法

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技巧を要せずに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、先に開発した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（以後、複合アルゴリズムと略記）の特性を明らかにし、その手法を太陽観測のために宇宙機を太陽に可能な限り接近させる最適ミッション軌道生成問題に適用した。もし必要なら、地球の周回軌道を出発した宇宙機は、途中、金星の重力場を利用してスイングバイを行い軌道変更を行うことによって近日点距離を最小化する場合も考慮する。従来、太陽会合ミッションのような複雑な制約条件の下での大規模問題に対して、システマティックに統一的に最適解を求めることのできる手法は見あたらなかった。また、解の精度が共役勾配法、最急降下法、あるいは微分法 (DDP) に匹敵する解を動的計画法を用いて求めることは極めて難しい問題であるとされてきた。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。

動的計画法の実行で必要となる計算数とメモリーサイズを低減させる研究として、様々な手法が提案されている[2)–7)]。Alekseev and Volodos[2] と Morin and Marsten[3] は、離散系の問題に対し、最適解の候補になり得ない状態を削除するために動的計画法と分枝限定法を併用した。一方、連続系の問題に対する主要な技法として、Larson[7] の state increment dynamic programming 法や Jacobson and Mayne の differential dynamic programming 法などが報告されている。前者の問題点は、真の解への収束を保証しないことと、状態遷移関数が、制御変数に関し可逆関数であることを要求することである。一方、後者では、初期制御方策の仮定を必要とし、解は局所的最適解（以下、局所解と表記）となるという問題があることや、微係数計算などの計算負荷の問題もあるとされている。Hanaoka and Tanabe[4] および Hanaoka[5] では、再突入飛行体の軌道最適化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（以下、基本形複合アルゴリズムと表記）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。

今回の複合アルゴリズムでは、この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と、局所解を避ける並列処理を組み入れた算法（以下、拡張複合アルゴリズムと表記）を提案する。それら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、局所解を回避することができる。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ている。一方、並列処理は、複数の最適経路候補を効率的に取り扱い、局所解へ陥ることを避けている。

拡張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、後述の「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。本論文で述べる拡張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), 7)]に置いている。

本稿では、以下で示す４つの概念（代表点、優先順位計算、限定操作、繰り返し推定）を併用することによって、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」を低減する方法について提案し、その特性を明らかにする。

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), 3), 4), 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。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が可能であり、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。

本論文では、拡張複合アルゴリズムの計算数、収束性、解の精度について調べている。また、実際問題への本法の適用性を調べるため、金星とのスイングバイを考慮した、低推力太陽探査軌道の生成問題が考察されている。

2　最適スイングバイ-低推力軌道生成への応用

2.1　問題の定義

太陽を観測するために宇宙機を太陽に可能な限り接近させるミッションを考える。宇宙機は、地球の周回軌道を出発し、一定レベルの低推力によって加速を受けながら、舵角 $\theta(t)$ によって制御されているものとする。ただし、途中、金星の重力場を利用した軌道変更（スイングバイ）を考慮に入れるものとし、このスイングバイ過程での制御変数を、近金点距離 λ (図 1)、および、スイングバイを行う際の惑星の側 s（太陽照射面／裏面）とする (図 2)。また、金星とのスイングバイ位置 p $(p = 1, 2, \ldots)$ を最適化するものとする (図 3)。ただし、p は、宇宙機が金星の公転軌道上で p 回目の交差点をもった時点でスイングバイを行うことを示す。このとき、問題は、あらかじめ指定した飛行時間 t_f 以内で、近日点距離 P_E を最小にする軌道を生成するように、最適

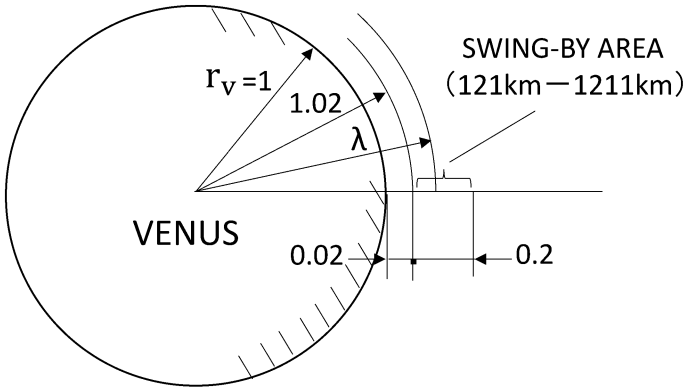

図 1: スイングバイを行う高度 (近金点距離 λ)

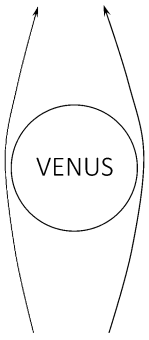

図 2: 金星重力場でスイングバイを行う側

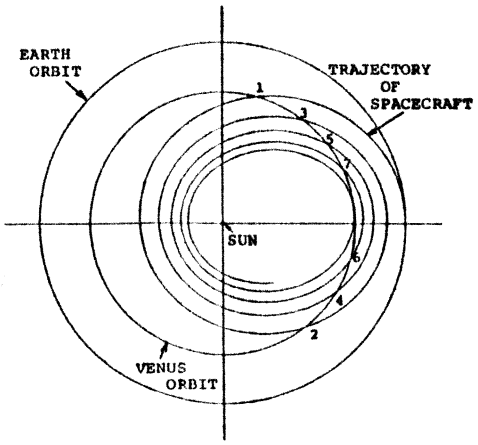

図 3: スイングバイを行う位置

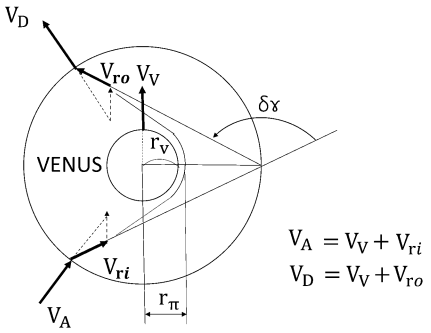

図 4: 金星スイングバイプロセス

制御列 $\{\theta(t), \lambda, s\}$ と最適スイングバイ位置 p を決定することである。金星スイングバイプロセスを図 4 に示す。

2.2 運動方程式と評価関数

この問題に対し、宇宙機の運動を平面に制約して考察する。質量 m の宇宙機の位置を極座標 (r, ϕ) で表わし、r を引力中心からの距離、ϕ を基準方向からの中心角、u を宇宙機の半径方向速度、また、v を円周方向速度とする。推力 T の方向を、この円周方向からの角度 θ（舵角）とする。これら状態量の定義を図 5 に示す。このとき、宇宙機の運動方程式を

$$\dot{r} = u \tag{1}$$

$$\dot{u} = \frac{v^2}{r} - \frac{\mu}{r^2} + \frac{T \sin\theta}{m_0 - |\dot{m}|(t - t_0)} \tag{2}$$

$$\dot{v} = -\frac{uv}{r} + \frac{T\cos\theta}{m_0 - |\dot{m}|(t - t_0)} \tag{3}$$

$$\dot{\phi} = \frac{v}{r} \tag{4}$$

と記述できる。ただし、μ を重力定数、m_0 と $-\dot{m}$ を、それぞれ、宇宙機の初期質量および推進剤の質量流量とする。また、t_0 と t_f $(t_f \leq t_{max})$ は所与であるとする。このとき、最小化すべき評価関数には太陽の近日点距離が取られ、

$$J = P_E(t_f) \tag{5}$$

となる。簡単化のため、スイングバイ回数を 1 回以下とし、スイングバイを行う際の近金点距離 λ の範囲を、$1.02 \leq \lambda \leq 1.22$ と仮定する。ただし、λ は金星半径で無次元化してある。ここでは、制約条件を、$0 \leq r \leq 1.5$ AU（天文単位）とした場合を考察する。飛行時間を 2 年とする。低推力のレベルを 2×10^{-5} gm/sec^2 (ケース A) および 1×10^{-5} gm/sec^2 (ケース B) の 2 ケースとする。また、推力方向角 θ の範囲を、$0 \leq \theta \leq 2\pi$ とし、初期速度増分と比推力を、それぞれ、5 km/sec と 5,000 sec とする。

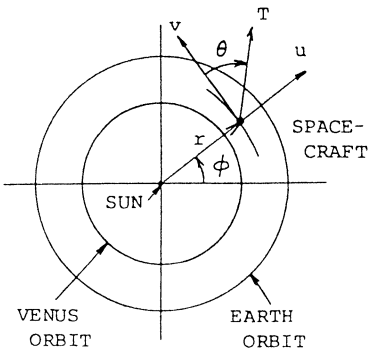

図 5: 状態量の定義

3　最適制御問題と動的計画法

いま、$\boldsymbol{x}_k$ 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 を段変数の添え字とする。ここで、

$$\boldsymbol{x}_k = \begin{bmatrix} r \\ u \\ v \\ \phi \end{bmatrix}, \quad \boldsymbol{u}_k = \begin{bmatrix} \theta \\ \lambda \\ s \end{bmatrix}$$

と定義し、$\boldsymbol{g}_k(\boldsymbol{x}_k, \boldsymbol{u}_k)$ を p 次元ベクトル値関数で表された状態遷移関数とすれば、最適スイングバイ-低推力軌道生成問題は以下のような最適制御問題として定式化できる。

$$\text{Minimize} \quad J = \sum_{k=0}^{N-1} L_k(\boldsymbol{x}_k, \boldsymbol{u}_k) + \Phi_N(\boldsymbol{x}_N) \tag{6}$$

$$\text{subject to} \quad \boldsymbol{x}_{k+1} = \boldsymbol{g}_k(\boldsymbol{x}_k, \boldsymbol{u}_k) \quad (k = 0, \ldots, N-1) \tag{7}$$

$$\boldsymbol{x}_0 = \boldsymbol{c}_0 \tag{8}$$

$$\boldsymbol{x}_N \in \Omega_F \tag{9}$$

$$\boldsymbol{x}_k \in X_k \quad (k = 1, \ldots, N-1) \tag{10}$$

$$\boldsymbol{u}_k \in U(\boldsymbol{x}_k) \quad (k = 0, \ldots, N-1) \tag{11}$$

ここで、$L_k(\boldsymbol{x}_k, \boldsymbol{u}_k)$ $(k = 0, 1, \ldots, N-1)$ 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。(8) と (9) 式は、それぞれ、初期条件と終端条件であり、Ω_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] の利点を保存しながら組み合わせ爆発による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を段に対して埋め込んでいる。しかしながら、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の一つ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が複雑なことや、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

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

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。

4　複合アルゴリズム

Bellman[1] の動的計画法は、状態の遷移方向、通常は時間の増加方向に関して、後向きに定式化していることは既に述べた。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

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 $\boldsymbol{x}_0$ からブロック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。

$$f_k(\boldsymbol{x}_{ki}) = \min_{\boldsymbol{x}_k}\{\tau(\boldsymbol{x}_k) \mid \boldsymbol{x}_k \in X_{ki}\}$$
$$(i = 1, \ldots, m_k,\ k = 0, \ldots, N) \tag{12}$$

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。

$$
\begin{aligned}
\tau(\boldsymbol{x}_0) &= 0 \\
\tau(\boldsymbol{x}_k) &= f_{k-1}(\boldsymbol{x}_{k-1i}) + L_{k-1}(\boldsymbol{x}_{k-1i}, \boldsymbol{u}_{k-1}) \\
\boldsymbol{x}_k &= \boldsymbol{g}_{k-1}(\boldsymbol{x}_{k-1i}, \boldsymbol{u}_{k-1}) \\
&\qquad (i = 1, \ldots, m_k,\ k = 1, \ldots, N)
\end{aligned} \tag{13}
$$

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、

$$
\begin{aligned}
f_0(\boldsymbol{x}_{01}) &= 0 \\
f_{k+1}(\boldsymbol{x}_{k+1i}) &= \min_{\boldsymbol{x}_{ki}, \boldsymbol{u}_k} \{f_k(\boldsymbol{x}_{ki}) + L_k(\boldsymbol{x}_{ki}, \boldsymbol{u}_k) \mid \boldsymbol{x}_{ki} \in X_k^\circ,\ \boldsymbol{u}_k \in U(\boldsymbol{x}_{ki})\} \\
&\qquad (i = 1, \ldots, m_k) \\
\boldsymbol{x}_{k+1i} &= \boldsymbol{g}_k(\boldsymbol{x}_{ki}, \boldsymbol{u}_k) \quad (k = 0, \ldots, N-1)
\end{aligned} \tag{14}
$$

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、

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

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、(13) 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

$$f^*_{0,N} = \min_{\boldsymbol{x}_N}\{f_N(\boldsymbol{x}_N) \mid \boldsymbol{x}_N \in \Omega_F\} \tag{15}$$

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 (6)〜(11) 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

$$\begin{aligned} M_k(\boldsymbol{x}_k) &\le J_k(\boldsymbol{x}_k), \quad \boldsymbol{x}_k \in X_k \\ &= \min_{\boldsymbol{u}_k \in U(\boldsymbol{x}_k),\, \boldsymbol{u}_{k+1} \in U(\boldsymbol{x}_{k+1}),\, \ldots,\, \boldsymbol{u}_{N-1} \in U(\boldsymbol{x}_{N-1})} \left\{ \sum_{i=k}^{N-1} L_i(\boldsymbol{x}_i, \boldsymbol{u}_i) + \Phi_N(\boldsymbol{x}_N) \right\} \\ &\quad (k = 0, \ldots, N-1) \end{aligned} \tag{16}$$

$$I \ge f^*_{0,N} \tag{17}$$

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

$$f_k(\boldsymbol{x}_k) + M_k(\boldsymbol{x}_k) > I \quad (k = 0, \ldots, N) \tag{18}$$

を満たすならば、

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

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[4)]。ただし、$f_0(\boldsymbol{x}_0)$ は、(13) 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、(18) 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、

$$\Delta a = I - f^*_{0,N}, \quad \Delta b_k = J_k(\boldsymbol{x}_k) - M_k(\boldsymbol{x}_k) \tag{19}$$

とおく。(19) 式の I と $M_k(\boldsymbol{x}_k)$ を (18) 式に代入すると

$$f_k(\boldsymbol{x}_k) + J_k(\boldsymbol{x}_k) > f^*_{0,N} + \Delta a + \Delta b_k \tag{20}$$

となる。(20) 式より、基本形複合アルゴリズムの計算量は、Δa や Δb_k の和 $\Delta\epsilon_k$ ($\Delta\epsilon_k = \Delta a + \Delta b_k$) に依存する。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、

$$\Delta\epsilon_k = \Delta a + \Delta b_k \geq 0 \tag{21}$$

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数が最小となるのは、$\Delta\epsilon_k = \Delta a + \Delta b_k = 0$ 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、$\Omega \leftarrow \Omega \setminus \{\boldsymbol{x}^*\}$ とせよ。ただし $\setminus$ 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。

(18) 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

$$u_{kmin}^{l,i} \leq u_k^{l,i} \leq u_{kmax}^{l,i} \tag{22}$$

ここで、

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

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

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、

$$I^i = f_{0,N}^{*i-1} \quad (i = 2, 3, \ldots) \tag{23}$$

$$\begin{aligned} M_k^i(\boldsymbol{x}_k) &= J_k^{i-1}(\boldsymbol{x}_k^{*i-1}) \\ &= \sum_{l=k}^{N-1} L_l(\boldsymbol{x}_l^{*i-1}, \boldsymbol{u}_l^{*i-1}) + \Phi_N(\boldsymbol{x}_N^{*i-1}) \\ &\quad (k = 0, 1, \ldots, N-1,\ i = 2, 3, \ldots) \end{aligned} \tag{24}$$

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。(23) 式による I^i は良好な上界値を与える。また、(24) 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

$$|f_{0,N}^{*i} - f_{0,N}^{*i-1}| \ll \epsilon_1 \tag{25}$$

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。

4.6 並列複合アルゴリズム

大域解を得るためには、状態変数の定義域全体に対して、$\Delta\epsilon \geq 0$ (クリアランスの条件 (21) 式) となる必要がある。そのために、もし、$\Delta\epsilon < 0$ の領域を含む場合は、(24) 式の下界値を下方修正し、

$$M_k^i(\boldsymbol{x}_k) = J_k^{i-1}(\boldsymbol{x}_k^{*i-1}) - |\Delta\tilde{\epsilon}_{min}|$$
$$(k = 0, 1, \ldots, N-1,\ i = 2, 3, \ldots) \tag{26}$$

と置く。ただし、$\Delta\epsilon$ の値が明示的でないため、$\Delta\tilde{\epsilon}_{min}$ を $\Delta\epsilon$ (< 0) の最小値の推定値とする。もし、$\Delta\epsilon \geq 0$ ならば、$\Delta\tilde{\epsilon}_{min} = 0$ と置く。しかし、コスト関数が多峰性である場合には、(26) 式のように、一つのパラメータ $\Delta\tilde{\epsilon}_{min}$ によって、全定義域に渡る下界値を設定することは、クリアランスの増加、したがって、計算数とメモリーサイズの増加を招く。

並列複合アルゴリズムは、このような場合に、大域解を求める実際的な算法である。最初に、並列処理のために、粗い量子化の下で経路候補群を生成する。これらの経路での値を、並列計算の初期値とする。これは原問題のコスト構造を調べることに相当する。すなわち、並列複合アルゴリズムでは、繰り返し複合アルゴリズムの初回 $(i = 1)$ の計算で得た、1 番目から j_{max} 番目までのコスト値をもつ最適経路候補群に対し、次回以降 $(i \geq 2)$、それぞれ個別に、繰り返し複合アルゴリズムを適用し、各々の逐次近似解を求める。したがって、並列複合アルゴリズムの i 回目 $(i \geq 2)$ の繰り返し計算が終了したとき、新たな最適経路候補群が生成される。並列複合アルゴリズムに対しても、i 回目のイテレーションでの状態集合と許容制御の量子化のしかた、および、第 j 番目の経路に関する上下界値の更新のしかたは、繰り返し複合アルゴリズムと同様である。すなわち、

$$I^{j,i} = f_{0,N}^{*j,i-1} \quad (j = 1, 2, \ldots, j_{max},\ i = 2, 3, \ldots) \tag{27}$$

$$\begin{aligned} M_k^{j,i}(\boldsymbol{x}_k) &= J_k^{j,i-1}(\boldsymbol{x}_k^{*j,i-1}) - |\Delta\tilde{\epsilon}_{min}^j| \\ &= \sum_{l=k}^{N-1} L_l(\boldsymbol{x}_l^{*j,i-1}, \boldsymbol{u}_l^{*j,i-1}) + \Phi_N(\boldsymbol{x}_N^{*j,i-1}) - |\Delta\tilde{\epsilon}_{min}^j| \\ &\quad (k = 0, 1, \ldots, N-1,\ j = 1, 2, \ldots, j_{max},\ i = 2, 3, \ldots) \end{aligned} \tag{28}$$

によって強化する。ここで、$\Delta\tilde{\epsilon}_{min}^j$ は、クリアランスの条件 $\Delta\epsilon \geq 0$ (21) 式を満たす範囲を、全定義域から第 j 番目の最適経路周りの回廊と呼ばれる領域 (図 6 参照) に縮小したときの推定値とする。

さらに、この上界値 $I^{j,i}$ は、並列処理によってすでに計算された、より小さな上界値、すなわち、

$$\tilde{I}^{j,i} = \min_{j_1 \in \{j, j+1, \ldots, j_{max}\},\, j_2 \in \{1, 2, \ldots, j-1\}} \{f_{0,N}^{*j_1,i-1}, f_{0,N}^{*j_2,i}\}$$
$$(j = 1, \ldots, j_{max},\ i = 2, 3, \ldots) \tag{29}$$

によって強化することが可能である。このとき、第 j 経路の繰り返し計算で、全ての経路が限定操作により削除され、したがって、第 j 局所解が消去される可能性を期待できる。また、i 回目の繰り返し計算では、より小さな上界値 $I^{j,i}$ をもつ経路を先に処理すれば、早い時点で、より強力な上界値を得る可能性が高くなる。

前述の並列計算では、固定した各 i $(i = 2, 3, \ldots)$ に対して、j を変化させた。一方、固定した各 j $(j = 1, \ldots, j_{max})$ に対して、i を変化させれば、もう 1 つの並列計算を構成できる。

並列複合アルゴリズムの実行は、前回 $(i = 1, 2, \ldots)$ の量子化単位下で計算した大域解と代表的な局所解の周りに、最適経路候補群からなる計算領域の回廊を逐次生成することに対応している。並列複合アルゴリズム (parallel hybrid dynamic programming algorithm、parallel HDP と略記) の計算領域の回廊を図 6 に示す。この回廊幅のサイズは、クリアランスのサイズ $\Delta\epsilon$ によって制御され、イテレーション回数 i の増加とともに、上下界値の逐次改良によって、徐々に絞られる。

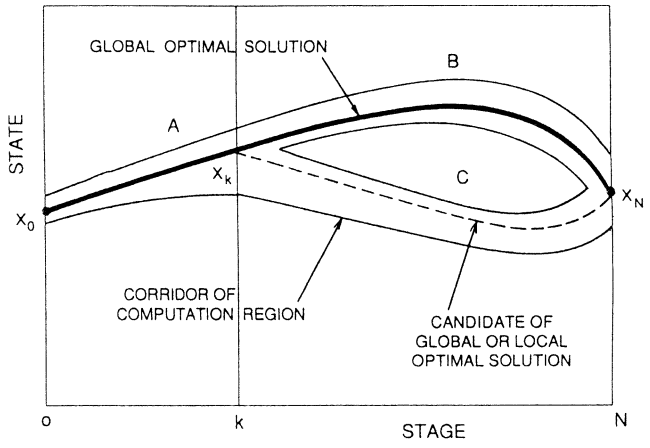

図 6: 並列複合アルゴリズムの計算領域の回廊

5 計算結果

最適スイングバイ-低推力軌道生成問題に複合アルゴリズムを適用して得られた結果を示す。この数値例では、暦による宇宙機の具体的な出発日時を含めた計算を行っていない。また、コースティングがない場合を考察する。この問題に対し複合アルゴリズムを適用するため、許容状態集合をブロックに分割する。初回の計算 $(i = 1)$ では、状態変数 r、u、v の定義域を各 16 分割した。一方、制御変数 θ、λ、s の定義域を、それぞれ、8、5 および 2 レベルに分割した。すべてのイテレーションを通じ、独立変数 t を 40 分割し、段 k と値 t_k を割り付ける。初回の計算での上

界値 I^1 は、粗い量子化レベル下での数回の試行計算によって推定した。一方、下界値 M_k^1 を零と置いた。以後のイテレーションでは、状態変数の量子化レベルを各120分割 $(i=8)$ まで増加させ、また、制御変数の可変域を徐々に減少させた。スイングバイ過程以外の状態の遷移およびコスト関数の計算には、4次-ルンゲクッタ法を単精度で用い、ステップサイズを $\pi/50$ とした。また、各区間 $t_k \le t \le t_{k+1}$ での推力方向角を、$\theta_k(t) = \theta(\theta_k^*, \theta_{k+1}, t)$ $(\theta_{k+1} \in U、t_k \le t \le t_{k+1})$ とし、この区間を直線近似した。ただし、θ_k^* を、代表点での最適制御とする。

まず、ケースAの場合を考える。並列複合アルゴリズム (parallel HDP) で計算した最適解の最小近日点距離 P_E は0.1115 AU（天文単位）であり、スイングバイの位置は1であった (図7)。この図より近日点距離はスイングバイによって瞬時に減少していることがわかる。図8に本ケースの最適制御と状態変数の変化の時間変化を示す。このときのコントロールの作り方をFREE (CONTROL I) と表記することにする。図9にスイングバイ位置を1に固定した下での典型的な推力計画であるlatus rectumとcircumferentialの制御レベルの時間変化をFREEと共に示す。この図より、latus rectumとFREEは近日点付近で推力レベルを大きく変化させており、この部分での推力方向角の量子化による影響を極力避けることが望ましい。そのために (latus rectum + FREE) を新たな推力レベルの自由パラメータとしてコントロールを作成した。このコントロールの作り方をlatus rectum + FREE (CONTROL II) と表記する。CONTROL IIを

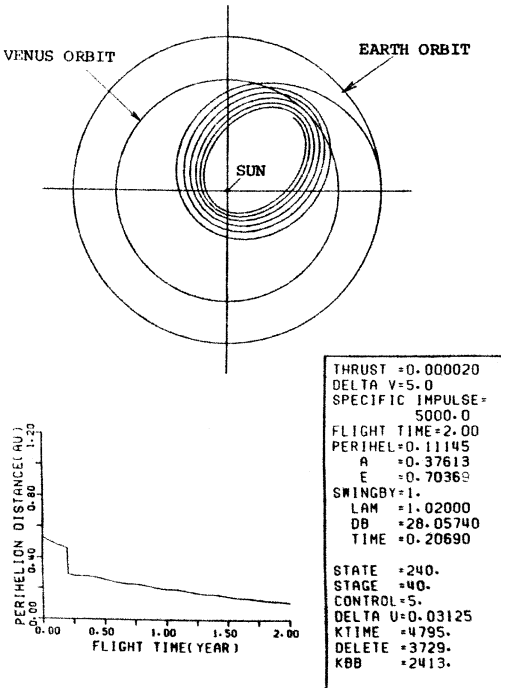

図7: 最適軌道と近日点距離 (ケースA、$p=1$、CONTROL I)

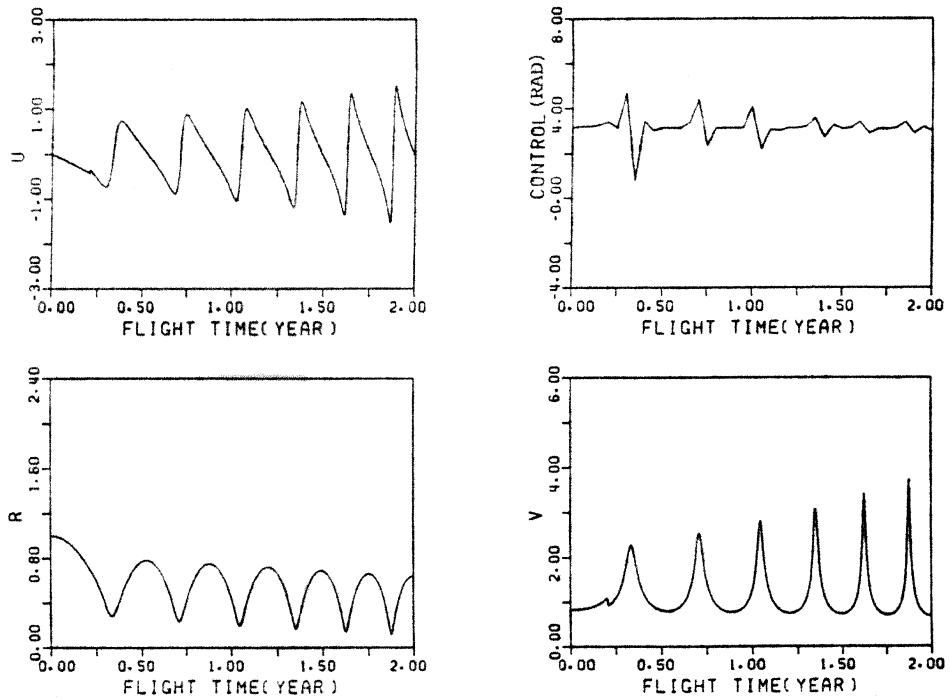

図 8: 状態変数 r、u、v と最適制御 I の変化

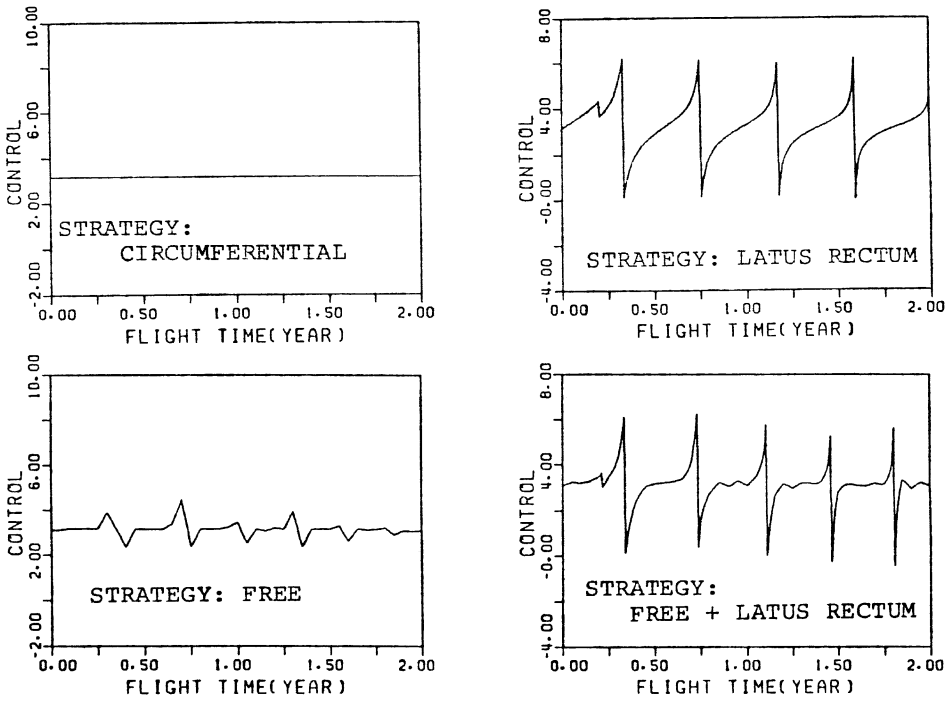

図 9: 典型的な 2 つの推力計画と CONTROL I および CONTROL II

用いた場合の最小近日点距離はスイングバイ位置 1 で起こり、0.0974 AU に改善された (図 10)。この場合の最適制御と状態変数の時間変化を図 11 に示す。比較のため、表 2 に、ケース A とケース B に対し、latus lectum（楕円の通径方向、したがって、短径方向)、tangential（宇宙機の速度方向）および、circumferential（円周方向）を推力方向とした固定推進戦略を用いた場合の近日点距離の数値解を、CONTROL II を用いた場合 (parallel HDP) と併記して示した。この表より、繰り返し複合アルゴリズムでの解は、ほかのいずれの固定推進戦略よりも、より小さな近日点距離を得ている。また、スイングバイを行わない場合 (NOTHING) と比べ、スイングバイの効果がかなり大きいことがわかる。上述のように、ケース A での最適解は、$p = 1$、$\lambda = 1.02$ の位置で、スイングバイを 1 回行った場合に得られ、最小近日点距離として、0.0974 AU を得た。この場合の最適軌道は、地球の公転軌道の内側に向かう。この様子を図 10 に示す。計算時間は、約 300 秒 (15〜20 MIPS 計算機) であった。従来型動的計画法に対する計算数の割合は、約 1/300 ($i = 1$) から 1/140,000 ($i = 8$) 程度であり、一方、計算領域は約 1/100 ($i = 1$) から 1/40,000 ($i = 8$) に減少した。

これらの結果より、並列複合アルゴリズムは、低推力推進系と惑星スイングバイを併用したような、実際的な複雑な軌道最適化に対して、十分、適用可能であると思われる。特に、低推力推進系を用いる場合には、低加速度ゆえに、解析にインパルス近似を用いることができないから、本法は、有効な手段を提供すると考える。

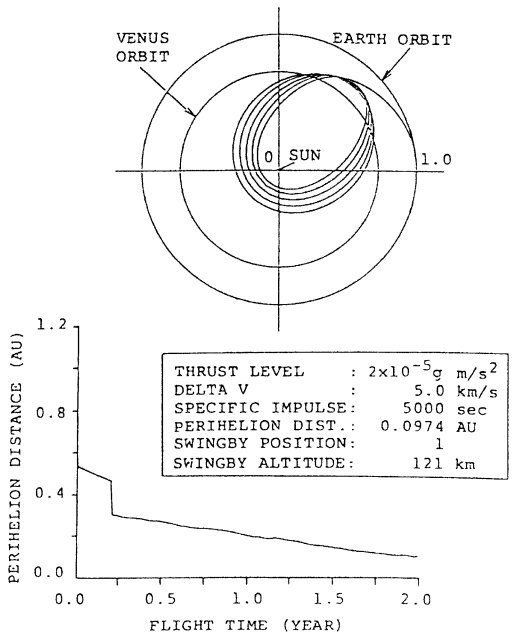

図 10: 最適軌道と近日点距離 (ケース A、$p = 1$、CONTROL II)

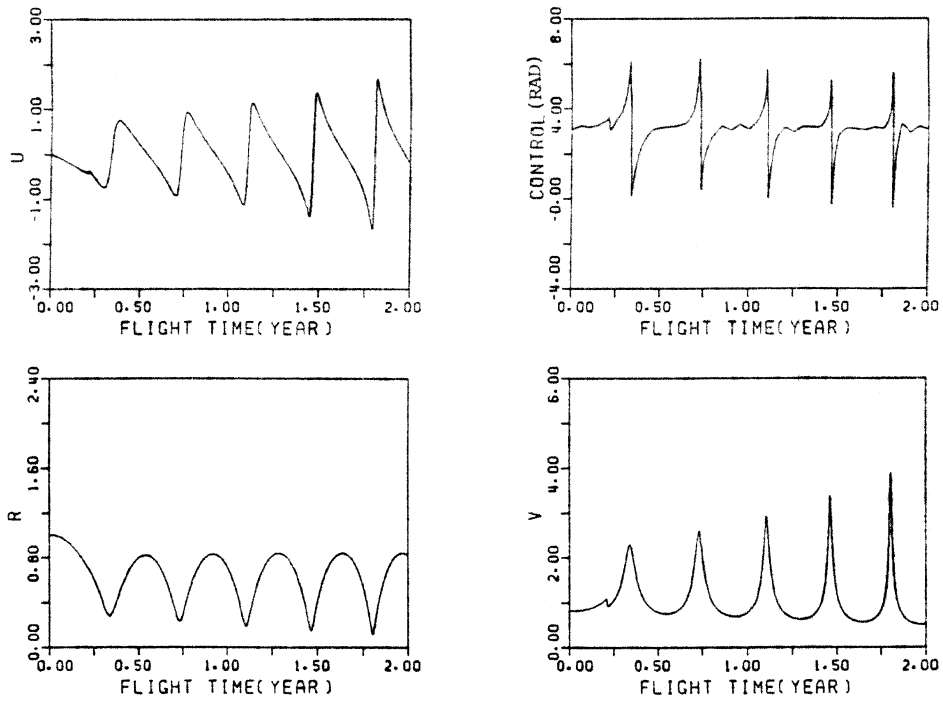

図 11: 状態変数 r、u、v と最適制御 II の変化

表 2: 種々の手法によって得られた近日点距離

UNIT : (AU)

CASE	SWINGBY POSITION	PARALLEL HDP	LATUS RECTUM	CIRCUM-FERENTIAL	TANGEN-TIAL
A	1	0.0974	0.100	0.123	0.140
	2	0.106	0.107	0.125	0.143
	3	0.0988	0.113	0.119	0.137
	4	0.103	0.118	0.152	0.212
	NOTHING	0.139	0.148	0.197	0.219
B	1	0.203	0.211	0.207	0.219
	2	0.204	0.220	0.208	0.218
	3	0.204	0.226	0.206	0.215
	4	0.207	0.234	0.210	0.221
	NOTHING	0.307	0.331	0.332	0.347

THRUST LEVEL A : 2×10^{-5}gm/s^2, B : 1×10^{-5}gm/s^2

6 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、代表的な状態制約最適制御問題に適用し、解の

収束性、解の精度、計算量（計算時間）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。

6.1　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

$$\text{Minimize} \quad J = \int_0^1 \left((x_1(t))^2 + (x_2(t))^2 + 0.005u(t)^2\right) dt \tag{30}$$

$$\text{subject to} \quad \dot{x}_1(t) = x_2(t), \quad x_1(0) = 0 \tag{31}$$

$$\dot{x}_2(t) = -x_2(t) + u(t), \quad x_2(0) = -1 \tag{32}$$

$$x_2(t) \leq 8(t - 0.5)^2 - 0.5 \tag{33}$$

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、基本形複合アルゴリズムによる最適コストの 1% とした。コストの収束の様子を図 12 に示す。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 13 と図 14 に示す。計算時間は 20 MIPS 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。

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 (33) 式

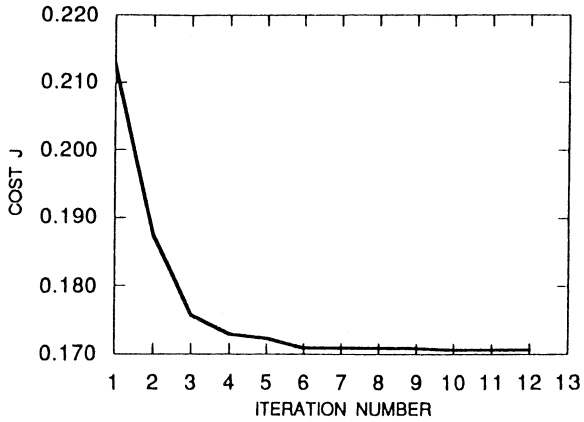

図 12: イテレーションに対するコストの収束

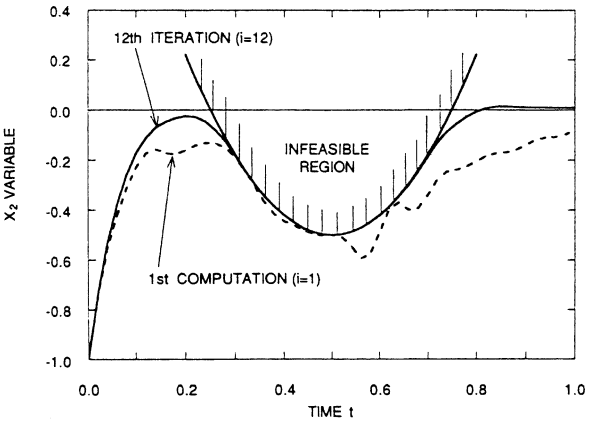

図 13: 経路 x_2 の収束

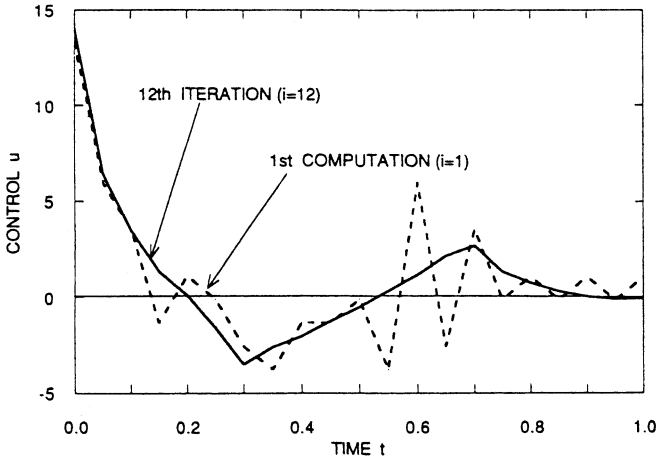

図 14: 制御入力の収束

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チューブの大きさ、すなわちクリアランス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、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、最適解の候補経路と成り得ることである。

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。

6.2　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 DDP 法などの他の計算法による初期解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、他の計算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。
- 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。

しかしながら、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適用
- 制約のある航空路の設計問題への応用
- 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3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、解の精度の観点においては最大原理から従来の計算法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15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

表 3: 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

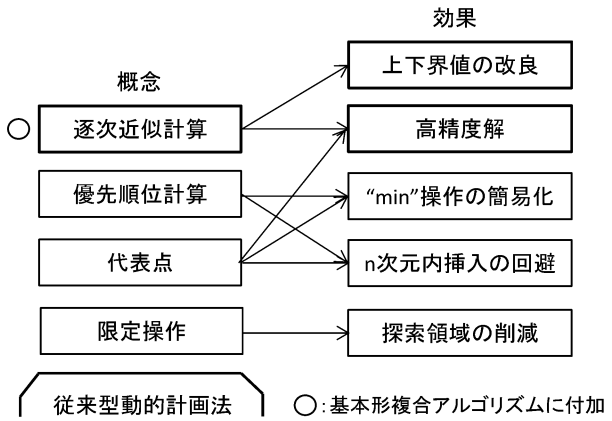

図 15: 繰り返し複合アルゴリズムの概念と効果の関係

関係である。これらの概念は従来の動的計画法に加えられた概念であり、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。

7　まとめ

本論文では金星スイングバイを考慮した低推力ミッション軌道最適化問題に対して並列繰り返し複合アルゴリズムを適用し最適解を求める手続きを提案した。種々の条件の下で得られた近日点距離は典型的な推力計画（固定方策）と比較されている。すべての場合において、複合アルゴリズムで得られた近日点距離は、固定方策よりもより小さな値を示している。また、従来型動的計画法との比較で、計算数を大幅 (約 $1/300$ $(i=1)$ から $1/140{,}000$ $(i=8)$) に低減させることができた。この結果は、低推力軌道最適化を含む、複雑な最適制御問題に対し、並列動的計画法

複合アルゴリズムを適用できることを示している。

また、複合アルゴリズムは従来の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。数値解による検討結果では、この拡張した並列複合アルゴリズムの解の精度は 0.13% 程度であり、differential dynamic programming のそれに匹敵する性能を示した。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,” *Operations Research,* **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 Society 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,* **41**-2, 289–310 (1998).

7) R. E. Larson: *State Increment Dynamic Programming,* American Elsevier 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).