オペレータを作用させたモジュールによる
機械システムのコンカレント設計*吉村 允孝^{*1}, 堀江 聡^{*2}Concurrent Design of Mechanical Systems
Using Operator-Acting Modules

Masataka YOSHIMURA and Satoshi HORIE

Concurrent Engineering is a methodology allowing designers and manufacturing engineers to work in parallel, or concurrently, during various stages of product design and manufacturing. Given the variety of concurrent engineering design methods, this paper proposes a concurrent design method for mechanical systems using operator-acting modules. In this method, in order to simplify the system being considered, it is divided into modules. Then, the modules' specifications and the system are changed by using operators which act to expand the design space. By using modules to express the mechanical system, several stages of design and manufacturing variables can be treated within the modules concurrently, in parallel, so that high performance and high quality machine products can be realized in a short time. The proposed method is applied to the design of a material handling manipulator consisting of several links, and the validity of this design method is discussed.

Key Words: Design Engineering, Production System, Optimum Design, Systems Engineering, Concurrent Engineering, Concurrent Design, Module, Operator, Design Space, Manipulator

1. 緒 言

現在の機械製品は、多様なニーズに応えるべく、多機能化、高品質化、そして低価格化の傾向にあるため、従来の製造業の環境下では、その要求をすべて満たすことが困難になりつつある。そこで、設計・生産分野における環境を統合・直結化することで、複雑化した機械システムの設計を実現しようとする動きが注目されている。その例が、コンカレントエンジニアリングやコンカレント設計⁽¹⁾⁽²⁾であり、本研究では、この設計手法を具体化するため、モジュール⁽³⁾の概念を導入した設計法を提案する。

モジュールとは、対象を機能的に分割することによって得られる、単純な要素のことである。モジュールは、明確に規定された外部仕様を有し、その組合せで全体としての挙動を実現する。本論は、コンカレントエンジニアリングの導入によって、設計問題が複雑化することを軽減するために、システムのサブシステムであるモジュールを用いて、対象システムのモデル化

を行うものである。特に、モジュールのもつ設計変数の集約効果を十分に発揮するため、本研究の適用対象は、いくつかの類似した要素が組合せてできている複雑なシステムである。例えば、リンク機構のように、単一リンクの組合せからなるシステムは、適用可能対象である。

本方法論は、まず、設計問題に含まれる設計変数を集約することで、問題の単純化を行い、モジュールを定義する。さらに、設計仕様に拡張性をもたせるため、モジュールの外部仕様を変更するオペレータを作成する。この方法論により、いくつかのモジュールの組合せからなる多様なシステムを、フラットなレベルで評価することができる。また、本論の適用例として、搬送用マニピュレータの設計問題を取上げる。

2. モジュールによるシステムの表現方法

2・1 コンカレント設計の実現 対象とする問題が、幾つかのサブシステムに分けられ、それらが相互作用した状態で表現できたとする。そして、そのサブシステムをモデル化し、規格化する。このサブシステムをモジュールと呼び、システムはこのモジュールの集合体と考えられる。また、モジュールを用いた設計法として、トップダウン法とボトムアップ法が存在す

* 原稿受付 1998年3月18日。

*¹ 正員、京都大学大学院工学研究科 (〒606-8501 京都市左京区吉田本町)。*² 京都大学大学院。

E-mail: yoshimura@prec.kyoto-u.ac.jp

Fig. 1 The design method using modules

る⁽⁴⁾。図1は、二つの設計法の概念図である。トップダウン法とは、システムを分割することによって生成される、解析や設計の可能なモジュール1～ n に対して評価をし、その評価を総合することによってシステム全体の評価を得るものである。逆に、ボトムアップ法は、あらかじめ問題に対するモジュール1～ n を用意しておき、その組合せでシステムを生成して、評価するものである。この二つの方法の違いは、トップダウン法では、モジュールの種類と数がシステムを単純化した際に決定されるのに対し、ボトムアップ法では、システム構築可能なら、モジュールの種類と数を自由に決めることができることにある。ただし、この二つの設計方法を同時に用いることは、現在の機械製品の製造環境下では困難である。なぜなら、従来の設計法は、設計段階が一方方向に進行する逐次設計法が主流であり、また仮に、その状況下で双方向の設計を行うと、扱うべき設計変数の増加を招き、設計に膨大な時間を費やしてしまうからである。

そこで、本研究では、この双方向設計を実現するため、モジュールを用いた独自の設計法を提唱する。この設計法は、モジュール化されたシステムに対して、スムーズにボトムアップ法が適用できるように、モジュールの表現に工夫を与え、さらに、多様なシステムを一つの設計過程で扱うため、システムの変更をモジュールレベルで実行するオペレータを提案した。以下の節では、本論におけるモジュールの表現方法について説明し、さらにシステムを変更するためのルールとなるオペレータの仕様について説明する。

2.2 モジュールの表現方法 設計過程におけるモジュール i ($=sf_i$) は次式のように表現される。

$$sf_i = \begin{bmatrix} g1_i(x_1^{g1_i}, x_2^{g1_i}, \dots, x_{n1}^{g1_i}) \\ g2_i(x_1^{g2_i}, x_2^{g2_i}, \dots, x_{n2}^{g2_i}) \\ \vdots \\ gm_i(x_1^{gm_i}, x_2^{gm_i}, \dots, x_{nm}^{gm_i}) \end{bmatrix} \dots\dots\dots (1)$$

ここで、 $g_i(\dots)$ ($j=1, 2, \dots, m$) はモジュールの構成部品であり、外部構造と称する部分である。このとき

sf_i は、 m 個の外部構造から成っている。そして、各外部構造が保持する変数 x_k^i ($j=1, 2, \dots, m, k=1, 2, \dots, n_j$) は、外部構造の詳細を決定する内部変数と呼ぶものであり、各外部構造は n_j 個の設計変数を含んでいる。式(1)で表されるモジュールは、二層構造になっており、大枠を外部構造が決定し、内部変数は、その中身についての具体的な情報を提供するものである。

このように、モジュールの表現を、上位構造である外部構造と、その下位の内部変数に分けて記すことで、上位レベルからモジュールの概要を、また、下位レベルからその詳細を適宜知ることができる。よって、設計者がモジュール間の関係を知りたいとき、その頻度に応じて、どのレベルまでアクセスするかを選択できるので、設計過程におけるモジュール間の関係調査のための時間損失を、小さくすることができる。また、便宜上、モジュール同士を区別するために、外部構造とその内部変数のセットが同じモジュールについても、添字は異なるものとする。

2.3 オペレータについて 2.2節で述べた方法で表現されたモジュール群を用いて、システムを構築する。本論で構築される最初のシステム f_1 は、アルゴリズムの中で、まず、トップダウン法により得られる最初のモジュール（モジュール0）のみによって形成される。モジュール0は、対象とする問題を単純化、規格化してシステムを集約することによって得られるものである。ただし、 f_1 は実際にユーザのニーズを満たすためには小規模で抽象的なシステムである。そのため、逆に単純化したシステムに対し、自由度を増やしたり、設計変数を追加することにより複雑化し、実用的なシステムを作り出す作業が必要になる。この時、効率よくシステムの再構築を行うため、複雑化させるためのルールが必要である。

そこで、本研究では“オペレータ”という、システム構築の際にモジュールをコントロールする操作子を発案した。オペレータには、(1) システムを構成するモジュールを追加する“operator 1”、(2) モジュールの外部構造に含まれる内部変数を変更する“operator 2”、という二つの種類が存在する。

(1)の operator 1 は、システムを構成するモジュール群に新たにモジュールを追加し、システムの自由度を増加させ、より高次元のシステムを構築する。

(2)の operator 2 は、モジュールそのものの性格を変えるオペレータである。このオペレータは、モジュールを構成する部品ともいえる外部構造について、設計変数を追加したり、変更するものである。ただし、複雑なシステムも、基本的な要素の組合せで表現する

というモジュール性を維持するため、このオペレータによって全く異なる外部構造の提案をするものとしな。そのため、どのような形態で内部変数を追加するかは対象問題によって異なるので、モジュールの内部変数変更については、適用する問題に対し個別に具体化する必要がある。

以上の2種類のオペレータにより、多様なシステムをモジュールの組合せで表現することができる。図2は、1), 2) それぞれのオペレータが作用して、システムおよびモジュールが更新された例である。

図2(a)は、もともと P 個のモジュールで構成されていたシステムが、モジュール追加のオペレータによりモジュールを追加して、新たに $P+1$ 個のモジュールから成るシステムを構築するようすを示している。また、図2(b)は、内部変数変更オペレータの作用によって、性質の異なるモジュールを定義したこと

を示している。

また、operator 2 については、内部変数変更のパターンごとに設定するので、数種類定義することができる。ただし、新たに加えたモジュールや、変更後のモジュールが元のモジュール群とその表現形式に関して整合性を保っていることが必要である。また、システム全体がオペレータ作用後に初期設定で定めた評価関数で評価できないものに更新されてはならない。このため、モジュールの追加については、モジュール0と同じ構成のモジュールを追加するものとし、内部変数オペレータについては、同一の評価関数で評価可能にするための制約条件を設けるものとする。なお、適用例では、モジュール追加オペレータと、2種類の、外部構造が保持する内部変数の変更オペレータを定義した。

2.4 システムの構築

sf_0 を定義した後、モジュールの組合せによるシステムの構築を行う。このとき、設計の初期点となる一番最初に構築されるシステムは、 sf_0 のみからなるものとする。そして、このときできるシステムを f_1 とする。なお、添字は構築されたシステムの通し番号である。その後、オペレータの作用により、 $f_2, f_3, \dots$ とシステムが構築される。この手順でシステムを構築していくと、すべてのシステムは、オペレータの性質から、モジュール0 (sf_0) から派生したモジュール群となる。このため、構築されるすべてのシステムにおけるモジュールの基本構造を統一することができ、オペレータの作用による問題の大規模化を解消している。図3は、三つのモジュールからなるシステムの例である。

このシステム f_3 は、オペレータによってモジュール数の変更と内部変数の変更により、更新された三つのモジュール sf_1, sf_2, sf_3 で構成されている。そして、このシステム f_3 がどのようなモジュールで構成されているかを明確にするために、 f_3 を次式のように表現する。

(a) Change in the system by operator 1 induced module composition change

(b) Change in the modules by operator 2 induced changes in design variables

Fig. 2 Change in system and modules specifications by the action of operators

Fig. 3 Representation of system f_3 consisting of 3 modules

$f_3: (sf_1, sf_2, sf_3) \dots\dots\dots(2)$

3. モジュールを用いた コンカレント設計法

本研究の目的は、機械製品に対するコンカレント設計の具体化である。そのため、従来の設計法のような一方向流れの設計の枠組みを解消するため、先に述べたボトムアップとトップダウンの双方向の流れが必要になる。よって、本論では、まず、対象とする問題から小システムを定義し、それを一般化することで、モジュール0を作成する。これは、トップダウン的な作業である。そして次に、アルゴリズム中で、オペレータの作用により、いくつものシステムを構築し、評価するというボトムアップの作業を行う。この手順をアルゴリズムの中で行うことで、コンカレント設計を実現した。本方法論のフローチャートを図4に示す。

この設計の流れは、データベースを用いた、一般的な最適化アルゴリズムに、モジュールによるシステム形成の効果を考慮したものである。

まず、ユーザからの要求により、問題提起が行われる。それに基づき、初期設定として、対象システムの簡略化と、低次元化を施すことにより、モジュール0 (sf_0) を作成する。ここで提案される対象システムは、

Fig. 4 Flowchart for concurrent design method using modules

類似した要素の集合体と考えられるものであり、 sf_0 は、類似した要素すべてに共通な構造と設計変数を保持するものである。そして、低次元化のため、以降に生成される、あらゆるモジュールの中で最もシンプルなものではなくてはならない。

sf_0 の決定後、そのモジュールの形態に沿った評価関数の設定、オペレータに対する制約条件の決定、オペレータの決定を行う。評価関数はモジュール数に依存しない形で、すべての構築されるシステムを平等に評価できる指標として定義する。また、オペレータに対する制約条件とは、オペレータの作用によってシステムが高次元化、および複雑化するレベルに上限を設けるためのものである。また、オペレータは、モジュールの追加とモジュールの内部変数の変更の二つを定義する。ここで、内部変数の変更形態は問題ごとに設定する。ただし、その変更形態は、オペレータを作用させてできた新システムにも、もとのシステムに対すると同様な評価関数を適用できるものに限る、という制約を設ける。

次に、そのモジュール0 (sf_0) のみによる初代システム f_1 を作成し、 f_1 に対する解析、最適化を施す。解析では、構築されたシステムがどのような性質をもっているかを調べる。また、この段階で評価関数に必要なシステムの数理解釈を得る。そして、最適化の段階で、評価関数の値について最良のものとなる内部変数の決定を行う。

上記の段階を経た後、データベースに、システムの形状や、解析の結果、評価関数の値を保存する。システムの表現法は、2・4 節で述べたシステムの表現を用いるものとする。また、どのシステムから派生したシ

Fig. 5 Data saved in tree form

システムがわかるように、生成されたシステム順に図5に示すように、ツリー形式でデータを保存するものとする。これは、どのシステムから如何なるシステムが派生したかを視覚的に明示するものであり、設計者が次のシステムを提案する際の重要なデータである。また、各枝中に示される operator $i(i=1, 2)$ は、オペレータ i の作用によってシステムが更新されたことを示す。

次に、システム検証について説明する。これは、先ほど登録したデータベースに、設計者がアクセスしてどのような解が得られたかを調べる作業である。この結果を検証することにより、設計者は、オペレータの付与によるシステムの再構築を行う。これにより、新しい世代のシステムが生成される。ただし、初期設定で定義した制約を違反したり、オペレータによりシステムを更新すると他のシステムと重複する場合は、オペレータを作用させず、そのシステムはツリーの末端となる。

このように、ツリーの成長が止まるまで、システムは再構築され、多様なシステムが評価されることになる。そして、それらすべてのシステム群のもつ解から、最良の解を選び、それを対象とする問題に対する最適解として、アルゴリズムは終了する。

4. 適用例：物体搬送用 マニピュレータの設計

本章では、本論における方法論を産業用マニピュレータの設計に適用する。ここで設計されるマニピュレータは、決められた動作時間内で、同一平面内に存在する点 A から点 B まで物体を搬送する、多リンク機構とする。また、制御方式としては、点から点への追従を実現する PTP 制御⁽⁵⁾ とする。

4.1 設計過程 以下では、3章で述べたアルゴリズムに基づいた設計過程を記す。

4.1.1 ユーザ要求 ユーザからの要求は、次の2点とする。

- (1) 消費エネルギーの最小化
- (2) リンク機構の操作性の最大化

4.1.2 初期設定 ここでは、(1)モジュールの決定、(2)評価関数の決定、(3)制約条件の決定、(4)オペレータの決定、の四つの設定を行う。ただし、これらの設定の前に、本問題に対するさまざまな数値設定をしているが、それらについては4.2.1項で説明する。

(1) モジュールの決定 本論では、リンク機構の最小単位として、1モータ1リンク機構をモジュール

0($=sf_0$) とし、 sf_0 を次式のように表す。

$$sf_0 = \begin{bmatrix} M_0(G_0, Mm_0, Rm_0, Ke_0, Kt_0) \\ Ar_0(l_0, m_0, E_0) \\ H_0(Wh) \end{bmatrix} \cdots \cdots (3)$$

ここで、

$M()$: モータ (G : 減速比, Mm : モータ慣性モーメント, Rm : 電機子抵抗, Ke : 誘起電圧定数, Kt : トルク定数)

$Ar()$: リンク (l : リンク長, m : リンク質量, E : ヤング率)

$H()$: 手先効果器 (Wh : 効果器質量)

とする。これらの内部変数はリンク機構における必要最低限のものである。このモジュールを、システム構築の開始点とする。 sf_0 の構造は式(3)に示すように、外部構造にモータ、リンク、手先効果器の三つを定める。なお、今回は手先部分の設計を行わないので、手先効果器の保持する内部変数はその質量のみとする。また、 $H()$ は、複数のモジュールが存在する場合、最も先端のリンクのみに付与する。そして、内部変数には、各外部構造にかかわる設計変数が記述されている。

(2) 評価関数の決定 sf_0 の仕様が決定された後、 sf_0 を基に構築されるシステムがユーザ要求をどれだけ満たしているかを評価する基準を設ける。この基準が評価関数であり、ユーザ要求から、消費エネルギーと操作性についての評価関数を作成する。まず、リンク機構では一般的な動的可操作度⁽⁶⁾ を操作性の指標として用い、そして、加えられたパワーの量で消費エネルギーを評価するものとする。動的可操作度は、その値が大きいほど操作性が高いとされる。また、本研究では、モジュール群の組合せからなるすべてのシステムに対して、同じ形の評価関数を用いるため、評価関数は次元に関する制約や、あるシステムのみに特化したものではない、一般的な関数として定義する。

したがって、消費エネルギーと操作性に関する二つの要求は、式(4)、(5)のように書ける。式(4)は、モータ i の消費エネルギーをすべてのモータについて足し合せた評価関数 ϕ_1 の最小化であり、式(5)は、動作中のいくつかの観測点における動的可操作度 ω_{d_j} の逆数の和からなる評価関数 ϕ_2 の最小化と、表すことができる。

$$\phi_1 = \sum_{i=1}^n \frac{R_{mi}}{K_{ti}^2} \int_0^T \tau_i^2 dt \rightarrow \text{最小化} \cdots \cdots (4)$$

$$\phi_2 = \sum_{j=1}^k \frac{1}{\omega_{d_j}} \rightarrow \text{最小化} \cdots \cdots (5)$$

ここで、 R_m は電機子抵抗、 K_t はトルク定数、 τ はリンクにかかるトルクであり、 T は動作時間である。また、式(4)における添字 i はリンクの番号であり、式(5)における添字 j は、可操作度の観測点の番号を示し、ここでは k 個の観測点を設けている。

(3) 制約条件の設定 本設計法では、オペレータによってモジュールの組合せや形態に変化を与えるため、設計過程で問題が大規模化したり、整理された形にならないため、的確な評価を与えられないという可能性がある。そのため、オペレータの作用に関して制約を設けることが必要である。ここでは、リンクの自由度の大きさの上限を制約にする。マニピュレータの駆動は平面内に限定しているため、自由度は2で十分であり、低コスト化の観点から冗長性は考えないものとする。すなわち、システムを構築するモジュールは、二つまでということである。また、内部変数の変更については、作用後のシステムが、先に決めた評価関数で記述できるものに限定する。すなわち、適用例では、モータ駆動で、運動方程式で表現可能であり、可操作度を定めることのできるシステムに限定するということである。

(4) オペレータの決定 オペレータは、モジュール群の組合せ変更(operator 1)と、システムの内部変数の変更(operator 2)を行う。ただし、operator 1については、制約条件から、1回までの使用となる。また、operator 2については、問題に対し、個別に設定する必要がある。マニピュレータ設計の例においては、評価関数の式形態を変化させないものとして、内部変数の変更は、1) リンク内の関節数の変化(operator 2-1)と、2) リンク形状の変更(operator 2-2)、の二つとする。これらが作用してシステムが変化する例を、図6に示す。

まず、システム f_1 から f_2 への変化は、モジュール増加のオペレータが作用した例で、オペレータの作用前は、1リンク機構であったのに対し、作用後はモジュールが二つになることで、自由度2の直列駆動形アーム機構が得られる。また、 f_2 から f_3 への変化は、モジュールのリンクに関する内部変数を変化させた例で、リンクに関節をもたせるoperator 2-1を作用させることにより、直列駆動から、台座に二つのモータを配した並列駆動形マニピュレータに変化させることができる。

このようにオペレータを用いることで、複雑なシステムも、単純なモジュールによる構成で、系統的に表現することができる。

Fig. 6 Examples of operator actions

4.1.3 システム構築 初期設定の後、アルゴリズム中で一番最初に構築されるシステム f_1 は、モジュール0のみからなるものである。それ以降において構築されるシステムは、オペレータの作用により、さまざまな形態をとる。

4.1.4 システムの解析 解析段階では、システム構築で得られた新システムがどのような運動特性をもっているかを調べる。本適用例では、自由度や機構の変化による、トルクと動的可操作度の変化を調べる。

4.1.5 最適化 ここでは多点探索法を用いた。これは、内部設計変数の領域を等分割し、その領域内で、いくつか点を選び計算し、結果を求める。そして、結果のよい点を含む領域でさらに細かく分割し、最良解を探していくという手順である。また、この方法で局所解に陥ることのないように、探索点をできるだけたくさん設けた。この方法は、変数が多種多様の場合困難であるが、本論のような、モジュールによって規格化されたシステムでは、十分適用可能である。

Fig. 7 The tree-form data for the design example

4.2 シミュレーション

4.2.1 設定内容 上記の方法に基づいて、計算機上で Mathematica⁷⁾ を用いてシミュレーションを行った。リンクについては、素材にアルミニウムを想定し、断面形状は口形の中空角柱を使用した。モータについては、回転数は一律 3 000 回転で、定格出力の異なるものを四つ用意した。また、動作距離は A-B 2 点間で 0.55 m で動作時間を 1.0 秒とした。動作速度については、前半 0.50 秒まで一定の加速度で加速し、後半同様に減速する、三角速度曲線に基づいて設定した。また、手先効果器と搬送物を含めたときの荷重を 9.8 N (1 kgf) とした。ここでは、低コスト化を目指すことから冗長性は認めないものとするので、平面内動作において、自由度 3 以上のリンク機構については評価対象外とする。

4.2.2 計算結果 図 7 は、データベース内に作られるツリー図であり、図 8 は、計算結果から得られた各システムのパレート最適解集合である。

図 7 のツリー形成履歴について説明する。図 7 中のノード間に存在する operator $i(i=1, 2-1, 2-2)$ は、作用させたオペレータを示す。operator 1 はモジュールの組合せを変更するオペレータのことで、operator 2-1 は内部変数変更オペレータのうち、リンクの関節数を変化させるものである。そして operator 2-2 は、質量中心位置を変更する内部変数変更オペレータである。

まず単一リンク機構であった f_1 から、モジュールの追加により直列駆動形のリンク機構 f_2 が得られる。そして、さらに operator 2-1 を作用させたものが、並列駆動形リンク機構 f_3 である。また、 f_2 に operator

Fig. 8 Changes in Pareto solutions for the example

2-2 を作用させ、リンクの断面形状を変化させることによってリンクの質量中心位置を変化させたものが f_4 である。 f_3, f_4 の評価の後、 f_3 は、さらに operator 2-2 を作用させて f_5 を得るが、 f_4 にはいずれのオペレータを作用させてもほかのシステムと重複するためここで末端となる。同じ理由で、 f_5 も末端となる。この結果、図 7 のツリーが得られる。

次に、最適化のプロセスを図 8 に示す。図 8 は、二つの評価関数 ϕ_1 と ϕ_2 からなる空間での、各システムのパレート最適解集合である。これら二つの評価関数の値は式 (4)、(5) から、ともに小さいほうがよいので、図 8 において、解は原点に近いほど望ましい。ただし、 f_1 のパレート図は、 f_1 の構造上、評価関数 ϕ_2 の値が常に無限大であるため記載していない。また、 f_2 については、モータ外部構造内における離散変数を含めた最適化の状況を、4 タイプのモータごとに分けて図示した。ほかのシステムについては、その中で最も良好なパレート解だけを記した。図 8 から解領域は $f_2 \rightarrow f_4 \rightarrow f_3 \rightarrow f_5$ の順に良好な傾向にあることがわかる。このとき、図 7 では、このようすを枝をたどることによって、どのようなシステムから良好な解が得られたか、という履歴を知ることができる。また、図 8 から、この問題に対する最適解集合は f_5 であることがわかる。この最良なシステムは 2 リンク機構の並列駆動形マニピュレータで、モータの自重による負荷をリンク上からなくしたことで、直列駆動形マニピュレータよりも良好な結果が得られたものと思われる。

5. 結 言

本研究では、コンカレント設計を実現するために、オペレータを作用させたモジュールの概念を用いた設計法を提唱した。これより、以下の結論を得た。

(1) 逐次設計法のような、現在一般に用いられている設計法では困難な、多様な解の同時評価を、モジ

ジュールを用いたシステム設計により実現することができた。これにより、さまざまな分野や領域における設計変数を同時に扱うことができ、大規模、または複雑なシステムにおけるコンカレント設計の具体的な手法を構築した。

(2) モジュール群の組合せによるシステム構築に関して、問題の大規模化を防ぐためにモジュールによって規格化を行い、システムを単純化した。また、逆に、システムの多様化を実現するため、オペレータを提案し、モジュールの構造と組合せを変更させた。これらシステムの単純化とシステムの多様化という、二つの矛盾する内容は、モジュールを、外部構造と内部変数に別れた二層構造で表現する手法など解消させた。

文 献

- (1) Yoshimura, M. and Takeuchi, A., Concurrent Optimization of Product Design and Manufacturing Based on Information of User's Needs, *Int. J. Concurrent Eng. : Research Appl.*, 2 (1994), 33-44.
- (2) R. E. Douglas Jr, and D. C. Brown (Parsaei H. R. and Sullivan, W.G. 編), Concurrent Accumulation of Knowledge in view of Concurrent Engineering, *Concurrent Engineering* (1993), 402-412, Chapman & Hall.
- (3) 日本機械学会編, 適応化・知能化・最適化法, (1996), 60-65, 技報堂出版.
- (4) 田村担之編, 大規模システム-モデリング・制御・意思決定-, (1986), 2-10, 昭光堂.
- (5) 川崎晴久, ロボット工学の基礎, (1991), 6-7, 森北出版.
- (6) 吉川恒夫, ロボット制御基礎論, (1988), 109-131, コロナ社.
- (7) Gray, W., Mathematica 方法と応用, (1996), 143-350, サイエンティスト社.