1999/5/15 2000/6/5

数式処理 *J.JSSAC (2001)*
Vol. 8, No. 2, pp. 19 - 35

論文

制約ベース型初期設計支援システム

沢田 浩之　　Xiu-Tian YAN

機械技術研究所*　　University of Strathclyde (UK) †

Abstract

In engineering design, a lot of design solutions are generated and more and more design parameters are defined when a design progresses. As more design parameters come into design consideration, designers are facing increasing difficulties in gaining an insight to the relationship among these parameters and it results in less optimal design solution. In order to overcome such difficulties, a design support system based on generic constraint solving technique is developed. The system has the following advantages: kinematic and energetic constraints about whole the product are automatically obtained only by connecting basic components selected from the component library, and an incomplete design solution can be appropriately evaluated.

1 はじめに — 初期設計過程 —

与えられた要求仕様をもとに新しく製品を設計する場合、その初期段階における作業は、通常、以下の手順にしたがって行われる。

1. 必要機能の設定
   与えられた要求仕様をもとに必要な機能を設定する。要求仕様とは、製品が達成すべき目的を意味する。また、機能とは、その目的を実現するための手段を意味する。例えば、要求仕様が「箱を移動させる」ことであるとき、機能としては「掴んで運ぶ」、「引きずって動かす」などが考えられる。

2. 物理的実体の決定
   必要機能を実現する具体的なモノを決定する。例えば、「掴んで運ぶ」という機能を実現する物理的実体として、ロボットハンドなどが考えられる。

3. 制約条件式の記述
   物理的実体が満足すべき制約条件を、数式として表現する。これらの条件として、幾何拘束や力の釣り合い条件などが挙げられる。

*hiroyuki.sawada@mel.go.jp
†x.t.yan@dmem.strath.ac.uk

4. 設計変数値の計算
   3. で記述された制約条件式を満足するような設計変数値を計算する。

これらの作業過程のうち、「3. 制約条件式の記述」および「4. 設計変数値の計算」には以下のような問題がある。

(a) 制約条件式の記述に関する問題
既存製品の手直しではなく新しく製品を設計する場合、その制約条件式は手書きのラフスケッチなどをもとに手作業で導かれることが多い。例えばロボットアームを設計する場合には、アームを線分で表現した簡単な図を描いて幾何拘束条件を導いたり、また、働く力を表現する矢印などを書き加えて力の釣り合い条件を導くなどといったことが行われる。このような方法では、作業そのものが設計者にとって大きな負担となるばかりでなく、誤解や勘違いに基づく人為的なミスを排除することがきわめて困難である。

(b) 設計変数値の計算に関する問題
設計変数値の計算は、通常、記述された制約条件式をもとにプログラムを作成し、これを実行することによって行われる。そのプログラムは、解析目的が異なるごとに異なったものが要求される。例えば、入力変数を与えて出力変数の値を求める場合（順問題）と、逆に、出力変数の値を与えて入力変数の値を求める場合（逆問題）とでは異なるプログラムを作成する必要がある。また、通常のプログラムでは入力変数のすべてに具体的な数値を与える必要があるが、設計の初期段階では、値が未定の設計変数が残されていることが多い。従来の方法では、値が未定の入力変数を含んだ式の扱いに困難がある。

さらに、異なった構造を持つ設計解について検討する場合には、上述の作業をその解ごとに行う必要がある。例えば、2 関節アームロボットと 3 関節アームロボットとでは幾何拘束も力の釣り合い式も異なるため、それぞれ別個に制約条件式を記述し、適正な設計変数値を計算しなくてはならない。これらの作業は設計者にとって大きな負担となるため、実際の作業では、様々な異なる解について充分な検討がなされないまま、最初に見つかった解で妥協してしまうことが少なくない。

本研究は、制約評価技術を応用することによってこれらの問題点を解決し、設計者の作業負担を軽減するソフトウェアシステムを構成することを目的とする。本論文の構成は以下の通りである。まず、次章において、本研究における問題解決のアプローチを示す。次に第 3 章でシステムの構成および実装について述べ、第 4 章では例題として 2 関節 2 本指ロボットの設計問題を取り上げる。

2 問題解決の手法

本章では、前章で提示した 2 つの問題、条件式の記述に関する問題および設計変数値の計算に関する問題について、その解決手法を提案する。

2.1 要素ライブラリ — 条件式記述に関する問題解決手法 —

機械設計では、それが新規設計であっても、すべての構成部品をまったくのゼロから設計することは事実上ありえない。むしろ、既存の部品を組み合わせることによって新しい構造を作り上げることが通常である。制約条件式の定義／生成という観点から見た場合、これは基本要素に関する局所的な制約条件集合と、要素間の接続条件を表す制約条件集合を組み合わせることによって、製品全体に関する制約条件集合を構成する操作として捉えられる。

ここで提案する手法は、機械設計において多く用いられる基本的な構成部品に関する制約条件式集合をあらかじめ記述しておき、それらを登録したデータベースを設計者に対して提供するというものである。このデータベースを要素ライブラリと呼ぶことにする。設計者は、要素ライブラリから必要な基本要素を取り出し、それら基本要素間の接続条件を追加することによって、製品全体に関する制約条件集合を得ることができる。要素ライブラリに登録さ

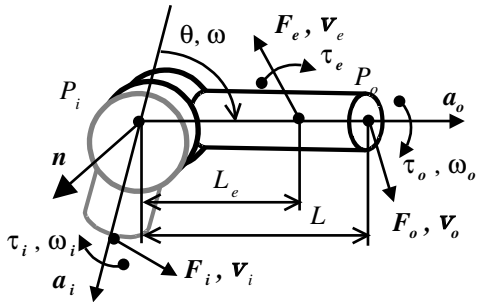

図 1: リンク要素

れている要素の一例として、図 1 のようなリンク要素を示す。その制約条件は以下のとおりである。

リンク要素に関する制約条件

幾何拘束 $|\boldsymbol{a_i}| = |\boldsymbol{a_o}| = |\boldsymbol{n}| = 1, \boldsymbol{a_i} \times \boldsymbol{a_o} = \boldsymbol{n} \sin\theta, \boldsymbol{a_i} \cdot \boldsymbol{a_o} = -\cos\theta, \overrightarrow{P_i P_o} = L\boldsymbol{a_o}.$

リンク軸回転速度ベクトルと相対回転速度の関係 $\boldsymbol{\omega_i} \cdot \boldsymbol{n} - \boldsymbol{\omega_o} \cdot \boldsymbol{n} = \omega$

リンク軸回転速度の伝達 $\boldsymbol{\omega_o} \times \boldsymbol{n} = \boldsymbol{\omega_i} \times \boldsymbol{n}$

リンク軸の剛体条件 $\boldsymbol{v_e} - L_e\boldsymbol{\omega_o} \times \boldsymbol{a_o} = \boldsymbol{v_o} - L\boldsymbol{\omega_o} \times \boldsymbol{a_o} = \boldsymbol{v_i}$

力の釣り合い $\boldsymbol{F_i} + \boldsymbol{F_o} = \boldsymbol{F_e}$

トルクの釣り合い $\boldsymbol{\tau_i} + \boldsymbol{\tau_o} - \boldsymbol{F_o} \times L\boldsymbol{a_o} = \sum(\boldsymbol{\tau_e} - \boldsymbol{F_e} \times L_e\boldsymbol{a_o})$

変数リスト $\boldsymbol{a_i}, \boldsymbol{a_o}$: リンク軸方向ベクトル、$\boldsymbol{n}$: リンク回転軸方向ベクトル、$P_i$: リンク回転軸位置、$P_o$: リンク先端位置、$L$: リンク長、$L_e$: 荷重点位置、$\theta$: リンク軸相対回転角、$\omega$: リンク軸相対回転速度、$\boldsymbol{\omega_i}, \boldsymbol{\omega_o}$: リンク軸回転速度ベクトル、$\boldsymbol{v_i}, \boldsymbol{v_o}$: リンク回

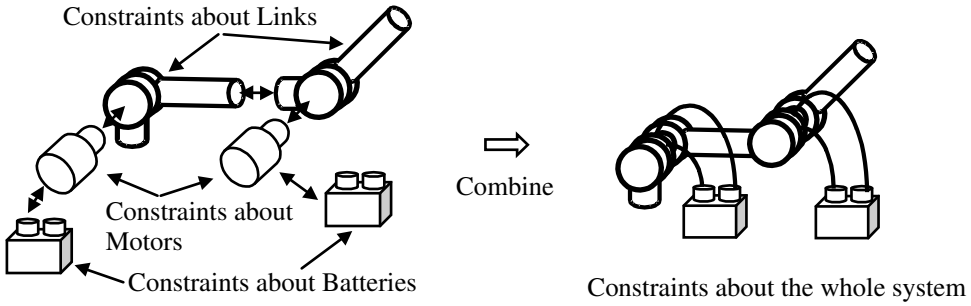

図 2: 制約条件集合の構成

転軸及びリンク先端における速度ベクトル、$\boldsymbol{v_e}$: 荷重点における速度ベクトル、
$\boldsymbol{\tau_i}, \boldsymbol{\tau_o}$: 生成されるトルク、$\boldsymbol{\tau_e}$: 外力トルク、$\boldsymbol{F_i}, \boldsymbol{F_o}$: 生成される力、$\boldsymbol{F_e}$: 外力。 ∎

また、要素ライブラリから抽出された基本要素間の接続条件を設定することによって、製品全体の制約条件集合が得られる例を図 2 に示す。図 2 の例では、要素ライブラリから、リンク、モータ、電源を選択して組み合わせることによって、ロボットアームが構成されている。設計者は、リンク同士を接続する点の位置座標の一致、リンク軸方向および回転軸方向の一致といった幾何的条件のほか、接続点における力の作用反作用といった力学的条件などを接続条件として定義すればよい。

2.2 代数制約評価系 — 設計変数値計算に関する問題解決手法 —

設計者は、設計変数の値を決定するために、順逆力学問題解析や最適化など様々な解析を行う必要がある。そのためには、それらの解析目的に応じて異なったプログラムを作成しなくてはならず、これが設計者にとって大きな負担となっている。こういった作業負担を軽減するために本研究で採用している手法は、様々な解析を自動的に行う代数制約評価系 [7] を設計者に対して提供するというものである。設計者は代数制約評価系に対して、製品全体を表現する制約条件集合を与えたのち、単に実行すべき解析を指示するだけで、その解析結果を得ることができる。すなわち、設計者はプログラム作成に要する手間を省くことができ、より多くの時間と労力を設計変数値の検討に使うことが可能となる。

初期設計支援を行うため、代数制約評価系には、過少制約問題を取り扱う機能が求められる。過少制約問題とは、解を特定するために必要な制約条件が不足している問題のことを指す。設計の初期段階では、決定すべき設計変数の数に比べて制約条件の数が少ないことが多く、通常、解の個数は有限ではない。例えば、「2 関節ロボットアームの先端は点 X および点 Y に到達できなくてはならず、かつ、各アームの長さは定められた範囲内になくてはならない」という制約条件は、代数不等式および代数方程式として表現されるが、アームの長さを具体的に定めるためには方程式の数が不足しており、解は無限個存在しうる。一方、点 X や点 Y の位置やアーム長さの範囲によっては、解がまったく存在しないこともあり得る。一般

にこのような過少制約問題は解決の見通しが悪く、設計者は自分自身の経験や勘に基づいた試行錯誤を余儀なくされる。その結果として、存在する解を見落とす可能性や、存在しない解を探し求めることに労力を費やす可能性は免れない [8]。本研究で開発した代数制約評価系は、これらの課題を解決するものであり、以下の機能を提供する。

1. 不等式間の矛盾検出
2. 矛盾解消のために修正すべき設計変数の特定
3. 最適化計算
4. 具体的な数値解の例示
5. グラフ描画

以下、各機能についてその概略を述べる。

2.2.1 不等式間の矛盾検出

製品全体を表現する制約条件集合を式 (1) とする。

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

ここで、$\boldsymbol{x} = (x_1, \ldots, x_n)$ は n 次元実数空間における変数、$f_i(\boldsymbol{x}), g_j(\boldsymbol{x}), h_k(\boldsymbol{x})$ は実数係数の多項式である。また、境界を含まない不等式制約 $e(\boldsymbol{x}) > 0$ ($e(\boldsymbol{x})$ は実数係数の多項式) が与えられた場合、それは $e(\boldsymbol{x}) \neq 0 \wedge e(\boldsymbol{x}) \geq 0$ として扱われるものとする。不等式間の矛盾検出機能とは、式 (1) の中で同時に成立させることのできない不等式の集合 $\{h_{i_1}(\boldsymbol{x}) \geq 0, \ldots, h_{i_m}(\boldsymbol{x}) \geq 0 \ (m \leq r)\}$ を導くものである。

スラック変数 $\boldsymbol{s} = (s_1, \ldots, s_q)$ および $\boldsymbol{t} = (t_1, \ldots, t_r)$ を導入することにより、式 (1) を次式 (2) に置き換える。

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

代数制約評価系は、式 (2) を用いてスラック変数 $t_1, \ldots, t_r$ のみで構成される等式 (3) を生成し、以下の規準により矛盾検出を行う。

$$c(t_{i_1}, \ldots, t_{i_m}) = 0 \ (m \leq r). \tag{3}$$

矛盾検出の規準

多項式 $c(t_{i_1}, \ldots, t_{i_m})$ の係数がすべて正で、かつ、定数項が正であるならば、不等式 $h_{i_1}(\boldsymbol{x}) \geq 0, \ldots, h_{i_m}(\boldsymbol{x}) \geq 0$ のすべてを同時に満足させることはできない。 ∎

例えば、「$t_1 + 2t_2t_3 = -1$」が導かれたとすると、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。

式 (3) の導出は、以下の 2 段階の手続きにしたがって行われる [7]。

1. グレブナ基底計算による制約条件式の正規化
2. 簡約木 (reduction-tree) の構成

図 3 にその例を示す。ここでは、直流モーターに関する特性式 (4) およびモーターに対する要求性能を表す式 (5) からなる制約条件集合を例として取り上げている。

$$\begin{aligned} \tau &= kI, \\ V &= RI + k\omega, \\ -V_0 &\leq V \leq V_0. \end{aligned} \tag{4}$$

$$\tau \geq \tau_0. \tag{5}$$

なお、各変数の持つ意味は以下のとおりである。

> τ: 出力トルク、ω: 回転速度、V: 入力電圧、I: 入力電流、k: トルク定数、
> R: 内部抵抗、V_0: 定格電圧、τ_0: 要求される出力トルク。

図 3 に示すように、まず、スラック変数 t_1, t_2, t_3 を含まない等式を用いてそのグレブナ基底を計算し、等式制約集合を正規化する。次に、そのグレブナ基底を用いて t_1, t_2, t_3 を含む等式を簡約化する[1)]。こうして得られた等式を初期ノードとして木構造を構成する。図 3 では、(1) から (6) の番号で示したものが初期ノードである。あとは、これらの等式を相互に簡約化することによって t_1, t_2, t_3 以外の変数を消去し、簡約化の結果得られた等式を元の等式の子ノードとして追加していく。例えば図 3 では、(2) の等式が (4) および (5) を適用することによって、それぞれ (7) および (8) の等式に簡約化されたことが示されている。このようにして構成される木構造を簡約木と呼ぶことにする。

簡約木の構成に際し、その項順序は、以下の条件を満足するように定められている [7]。

1. スラック変数 t_k は、スラック変数ではないどの変数よりも辞書式順序のもとで下位にある。
2. スラック変数 t_k を含む等式に現れる変数は、そこに現れない変数よりも下位にある。

2.2.2 矛盾解消のために修正すべき設計変数の特定

前項で述べた方法により矛盾が検出された場合、その矛盾解消方法を示唆することは、設計作業の支援上、意義が大きい。代数制約評価系は、矛盾を解消するために値を変更すべき設計変数を特定し、そのような設計変数のリストを設計者に対して提示する。

[1)] この例では、これらの等式をこれ以上簡約化することはできない。

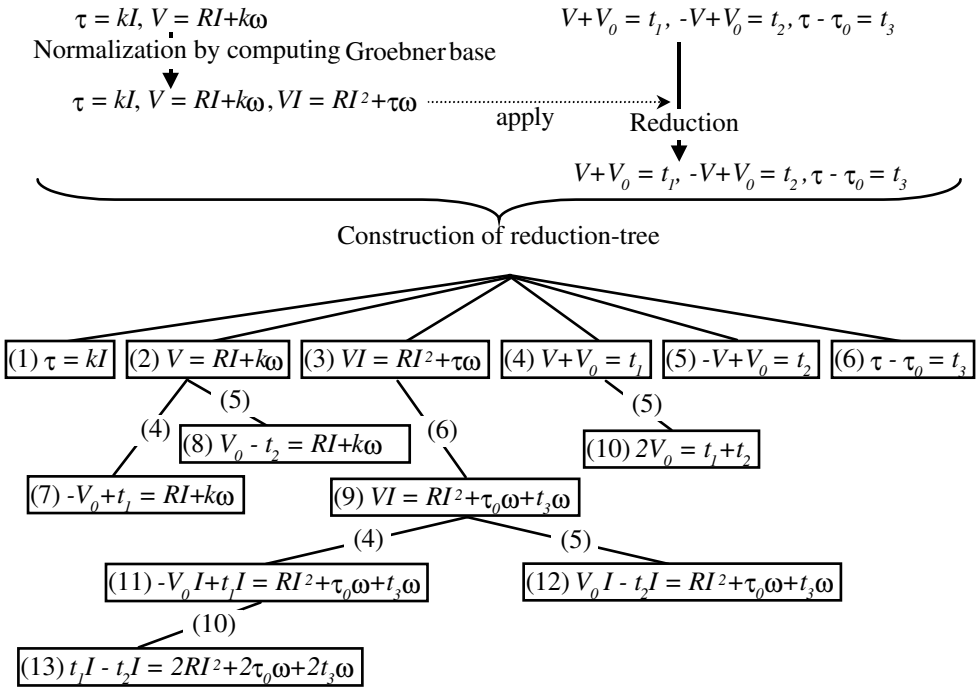

図 3: 簡約木の構成

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 (6) で表される。

$$x_k = a. \tag{6}$$

したがって、式 (3) を導くために式 (6) が必要であるならば、式 (6) 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 (6) が式 (3) を導くために不要であれば、式 (6) 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 (6) 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 (6) 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 (6) 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 (6) 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 (6) 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 (6) 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 [7]。

式 (3) で示される矛盾と式 (6) との関係

式 (2) から式 (6) 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 (6) 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 (6) 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。∎

2.2.3 最適化計算

設計過程では、「重量を最小化する」、「消費電力を最小化する」といった最適化計算を行うことが多い。代数制約評価系は、設計者が指定した目的関数について、式 (1) のもとにおける最小値あるいは最大値を計算する。

制約条件付最適化問題の解法についてはこれまでに多くの研究がなされており、ペナルティ法やバリア法などの数値解法が考案されている [5]。しかしながら、これら従来手法の多くは反復法に基づいた数値計算法であるため、解が収束しなかった場合、それが初期条件や収束条件が適切でなかったためなのか元の問題に解が存在しなかったためなのかが明確ではない。そのような問題を回避するため、代数制約評価系では、数式処理手法に基づいた制約条件付最適化計算法 [8] を採用している。その概要は以下の通りである。

設計者が指定した目的関数を $u(\boldsymbol{x})$ とする。式 (1) と式 (2) とは同値であるので、式 (1) のもとで $u(\boldsymbol{x})$ を最適化することは、式 (2) のもとで $u(\boldsymbol{x})$ を最適化することと同値である。$(n+q+r)$ 次元実数空間において、式 (2) で表される領域を A とする。最適化計算の手順は以下のようになる。

最適化計算の手順

1. 領域 A の部分領域への分割
関数 $u(\boldsymbol{x})$ の最適値を与える点は、$u(\boldsymbol{x})$ の導関数を導き、その値が 0 となる点を計算することによって求められる。その際、導関数の値が不連続となる部分が存在すると計算が正しく行われない。したがって、領域を導関数が連続な部分領域に分割し、各部分領域において最適化計算を行う必要がある。式 (2) の場合、その第 4 式によって与えられる領域 A の境界部において導関数の値が不連続となるので、領域 A を次式 (7) のように分割する。

$$\begin{aligned} A &= \bigcup A_{ij}, \\ A_{ij} &= A \cap \{(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t}) | t_{a_j(i,1)} = \ldots = t_{a_j(i,j)} = 0, t_{a_j(i,j+1)} > 0, \ldots, t_{a_j(i,r)} > 0\} \end{aligned} \quad (7)$$

ここで、$\{a_j(i,1), \ldots, a_j(i,j)\}$ は、1 から r までの整数のうちの j 個の整数からなる組合せであり、全部で ${}_rC_j$ 通りの組合せが存在する。i はこれらの組合せを区別するためにつけられる添字であり、1 から ${}_rC_j$ までの整数値をとる。

2. 各部分領域 A_{ij} における最適化計算
各部分領域における最適化計算を実行する。Lagrange 乗数法 [5] を用いて最適値を与える点に関する連立方程式を導いたのち、GSL 法 [1, 8] によって数値解を計算する。

3. 最適値の選択
   各部分領域で求められた関数 $u(\boldsymbol{x})$ の値の中から最適なものを選択する。 ∎

2.2.4 具体的な数値解の例示

設計変数の値は、そのすべてが合理的根拠に基づいて一意的に決定されるわけではない。むしろ、設計者の知識や経験をもとに適正と考えられる値を割り当てられるものが少なくない。与えられた制約条件をすべて満足する具体的な数値解を例示することは、設計者のそのような決断に関する手がかりを与えるという意味で、有効な支援手段となる。

代数制約評価系は、以下に述べる方法 [8] で具体的な数値解を計算する。まず、次の性質を持つ目的関数 $u(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t})$ を導入する。ただし、$u(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t})$ の定義域は $(n+q+r)$ 次元実数空間、$u(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t})$ の値は実数値である。

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

1. 関数 $u(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t})$ は、$(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t})$ 空間において連続である。
2. 関数 $u(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t})$ は、$(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t})$ 空間において最小値を持つ。
3. $x_1,\ldots,x_n,s_1,\ldots,s_q,t_1,\ldots,t_r$ のうちの少なくとも 1 つがプラス無限大もしくはマイナス無限大となるとき、関数 $u(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t})$ はプラス無限大に発散する。 ∎

式 (2) で表される領域 A の境界はすべて A に含まれているので、A が空集合でなければ、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) の解がその最小点の座標値として求められる。関数 $u(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t})$ が領域 A において最小値を持たなければ、それは A が空集合であることを意味し、式 (2) に実数解は存在しない。最小値の計算は、第 2.2.3 項で述べた方法による。

2.2.5 グラフ描画

設計変数の値を決定する場合や設計解の検証を行う場合、特定の変数間の依存関係を表現したグラフを用いることが多い。代数制約評価系は、設計者が指定した 2 変数 x_i,x_j $(i<j)$ に関する関係式を導き、そのグラフを描画する。その際、x_i,x_j に関するグラフのプロットを行うだけでなく、解の存在が許されない禁止領域の表示も行う。なおグラフ描画には、文献 [3] の方法を用いている。

プロット関数の導出方法

1. $x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_{j-1},x_{j+1},\ldots,x_n,s_1,\ldots,s_q,t_1,\ldots,t_r \succ x_i,x_j$ に関する辞書式順序 [6] のもとで式 (2) のグレブナ基底 G_p を計算する。
2. G_p の中から x_i,x_j のみを含む方程式を選び出す。これがプロットされるグラフを表す関係式となる。x_i,x_j のみを含む方程式が G_p に含まれない場合、それは式 (2) の中に x_i,x_j のみで記述される関係が存在しないことを意味する。 ∎

禁止領域の導出方法

1. $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n, s_1, \ldots, s_q \succ t_1, \ldots, t_r, x_i, x_j$ に関する辞書式順序のもとで式 (2) のグレブナ基底 G_r を計算する。

2. G_r の中から $t_1, \ldots, t_r, x_i, x_j$ 以外の変数を含まない方程式をすべて選び出し、これらの方程式からなる集合を G_s とする。

3. 解が存在する領域は、次式 (8) を満足する点 (x_i, x_j) の集合として定義される。

$$G_s \cup \{t_1 \geq 0, \ldots, t_r \geq 0\}. \tag{8}$$

   したがって、Quantifier Elimination [2] を用いて、次式 (9) から変数 $t_1, \ldots, t_r$ を消去すれば、解の存在する領域を与える連立方程式ならびに不等式が得られる。

$$\exists(t_1, \ldots, t_r)[G_s \cup \{t_1 \geq 0, \ldots, t_r \geq 0\}]. \tag{9}$$

4. 解の存在が許されない禁止領域は、解が存在する領域の否定として求められる。 ∎

ここで、プロット関数が存在しない場合と存在する場合とでの、禁止領域の表示の仕方の違いについて述べておく。プロット関数が存在しない場合、上述の手順で得られた禁止領域は赤色のハッチングによって示される。一方、プロット関数が存在する場合には、上述の手順で得られた禁止領域はグラフ曲線上にないすべての点を含むことになるので、これをハッチングで示すことは不適切である。したがって、プロット関数が存在する場合には、グラフ曲線上の点のうち禁止領域に含まれるものを赤色で表示するにとどめ、特に禁止領域のハッチングは行わない。

3 システム構成

図 4 にシステムの構成を、また、図 5 にシステムの概観を示す。システム本体は Visual C++、代数制約評価系は Risa/Asir [4] Windows 版で実装されている。

設計者は、ユーザインタフェースを介して操作を行う。まず、製品構成ツリーにおいて、製品全体を構成要素に分割する。大規模な製品を設計する際には、製品全体をいくつかの構成要素に分割したのちそれらを独立に設計し、最後にそれら構成要素を組み合わせて製品全体を構成するといった方法が多く用いられる。製品構成ツリーは、そのような作業環境を提供するものである (図 5 左上)。

次の作業は製品モデルの構成であり、これは制約エディタ上で行われる。すでに第 2.1 節で述べたように、設計者は必要な部品を要素ライブラリから抽出し、それら部品間の接続条件を定義することによって製品モデルを構成する。今回インプリメントされたシステムでは、要素ライブラリからの部品抽出は、部品アイコンのドラッグ&ドロップによって行われる。また、抽出された部品間に線を引くことによってそれらの間に接続条件が存在することが示される (図 5 右下)。

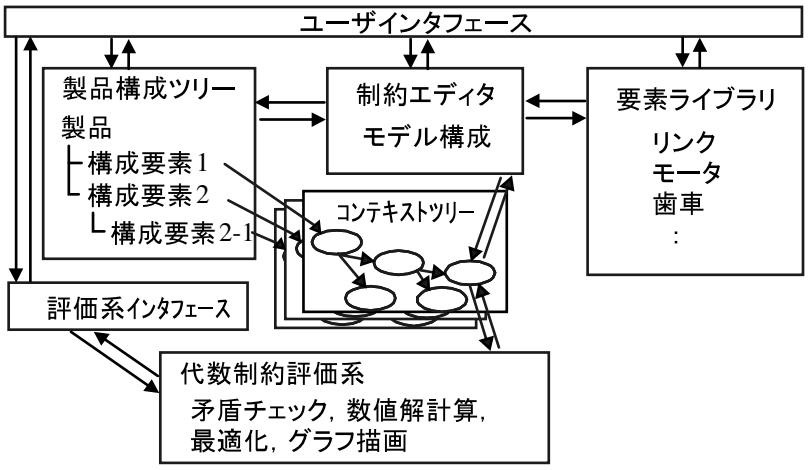

図 4: システム構成

構成された製品モデルはすべてコンテキストツリーに登録される。設計過程では、構造の変更や寸法の修正などにともない、何種類もの製品モデルが構成される。これらの製品モデルをその制約条件集合の包含関係に基づいて、木構造として分類／整理するものがコンテキストツリーである。コンテキストツリーは、製品構成ツリーで分割された各構成要素ごとに 1 つずつ生成される (図 5 中央やや左)。ある程度具体化された製品モデルは代数制約評価系で評価される (図 5 奥)。設計者はその評価結果を参照することにより、さらに製品モデルの改善を進める。

4 例題 — 2 関節アームロボットの設計 —

例題として、図 6 のようなワイヤ駆動式 2 関節アームロボットの設計を考える。ここで行う作業は、以下のオペレーションおよび前提条件が与えられているときに、各関節について、**適切なモーターを選択し、ギア比を決定する**というものである。

オペレーション 質量 2 [kg] のものを、速度 30 [cm/s] で持ち上げる。動作範囲は、$x = 25$ [cm]、0 [cm] $\leq y \leq 25$ [cm] である。

前提条件 アームの構造および姿勢は以下のように与えられるものとする。

- アームの長さおよび質量は、それぞれ 20 [cm]、1 [kg] である。
- Joint A は常に $y \geq 0$ の位置にあり、かつ、Joint A における関節回転角 θ は 0 以上で、逆方向には曲がらない。また、Joint B の位置は原点である。

設計作業は以下の手順で行われる。

1. 製品モデル構成

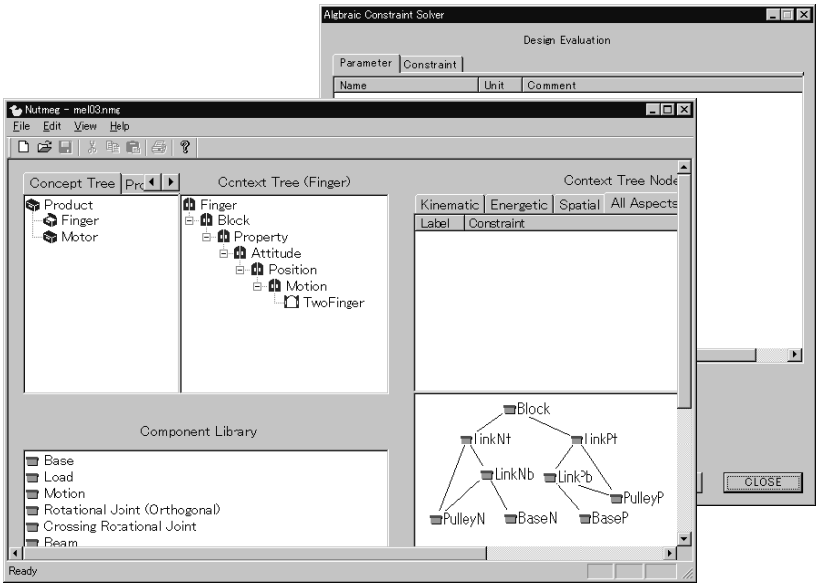

図 5: システム概観

2. 各関節最大出力の計算
3. **モーター選定**
4. **適正なギア比の決定**
5. その他必要な解析

各作業の詳細について述べる。

1. 製品モデル構成
   要素ライブラリから必要な要素を選択し、制約エディタ上で製品モデルを作成する (図 7)。製品モデルは、構成要素間の依存関係を示すグラフとして表現される。必要に応じて、寸法や要素間の接続条件を定義する。図 7 は、接続条件定義用のウィンドウが表示され、デフォルトの接続条件が自動的に生成されたところを示している。システムの内部では、自動的に製品全体に関する制約条件集合が構成される。この例題の場合、設計変数 132 個、制約条件 146 個からなる制約条件集合が構成される。

2. 関節最大出力の計算
   制約評価系に各関節の最大出力を計算させる。この計算は以下の手順で実行される。まず、制約評価系の「Maximum/Minimum」チェックボックスをチェックする。そして、「Start」ボタンを押したのちに表示されるウィンドウで、目的関数を定義する (図 8)。この場合、目的関数として関節出力トルクと回転速度の積を定義し、以下の結果が得られる。

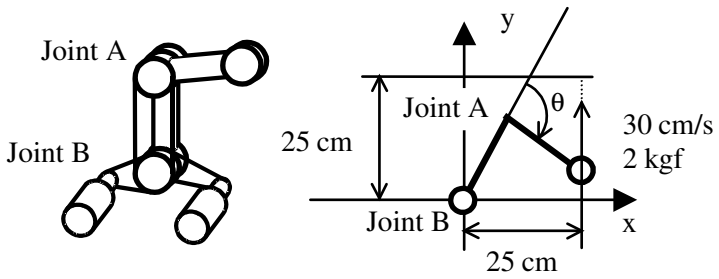

図 6: 2 関節アームロボット

Joint A の最大出力	8.12 [W]
Joint B の最大出力	5.15 [W]

3. **モーター選定**
   求められた関節最大出力に基づいて、市販品のカタログから必要な出力を持つモーターを選定する。ここでは、以下のモーターを選んだとする。

最大出力 [W]	21.2
定格電圧 [V]	12
トルク定数 [mNm/A]	20.9
内部抵抗 [Ω]	1.7

   これらの値をモーターの属性値として設定する。

4. **適正なギア比の決定**
   一般にギア比を決定する場合には、関節の最大トルクや最大角速度を計算し、それらを実現できるようなギア比を計算するという方法が多く用いられる。が、ここでは、アーム先端位置とギア比のグラフを制約評価系に描画させ、それによってオペレーションを実行可能とするギア比の目処をつけるという方法をとる。アーム先端位置と Joint A のギア比のグラフを図 9 に示す。グラフの縦軸はアーム先端位置の y 座標、横軸は Joint A のギア比である。ハッチングされた部分は、解として認められない禁止領域を示す。この図から、オペレーションを実行可能な Joint A のギア比を求めることができる。例えば、ギア比として 0.04 を選んでオペレーションを実行した場合、ある高さまで達すると禁止領域にぶつかるが、これは、その時点で必要な力あるいは速度を出すことができなくなることを意味する。禁止領域内の点で右クリックを行うと、その点の座標および抵触する不等式制約条件が示される。図 9 左下のウィンドウで丸印のついた条件式が、この点における抵触条件である。

   あとは、必要とあれば、何らかの最適化計算を行うことによってギア比を決定する。ここでは、関節出力最大時のモーター消費電力が最小となるようなギア比を制約評価系に計算させ、以下のようにギア比を決定する。

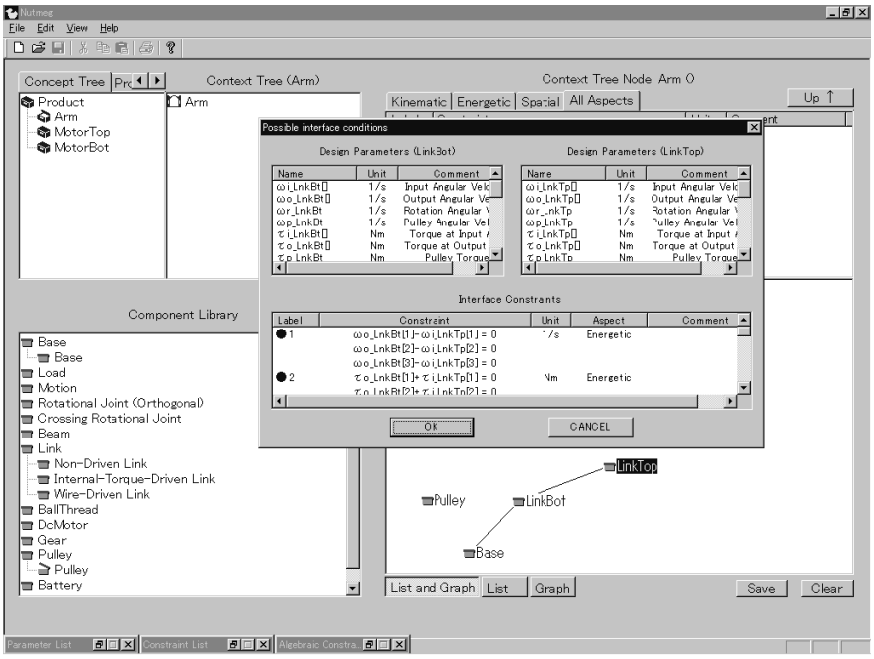

図 7: 製品モデルの構成

Joint A のギア比　1/245
Joint B のギア比　1/415

この計算は以下の手順で行われる。「2. 関節最大出力の計算」で行われた計算により、各関節について、その出力が最大となるアーム先端位置の y 座標がすでに求められている。したがって、その値を制約条件として追加した後、「2. 関節最大出力の計算」で行ったものと同じ手順にしたがい、目的関数としてモーター入力電圧と入力電流の積を定義し、その最小値を計算させればよい。

5. その他必要な解析

すでにアームロボットの構造、モーター、ギア比は、与えられたオペレーションを実行できるように決定された。通常の設計では、さらに、このアームロボットの性能をより正確に把握するための解析が行われる。これらの解析は、いわば、アームロボットの余力を測るためのものであると言ってもよい。ここでは、アーム先端が x 軸上で静止している時に出すことのできる力を、グラフとして描画させることにする。そのグラフを図 10 に示す。このグラフは、言葉を換えていうと、アーム先端がどれだけの外力に抗して静止していられるかを示すものである。図の縦軸は y 方向の力、横軸は x 方向の力である。アーム先端が出すことのできる力を示す白色の領域は、その中心が原点よりもやや下側にずれている。これは、自重の分、上側には大きな力を出せないためだと考えられる。

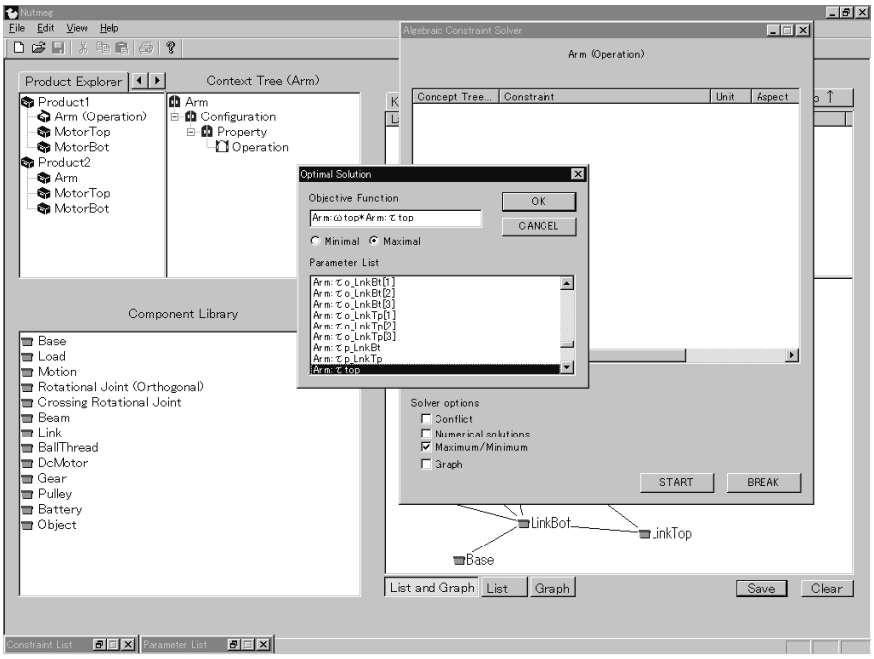

図 8: 関節最大出力の計算

5 おわりに

初期設計における作業負担を軽減し、その作業効率を向上させることを目的とした初期設計支援システムを開発した。本システムを使うことにより、設計者は、様々な設計解を広い範囲の観点から効率的に検討することが可能となる。本システムは、汎用的な制約評価技術、特に数式処理技術を応用した手法を用いており、以下の特長を持つ。

1. 単純な基本要素を組み合わせることにより、広い範囲の構造の構成／解析が可能である。
2. 解析プログラムの作成が不要である。
3. 設計者の要求に応じた様々な解析が可能である。

これらの特長は、製品を表現する制約条件式を直接操作することのできる数式処理技術を活用することによって実現できたものである。特に、入力変数と出力変数を区別して考える必要がないことは、大きな利点と考えられる。例えば、本論文の例題では、ロボットの構造が与えられた上でモータ選択と減速比決定を行う問題を扱ったが、逆にモータが与えられた上で構造を決定する問題もまったく同様に扱うことができる。

本システムは、現在、機械分野における設計を対象として開発されている。しかしながら、制約条件を代数方程式あるいは代数不等式で表現できるものであれば、扱うことのできる分野に制限はない。したがって、より一般的な設計支援システムへと拡張していくことも可能である。

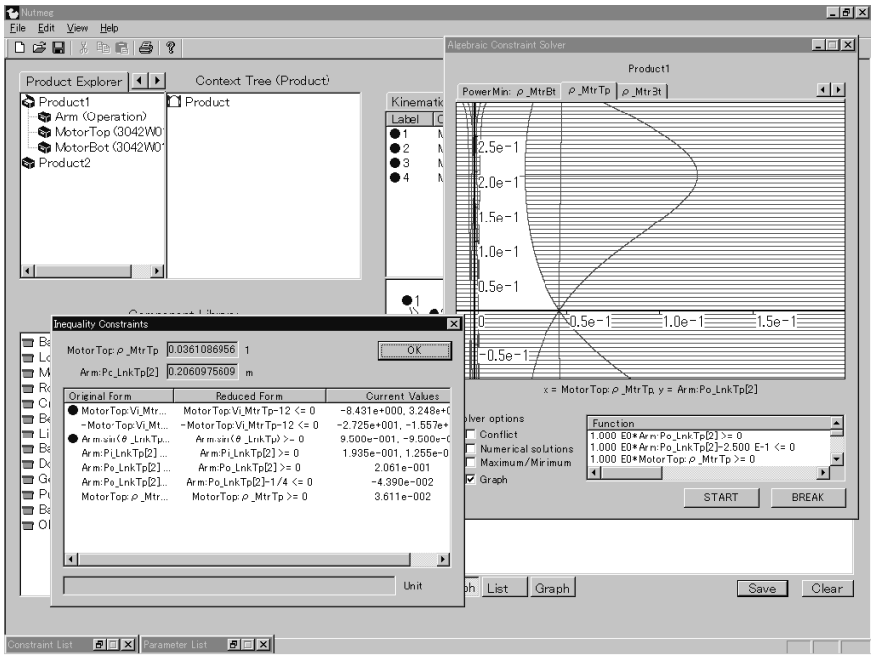

図 9: アーム先端位置–ギア比のグラフ

謝 辞

本研究にあたり、適切な助言を頂いた機械技術研究所極限技術部長の小鍛治繁博士に深く感謝します。また、Quantifier Elimination についてお教え頂いた富士通研究所の穴井宏和氏ならびに Risa/Asir でのインプリメントに関し助言を頂いた富士通研究所の野呂正行氏にこの場を借りて感謝申し上げます。

参 考 文 献

[1] Alonso, M.-E., Becker, E., Roy, M.-F. and Wörmann, T.W.: Zeros, Multiplicities, and Idempotents for Zero-dimensional Systems, *Progress in Mathematics*, **143**, 1996, 1–15.

[2] Weispfenning, V.: Quantifier Elimination for Real Algebra – the Quadratic Case and Beyond, *AAECC*, **8**, 1997, 85–101.

[3] 齋藤友克, 近藤祐史, 三好善彦, 竹島卓: Displaying Real Solution of Mathematical Equations, 数式処理, **6**(4), 1998, 2–21.

[4] 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.

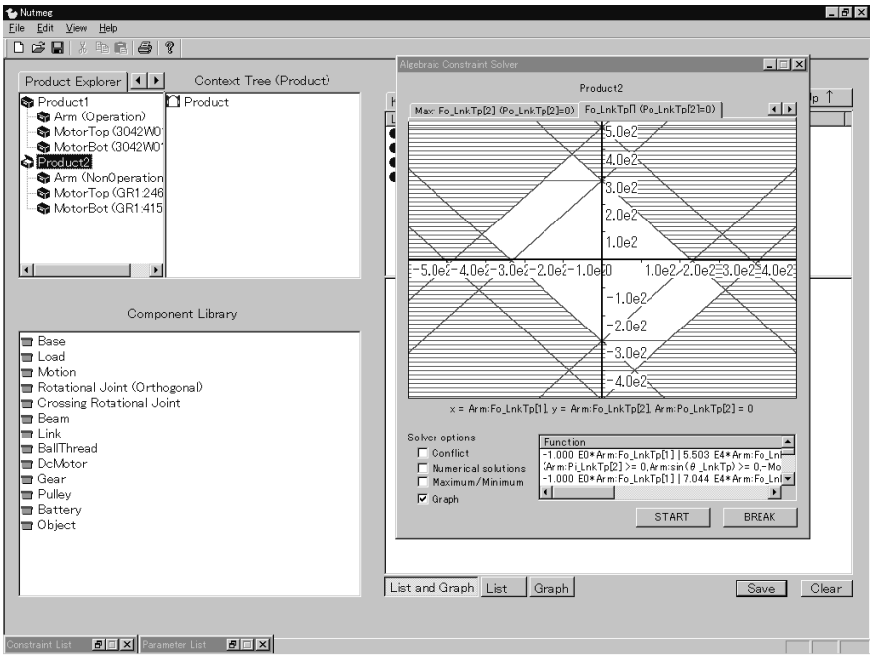

図 10: アーム先端が出すことのできる力

[5] 坂和正敏: 非線形システムの最適化＜一目的から多目的へ＞, 森北出版, 東京, 1986.

[6] 佐々木建昭, 今井浩, 浅野孝夫, 杉原厚吉: 計算代数と計算幾何, 岩波講座 応用数学, 岩波書店, 東京, 1993.

[7] Sawada, H.: The Algebraic Constraint Solver as a Design Tool, *Electrical Proc. IMACS ACA'98* (Liska, R., eds.), Prague, 1998, http:// math.unm.edu/ ACA/ 1998/ sessions/ industr/ sawada/ index.html.

[8] 沢田浩之: 数式処理的手法に基づく過少代数制約問題の数値解法, 情報処理学会論文誌, **40**(5), 1999, 2314–2324.