数理解析研究所講究録
1085巻 1999年 49-59

制約評価技術に基づく初期設計支援システムの開発

機械技術研究所 沢田　浩之 (Hiroyuki SAWADA) *

概　要

This paper describes development of the preliminary design support system. It supports the embodiment design stage in engineering design by providing a constraint solver which solves constraints generated during design process. The constraint solver provides the following functions: (1) detecting inequalities which cannot be simultaneously satisfied, (2) finding out variables whose values should be changed in order to remove inconsistency, (3) computing numerical solutions. Furthermore, in order to support defining constraints, a component library is provided.

1　はじめに

一般に設計とは、与えられた要求仕様を満足する製品を作り出すために必要なすべての情報を生成する活動として定義される [1]。この与えられた要求仕様を製品が満足すべき制約条件とみなすことにより、設計問題を制約充足問題として定式化することが可能である。著者は、このような観点に基づき、主にメカトロニクス分野を対象とした初期設計支援システムの開発を進めている [3]。ここでは、当初期設計支援システム開発に関し、その基礎となる設計プロセスモデル、設計支援手法および現時点における課題について述べる。

本稿の構成は以下の通りである。まず、第2章で本研究で用いている設計プロセスモデルについて述べる。次に第3章において、本研究における設計支援手法について説明するとともにその応用例を示す。第4章では、現時点における課題である制約条件の記述方法に関する問題点を挙げ、その解決方法について述べる。

2　設計プロセスモデル

設計は人間の複雑な知的活動であり、その活動形態の少なからぬ部分は設計者個人の思考形態に依存する。したがって、万人の設計活動すべてを包括的に支援するようなソフトウェアシステムを作成することは不可能であり、むしろ、ある限定された側面を捉え、その範囲内でのみ有効な支援手段を提供することが現実的なアプローチと考えられる。それ

*hiroyuki.sawada@mel.go.jp

には、設計問題をある特定の観点に基づいて定式化し、その定式化に基づいて設計過程をモデル化した設計プロセスモデルを構築することが重要である。設計プロセスモデルは、作成する設計支援システムの位置付およびその支援対象を明確にするものである。本章では、設計問題を制約充足問題として定式化することによって構築された設計プロセスモデルについて述べる。

設計問題を制約充足問題として定式化するとき、設計対象製品は、その製品を規定する制約条件の集合として表現される。この制約条件集合を製品モデルと呼ぶことにする。設計の初期段階では、製品を規定するものは要求仕様しか存在せず、製品モデルは具体的な寸法等を持たない抽象的なものである。設計過程を経るにしたがって製品に関する制約条件が追加され、製品モデルは次第に具体化する。そして最終的にはすべての設計変数に値が割り当てられ、製品モデルは完全に具体化する。すなわち設計プロセスは、制約条件を付加することによって製品モデルを抽象的なものから具体的なものへと発展させる過程として捉えられる。

図1に設計プロセスモデルを示す。初期状態では要求仕様のみが存在する。その初期状

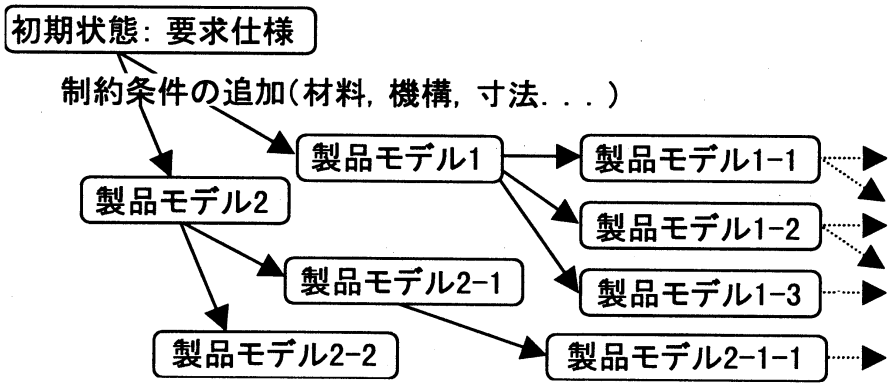


図 1: 設計プロセスモデル

態に対して、使用される材料の特定、機構の決定、寸法値の割り当てといった制約条件が追加されることにより、製品モデルは次第に具体化される。追加される制約条件に選択肢が存在する場合には分岐が生じ、異なる製品モデルが生成される。設計過程では、このようにして数々の製品モデルが生成され、それらの中から最適と評価されたものが最終的な設計案として採用される。次章では、この設計プロセスモデルに基づいた設計支援手法について述べる。

3 設計支援手法

図1によってモデル化された設計プロセスを支援するためには、設計支援システムに以下の機能が求められる。

1. 設計プロセス管理機能
   製品モデルが生成される設計プロセスの管理を行う機能。これはすなわち、生成された製品モデルを整理／分類する機能である。これによって、設計者は自分自身の設計作業履歴をたどることができるとともに、設計案の整理を行うことができる。
2. 製品モデル評価機能
   設計プロセスにおいて生成される製品モデルの評価を行う機能。これはすなわち、製品モデルを表現する制約集合の妥当性を判断する機能である。通常、設計過程において生成される設計案は、すべての設計変数値が決定されて完全に具体化されるよりも前の段階で評価される。この機能は、未知の設計変数が含まれた不完全な設計案の妥当性を判断し、不適切な設計案を早い段階で排除するためのものである。

本稿では、上記機能のうち「2. 製品モデル評価機能」についてのみ議論を行う。次節では、製品モデル評価機能を提供する代数制約評価系について述べる。

3.1 代数制約評価系

メカトロニクス設計における制約条件の多くは、実数係数多項式の方程式もしくは不等式として表現可能であり、製品モデルは代数制約集合として表現される。すでに述べたように、設計過程において評価される製品モデルには未知の設計変数が含まれているのが通常である。したがって、製品モデルを表現する代数制約集合は、過少制約問題を与えることになる。

過少制約問題とは、設計変数の数に比べて方程式の数が少ないために解が一意に定まらない問題を指す。通常、過少制約問題の解は無限個存在するが、一方、それら無限個の解の中にすべての不等式を満足するものが存在しないこともあり得る。従来、過少制約問題は試行錯誤によって解かれることが多く、存在する解を見過ごす可能性や存在しない解を探し求めることに労力を費やしてしまう可能性は否めない。代数制約評価系 [4] は、そのような過少代数制約集合を適切に評価するために開発されたものであり、以下の機能を持つ。

1. 不等式制約間の矛盾検出。
2. 矛盾を解消するために値を変更すべき設計変数の特定。
3. 数値解の計算。

以下、第 3.1.1 項から第 3.1.3 項において、それぞれの機能を実現する手法について述べる。

3.1.1 不等式制約間の矛盾検出法

与えられた代数制約集合を式 (1) とする。

$$\begin{aligned} &f_1(\boldsymbol{x}) = 0, \ldots, f_p(\boldsymbol{x}) = 0, \\ &g_1(\boldsymbol{x}) \neq 0, \ldots, g_q(\boldsymbol{x}) \neq 0, \\ &h_1(\boldsymbol{x}) \geq 0, \ldots, h_r(\boldsymbol{x}) \geq 0. \end{aligned} \tag{1}$$

ただし、$\boldsymbol{x} = (x_1, \ldots, x_n)$ は n 次元実数変数、$f_i(\boldsymbol{x}), g_j(\boldsymbol{x}), h_k(\boldsymbol{x})$ は実数係数多項式である。また、等号を含まない不等式 $e(\boldsymbol{x}) > 0$ は、$e(\boldsymbol{x}) \neq 0 \wedge e(\boldsymbol{x}) \geq 0$ と表されるものとする。

スラック変数 $s_1, \ldots, s_q, t_1, \ldots, t_r$ を導入することにより、式 (1) を式 (2) へ等値変換する。

$$\begin{aligned} &f_1(\boldsymbol{x}) = 0, \ldots, f_p(\boldsymbol{x}) = 0, \\ &g_1(\boldsymbol{x}) \cdot s_1 = 1, \ldots, g_q(\boldsymbol{x}) \cdot s_q = 1, \\ &h_1(\boldsymbol{x}) = t_1, \ldots, h_r(\boldsymbol{x}) = t_r, \\ &t_1 \geq 0, \ldots, t_r \geq 0. \end{aligned} \tag{2}$$

式 (2) を用いてスラック変数 $t_1, \ldots, t_r$ のみからなる方程式を導く。これを式 (3) とする。

$$c(t_1, \ldots, t_r) = 0. \tag{3}$$

式 (3) の係数と定数項を調べることにより、以下の規準にしたがって矛盾を検出できる。

矛盾検出の規準

1. 多項式 $c(t_1, \ldots, t_r)$ のすべての係数が正で、かつ、定数項が正であるならば、制約集合 (1) には同時に満足させることのできない不等式が存在する。
   例えば、$c(t_1, \ldots, t_r) = t_1 + t_2 t_3 + 3$ である場合、スラック変数 t_1, t_2, t_3 に対応する不等式 $h_1(\boldsymbol{x}) \geq 0, h_2(\boldsymbol{x}) \geq 0, h_3(\boldsymbol{x}) \geq 0$ を同時に満足させることはできない。
2. 多項式 $c(t_1, \ldots, t_r)$ のすべての係数が正で、かつ、定数項が 0 であるならば、制約集合 (1) には方程式で置き換えることのできる不等式が存在する。
   例えば、$c(t_1, \ldots, t_r) = t_1 + t_2 t_3$ である場合、スラック変数 t_1, t_2, t_3 に対応する不等式 $h_1(\boldsymbol{x}) \geq 0, h_2(\boldsymbol{x}) \geq 0, h_3(\boldsymbol{x}) \geq 0$ は、以下の方程式で置き換えることができる。

$$h_1(\boldsymbol{x}) = 0 \wedge \{h_2(\boldsymbol{x}) = 0 \vee h_3(\boldsymbol{x}) = 0\}.$$

∎

3.1.2 矛盾を解消するために値を変更すべき設計変数の特定法

同時に満足させることのできない不等式を示すとともに、そのような矛盾を解決する方法を示唆することは、設計作業を支援する上で有効な手段となる。本項では、矛盾を解消するために値を変更すべき設計変数の特定法について述べる。

式 (3) によって矛盾が示されているものとする。また、" 設計変数 x_k に値 a が与えられている" ことは、通常、一変数線形方程式 (4) で表される。

$$x_k = a. \tag{4}$$

したがって、式 (3) を導くために式 (4) が必要であるならば、式 (4) は式 (3) で示される矛盾と関わりがあることになり、設計変数 x_k の値 a を変更することによって矛盾を解消できる可能性があると考えられる。一方、式 (4) が式 (3) を導くために不要であれば、式 (4) は式 (3) で示される矛盾とは無関係であり、設計変数 x_k の値を変更する必要はない。

式 (4) が式 (3) を導くために必要か不要かは、以下の方法で調べることができる。式 (2) から式 (4) を除いた連立方程式によって定義されるイデアルを I とする。式 (3) が I に属していないならば、式 (4) は式 (3) を導くために必要であると結論される。これに対して、式 (3) が I に属しているならば、式 (4) が存在しなくても式 (3) を導くことができることになり、式 (4) は不要であると結論される。したがって、I のグレブナ基底 G を用いることにより、式 (4) が式 (3) で示される矛盾と関係があるかないかを判定することができる。

式 (4) と式 (3) で示される矛盾との関係

式 (2) から式 (4) を取り除いた連立方程式によって定義されるグレブナ基底を G とする。

1. G が $c(t_1, \ldots, t_r)$ を 0 に簡約化しない。
   ⇒ 式 (4) は式 (3) で示される矛盾と関わりがあり、x_k の値を変更することによってこの矛盾を解消できる可能性がある。
2. G が $c(t_1, \ldots, t_r)$ を 0 に簡約化する。
   ⇒ 式 (4) は式 (3) で示される矛盾とは無関係であり、x_k の値を変更する必要はない。

∎

3.1.3 数値解の計算法

過少制約問題の解は一般に無限個存在するため、すべての数値解を求めることは不可能であり、それらのうちのいくつかを例として計算することになる。ここでは、過少制約問題を条件付き最小値問題に帰着させることによってその数値解を計算する方法について述べる。

式 (2) によって定義される $(n+q+r)$ 次元実数空間における領域を A とする。

$$\begin{aligned} A \;=\; \{(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t}) | & f_1(\boldsymbol{x}) = 0, \ldots, f_p(\boldsymbol{x}) = 0, \\ & g_1(\boldsymbol{x}) \cdot s_1 = 1, \ldots, g_q(\boldsymbol{x}) \cdot s_q = 1, \\ & h_1(\boldsymbol{x}) = t_1, \ldots, h_r(\boldsymbol{x}) = t_r, \\ & t_1 \geq 0, \ldots, t_r \geq 0\}. \end{aligned} \tag{5}$$

ここで、以下の条件を満足する目的関数 $u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ を導入する。

目的関数 $u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$

1. $u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ は $(n+q+r)$ 次元実数空間において最小値を持つ。
2. $u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ は $(n+q+r)$ 次元実数空間において連続である。
3. $u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ は、無限遠方においてプラス無限大に発散する。

∎

領域 A の境界はすべて A に含まれているので、領域 A が空でない場合、目的関数 $u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ は領域 A において最小値を持つ。したがって、領域 A における目的関数 $u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ の最小値を計算することにより、式 (2) の解が目的関数 $u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ の最小点の座標値として求められる。最小値が存在しない場合、それは領域 A が空であることを意味する。

3.2 代数制約評価系の応用例

代数制約評価系を製品モデルの評価に応用した簡単な例として、図 2 に示す直動関節型ロボットアームの設計を取りあげる。このロボットアームは、電源、直流モーター、ボー

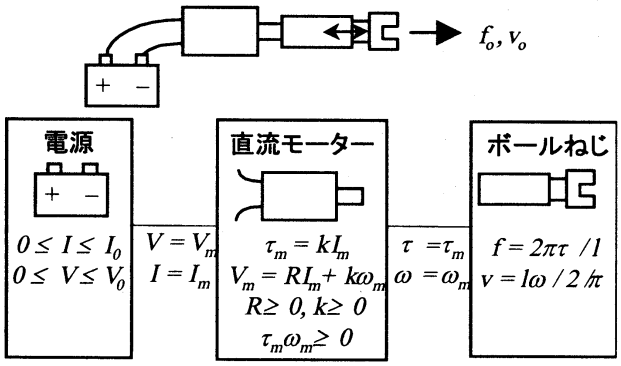


図 2: 直動関節型ロボットアーム

ルねじから構成され、その製品モデルを表現する制約集合は、以下に示すように、各機械要素に関する制約条件と個々の要素を接続するための条件の和集合として与えられる。

1. 電源

   $0 \le I \le I_0,\ 0 \le V \le V_0.$

   I: 出力電流、I_0: 出力電流の上限値、V: 出力電圧、V_0: 出力電圧の上限値。
2. 直流モーター

   $\tau_m = kI_m,\ V_m = RI_m + k\omega_m, R \ge 0,\ k \ge 0,\ \tau_m\omega_m \ge 0.$

   τ_m: 出力トルク、ω_m: 出力角速度、I_m: 入力電流、V_m: 入力電圧、k: トルク定数、

R: 電気抵抗。

3. ボールねじ

   $f = 2\pi\tau/l,\ v = l\omega/2/\pi.$

   f: 力、v: 速度、τ: 入力トルク、ω: 入力角速度、l: リード。

4. 電源と直流モーターの接続

   $I_m = I,\ V_m = V.$

5. 直流モーターとボールねじの接続

   $\tau = \tau_m,\ \omega = \omega_m.$

6. ボールねじ出力とロボットアーム出力の関係

   $f_o = f,\ v_o = v.$

ここでは、各要素の特性値を以下のように固定し、ロボットアームの力 f_o と速度 v_o に関する制約条件を変化させ、それぞれの場合における代数制約評価系の評価結果を見ることにする。

各要素の特性値

電源: $I_0 = 1[\mathrm{A}],\ V_0 = 9[\mathrm{V}].$

直流モーター: $k = 0.02[\mathrm{Nm/A}],\ R = 2[\Omega].$

ボールねじ: $l = 0.001[\mathrm{m}].$

ロボットアームの力 f_o と速度 v_o に関する制約条件

1. $f_o > 90[\mathrm{N}],\ v_o > 0.07[\mathrm{m/s}]$ の場合

   代数制約評価系は、以下の制約条件間に存在する矛盾を検出する。

   $$f_o > 90, v_o > 0.07, V \leq V_0, V_0 = 9, R = 2, k = 0.02, l = 0.001.$$

   設計者には、これらの条件のうちの少なくとも一つを緩和することが求められる。すなわち、ロボットアームの力と速度に対する要求を緩めるか、電源をより出力電圧の高いものに取り替えるか、直流モーターあるいはボールねじを別のものにするかである。電源電流の上限値は、この場合関係ない。

2. $f_o > 90[\mathrm{N}],\ v_o > 0.06[\mathrm{m/s}]$ の場合

   代数制約評価系は、可能な設計解として以下の数値解を計算する。

   $$f_o = f = 90.05, v_o = v = 6.005 \times 10^{-2}, \tau = \tau_m = 1.441 \times 10^{-2},$$
   $$\omega = \omega_m = 3.753 \times 10^2, I_m = I = 0.7204, V_m = V = 8.947.$$

   設計者はこの数値解を参照し、より望ましい解を得るために数値の変更や制約条件の追加を行う。

現在開発を進めている初期設計支援システムの概観を図 3 に示す。開発は Windows 95 上で Visual C++を用いて行われており、代数制約評価系のインプリメントには Risa/Asir[2]

を使っている。主ウィンドウと Risa/Asir 間の通信は、Windows のメッセージ関数によるデータ送受信およびファイル入出力によって行われている。

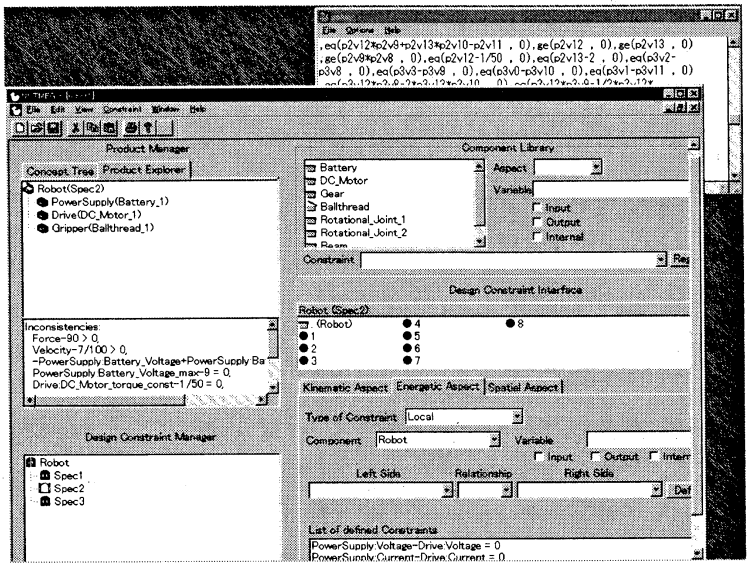


図 3: 初期設計支援システムの概観

4 制約条件の記述に関する問題点

本初期設計支援システムは、代数制約集合として定義された製品モデルの管理／分類／評価を行う環境および機能を提供するものであり、制約条件の定義／生成は設計者に任されている。その際、製品モデルを表現するためには多くの制約条件を記述する必要がある。さらに、代数制約評価系ではグレブナ基底に関する計算を行っているため、変数順序ならびに項順序の設定を行う必要がある。項順序に関しては、一般に全次数逆辞書式順序が最も計算効率がよいとされているが、変数を入力変数と出力変数とに分類できる場合には、(出力変数) > (入力変数) に関する辞書式順序を採用した方が効率的であることはよく知られている。

代数制約式をすべて手作業で記述することは設計者にとって大きな負担であり、また、グレブナ基底理論に関する知識を持ち合わせていない設計者がほとんどであるので、本初期設計支援システムを実際の設計作業に用いるためには、制約条件の記述支援ならびに変数順序／項順序の自動設定機能の提供が必要不可欠である。本稿では、これらの課題の解決を目的として開発中の Component Library について述べる。

4.1 Component Library

Component Library とは、多く用いられる基本的な機械要素について、その機能を表現する制約式ならびに変数順序／項順序をあらかじめ記述しておくものである。設計者は Component Library から必要な機械要素モデルを取り出し、それらの間の接続条件のみを定義する。あとはシステムが自動的に制約式の生成と変数順序／項順序の設定を行う。Component Library に登録される各機械要素モデルは以下のように構成される。

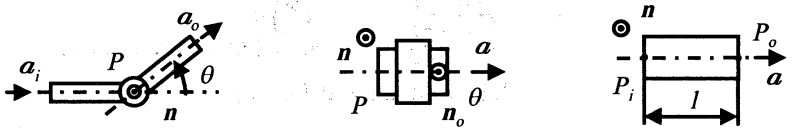
入出力変数 外部との入出力を表す変数。

状態変数 機械要素モデルの特性や状態を表現するために用いられる変数。

制約式 変数間の制約式。

変数順序 項順序はブロック順序を採用。(出力変数) > (入力変数) > (状態変数) に関する辞書式順序とする。また、出力変数、入力変数、状態変数それぞれのブロックの中では、全次数逆辞書式順序を用いている。

図 4 に、機械要素モデルの例を示す。各機械要素モデルの構成は以下の通りである。



(a) Rotational Joint 1　(b) Rotational Joint 2　(c) Beam

図 4: 機械要素モデル

(a) Rotational Joint 1

入出力変数 $\boldsymbol{a_i}, \boldsymbol{a_o}$: 入出力軸方向ベクトル、$\boldsymbol{n}$: 関節回転軸方向ベクトル、$P$: 関節位置。

状態変数 θ: 関節回転角。

制約式 $|\boldsymbol{a_i}| = |\boldsymbol{a_o}| = |\boldsymbol{n}| = 1, \boldsymbol{a_i} \times \boldsymbol{a_o} = \boldsymbol{n} \sin\theta, \boldsymbol{a_i} \cdot \boldsymbol{a_o} = \cos\theta.$

変数順序 $\boldsymbol{a_o} > \boldsymbol{a_i}, \boldsymbol{n}, P > \sin\theta, \cos\theta, \theta.$

(b) Rotational Joint 2（入出力軸回りの回転関節）

入出力変数 $\boldsymbol{a}$: 入出力軸方向ベクトル、$\boldsymbol{n_i}$: 入力軸に固定された $\boldsymbol{a}$ に垂直なベクトル、$\boldsymbol{n_o}$: 出力軸に固定された $\boldsymbol{a}$ に垂直なベクトル、P: 関節位置。

状態変数 θ: 関節回転角。

制約式 $|\boldsymbol{a}| = |\boldsymbol{n_i}| = |\boldsymbol{n_o}| = 1, \boldsymbol{a} \cdot \boldsymbol{n_i} = 0, \boldsymbol{n_i} \times \boldsymbol{n_o} = \boldsymbol{a} \sin\theta, \boldsymbol{n} \cdot \boldsymbol{n_o} = \cos\theta.$

変数順序 $\boldsymbol{n_o} > \boldsymbol{n_i}, \boldsymbol{a}, P > \sin\theta, \cos\theta, \theta.$

(c) Beam

入出変数 $\boldsymbol{a}$: 軸方向ベクトル、$\boldsymbol{n}$: $\boldsymbol{a}$ に垂直なベクトル、P_i, P_o: 両端点。

状態変数 l: 棒材長さ。

制約式 $|\boldsymbol{a}| = |\boldsymbol{n}| = 1, \boldsymbol{a} \cdot \boldsymbol{n} = 0, \overrightarrow{P_i P_o} = l\boldsymbol{a}.$

変数順序 $P_o > \boldsymbol{a}, \boldsymbol{n}, P_i > l.$

図5に示すロボットアームを構成する場合、その制約集合は Component Library に登録されている個々の機械要素に関する制約式と、それらの機械要素を接続するための条件との和集合として求められる。この例の場合、接続位置における2つの機械要素の位置と姿勢が等しくなるということが接続条件であり、具体的には以下のようになる。

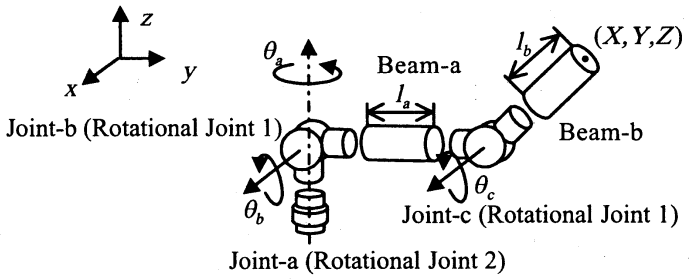


図 5: ロボットアーム

1. Joint-a と基準座標
   $\boldsymbol{a}(\text{Joint-a}) = (0,0,1), \boldsymbol{n_i}(\text{Joint-a}) = (1,0,0), P(\text{Joint-a}) = (0,0,0).$
2. Joint-b と Joint-a
   $\boldsymbol{a_i}(\text{Joint-b}) = \boldsymbol{a}(\text{Joint-a}), \boldsymbol{n}(\text{Joint-b}) = \boldsymbol{n_o}(\text{Joint-a}), P(\text{Joint-b}) = P(\text{Joint-a}).$
3. Beam-a と Joint-b
   $\boldsymbol{a}(\text{Beam-a}) = \boldsymbol{a_o}(\text{Joint-b}), \boldsymbol{n}(\text{Beam-a}) = \boldsymbol{n}(\text{Joint-b}), P_i(\text{Beam-a}) = P(\text{Joint-b}).$
4. Joint-c と Beam-a
   $\boldsymbol{a_i}(\text{Joint-c}) = \boldsymbol{a}(\text{Beam-a}), \boldsymbol{n}(\text{Joint-c}) = \boldsymbol{n}(\text{Beam-a}), P(\text{Joint-c}) = P_o(\text{Beam-a}).$
5. Beam-b と Joint-c
   $\boldsymbol{a}(\text{Beam-b}) = \boldsymbol{a_o}(\text{Joint-c}), \boldsymbol{n}(\text{Beam-b}) = \boldsymbol{n}(\text{Joint-c}), P_i(\text{Beam-b}) = P(\text{Joint-c}).$

全体の変数順序は、接続条件において等しいとされた変数がなるべく同じブロックに属し、かつ、個々の機械要素についてあらかじめ設定された変数順序と矛盾しないように構成される。

Component Library を用いることにより、接続条件を定義するだけで全体の制約集合の生成と変数順序／項順序の設定を自動的に行えるようになり、制約条件の記述に要する設計者の負担を軽減できるものと考えられる。

5 おわりに

設計問題は制約充足問題として定式化可能である。特にメカトロニクス設計においては、制約条件の多くを代数制約として表現することが可能であるので、代数制約評価技術を応用した初期設計支援システムの開発を現在進めている。これを実際の設計作業に適用する場合、現実の設計作業において現れる問題を代数制約問題にブレイクダウンする具体的な手段が必要となる。本稿ではそのための一方法として、Component Library を用いた制約条件式ならびに変数順序／項順序の自動生成手法を提案した。

今後は、設計プロセスモデルと実際の設計作業過程との詳細な比較／検討を行い、より現実の作業に則した設計支援方法について研究を進める予定である。

参 考 文 献

[1] 畑村洋太郎: 実際の設計, 日刊工業新聞社, 東京, 1988.

[2] 齋藤友克, 竹島卓, 平野照比古: 日本で生まれた数式処理ソフト　リサアジールガイドブック, SEG 出版, 東京, 1998.

[3] Sawada, H.: Constraint-Based Design, Technical Report, CADC/98-11/R/04, CAD Centre, University of Strathclyde, U.K., 1998.

[4] 沢田浩之: 過小代数制約評価系とその機械設計への応用, 数理解析研究所講究録, No.1038, pp.85–95, 1998.