花原和之, “最適設計の基本的な考え方”, 日本機械学会関西支部第 262 回講習会
「製品開発, 技術開発のための最適設計技術と応用」, 1-14, 2003 より.

最適設計の基本的な考え方

神戸大学工学部　花原和之

1　はじめに

論理の探求が数学者の本能であり, 森羅万象の究明が理学者の使命であるのと同様に, 最適化は技術者の本能であり使命である, と言うことができる. このことを, 例えば瀬口 [(1)] は以下のように表現している.

> …技術者はできるかぎりよい設計をしようとする. よい設計を求めることは設計者の義務でもあるし, 設計者とはそうすることにより無上の喜びを得ることができるようなたぐいの人々の呼び名である.

言い換えれば, 全ての設計が (設計者にとって, 何らかの意味での) 最適設計であるべきなのは必然である. しかしながら, 最適設計というものを設計者個人の自己満足以上のものとしてとらえてゆくためには, 何をもって最適とするのか, どのように最適にするのか, といった明確な観点が必要となる.

最適設計手法とは, 「よい設計」を可能なかぎり合理的に追求してゆくための方法論である[1]. 本稿では, 最適設計 (optimal design) を行うための基本的な考え方について述べる. 設計問題をどのように取扱うか, といった点から始めて評価規範の決定, 最適設計問題の定式化, 最適化処理の実施までを概観する.

最適設計の数理的取扱いや最適化手法に関しては, 丁寧に書かれている書籍 [(2)] が既に存在するため, 本稿は最適設計を行うための「心構え」とでも言うべきものにやや重きを置いた内容となっている. ただし, 最適設計の数理的側面以外を一般論として議論することには困難な点も多く, 筆者個人の主観が少なからず反映されているものであることを予めお断りしておく.

2　設計過程と最適設計

設計は, 人間が必要とする機能を一つの製品あるいはシステムとして具体化してゆく過程であると定義することができる [(3)]. このような設計活動の流れは, 抽象から具象, 定性から定量へと向かういくつかの特徴的な段階からなる設計過程 (design process) としてモデル化される. ここでは, 一般的な設計過程について概説し, 各々の段階で最適設計をどのようにとらえてゆくべきか, という点について概観する.

2.1　設計過程

一般的な設計過程は図 1 のように表すことができる[2]. ここで, 図 1 の設計過程における各段階の概略は以下のようなものである.

製品企画 (product planning)　「要求把握 (needs analysis)」と呼ばれる場合もある. 製品に求められる要求や製品の果たす役割を明確化する. 設計要求 (design requirements) を把握し, 設計仕様 (design specifications) を策定する.

概念設計 (conceptual design)　設計仕様に合う設計案 (design concept) を作成する. 設計解の様式を定める.

基本設計 (preliminary design)　「実体設計 (embodiment design)」と呼ばれる場合もある. 評価を可能とする設計モデル (design model) を構築する. また, 得られたモデルに含まれるパラメータの中から設計変数 (design variable) を選択する.

詳細設計 (detail design)　詳細な寸法や形状を決定する. 設計解 (design solution) を決定する.

工学設計において技術者が直面する課題では, 他の様々な事情により設計仕様が既に与えられている場合が少なくない. この場合, 設計過程の最も上流の段階は概念設計となる. 本稿では, 製品企画が設計過程の最初の段階となる場合を「広義の設計問題」, 概念設計が最初の段階となる場合を「狭義の設計問題」と呼ぶが, 特に混乱が生じない場合には何れも単に設計問題と呼ぶ.

典型的な最適設計問題の例題である「コート掛け問題」の場合[3], 設計過程の各段階は, 例えば以下のようになる (図 2).

[1] 後述するように, 設計問題の特質から, 設計活動の全ての側面において合理的なアプローチが適用可能なわけではない.

[2] 実際には設計過程のモデルとして様々なものが考えられている [(3)−(6)]. 例えば, 「製品企画」の前に「初期研究 (feasibility study)」の段階や, 「詳細設計」の後に「生産設計 (production design)」の段階を想定しているものもある. しかしながら, 本稿の主題とは離れるため, ここでは設計過程の詳細には立ち入らない.

[3] ただし, 一般に「コート掛け問題」という場合には, この例題における概念設計以降の狭義の設計問題である「片持はり設計問題 (cantilever beam design problem)」を指すことが多い.

製品企画

⇓ (⇑)

概念設計

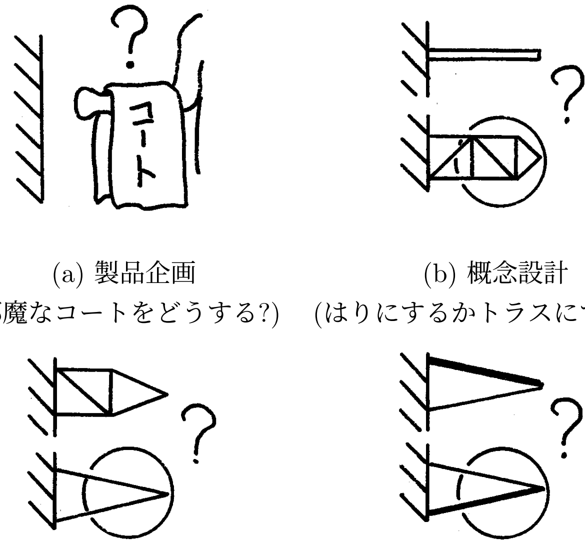
⇓ (⇑)

基本設計

⇓ (⇑)

詳細設計

図 1: 設計過程



(a) 製品企画
(邪魔なコートをどうする?)

(b) 概念設計
(はりにするかトラスにするか?)

(c) 基本設計
(どんなトポロジにするか?)

(d) 詳細設計
(部材の太さは?)

図 2: コート掛け問題

(a) 製品企画 コートを手に持たずに済ますための「何か」が必要であるという要求から,「壁面に固定するコート掛け」という設計仕様を策定.

(b) 概念設計 「壁面に固定するコート掛け」の構造様式を決定する. ここではトラス構造の採用を決定.

(c) 基本設計 設計解であるトラスのトポロジあるいは部材レイアウトを決定する. これにより, 構造解析が可能となる. ここでは最も単純な 2 部材トラスの採用を決定.

(d) 詳細設計 部材断面積を決定する.

ただし, これらの各段階の分類は厳密なものではなく, 実際の問題や設計の状況によって異なるとらえかたが可能である.

2.2 最適設計の適用と設計過程

設計過程の各段階における最適設計の適用について見てみよう.

製品企画

この段階の目的は「"最適設計問題の"(最適) 設計」である. どのような観点に立って何を達成しようとするのか, という最適設計過程全体を支配する設計目的 (design objective) を決定する. 「最適な仕様とは何か」「そもそも何をもって最適であると判断するか」といった, 人間の認知にも大きく左右される部分が多く, 明確な問題の定義すら困難であることもしばしばである. このような問題は悪定義問題 (ill-defined problem) と呼ばれるクラスに属する (7). この種の問題に対する, 合理的と言えるアプローチは存在しない. 状況に対する**綿密な情報収集**と, **幅広い視野と経験**にもとづいて決定を行うことが肝要であろう. また, **先入観の排除**も非常に重要である.

例えば, 先に述べた「コート掛け問題」において, 「コート掛けは '壁面に支点を持つ' 構造物」という先入観を排除すれば, 「'天井に支点を設定した' コート掛け」という設計仕様の代替案を構築することも可能かも知れない[4]. 一般に, 張力のみで荷重を支える構造のほうが片持ちはり構造よりも強度やコストの点では有利である. しかしながら, この代替案を採用するためには, 先の「壁面に固定するコート掛け」という設計仕様では考慮されていなかった, 天井の高さやその構造強度といった情報を保有し, 吟味する必要がある. また, このような設計仕様を採用する場合には, それまではあまり考慮されていなかった (強度やコスト以外の) 観点にまで視野を広げることが特に求められる. そうすれば, 先の仕様には含まれていなかった, 天井付近の景観等 (照明に及ぼす影響等) についての配慮が必要であることに気がつくであろう.

[4] それどころか「コート掛けが必要である」という先入観を排除した「どこかにクロークを設ける」という代替案や, 「コートを身体のどこかに掛けておくための器具を開発する」という代替案, もっと遡って先入観を排除して「コートを持ってこさせないようにする」という代替案すら可能であるが, ここでは深入りしない.

このように, 先入観を排除し, 情報を収集し, 広い視野の下で状況を検討することにより, 改善された設計仕様を構築することができる場合がある.

概念設計

この段階の目的は「与えられた仕様を最適に満たす (ことが期待される) 設計案の作成, すなわち設計目的に適う設計解の様式の決定」である. この段階が設計過程の起点となっている場合も少なくない. 概念設計問題は, 典型的な不良設定問題 (ill-posed problem) として知られている. 仕様に対応する設計案の候補は, その時点で既知でないものも含めて 無数に存在する. しかしながらそれらの候補の中から, 実行可能 (feasible) となり, さらには最適となりうるものを先験的に見出すための一般的な方法は存在しない.

「コート掛け問題」においては, トラス構造の他に片持ちはり構造, ラーメン構造, それら各種の組合せといった様々な構造様式が候補として挙げられる[5]. 設計者はこれらの構造様式の種々の **客観的な特徴**にもとづいて, 設計案の**主観的な決定**を行うことになる. 壁面から荷重支持点までの距離が比較的短い場合や要求荷重が小さい場合であれば, 単純な片持ちはり構造で十分であろう. しかしながら, 荷重支持点までの距離が長くなり, 要求荷重が大きくなれば, ラーメン構造やトラス構造といった高剛性を特徴とする構造を選択すべきである.

概念設計段階は不良設定問題でもあり, 一般に, 最適設計の観点からの取扱いは容易ではない. しかしながら, 得られた設計案が最終的な設計解の最適性に及ぼす 影響は多大 であり, 非常に重要な段階である. 例えば「コート掛け問題」において, 壁面からの距離がある程度大きいにもかかわらずこの段階で片持ちはり構造を選択した場合, 基本設計や詳細設計の段階でどんなに緻密な最適化を行った場合でも, 軽量性や剛性といった観点からは単純なトラス構造の最適性に劣るであろう.

概念設計段階の意志決定は, 設計者自身の知識や経験に依るところが非常に大きい. 逆に言えば, 設計者の創意と工夫が最も威力を発揮する段階である. 近年では, 知識工学的手法にもとづくアプローチも研究されており, 各種の支援システムも開発されているが, 最終的な決定を行うのは設計者自身 であることを肝に命じておくべきである.

基本設計

この段階の目的は「設計案にもとづく具体的な設計モデルを最適に構築すること」であり, 数理的最適設計を行う場合には設計モデルの定式化が行われる. 一般に, この段階はいわゆる悪構造問題 (ill-structured problem) であり, モデル構築作業のマニュアル化やアルゴリズム化が容易でないことが多い. しかしながら, 設計者自身の経験や過去の事例にもとづいて最適設計のための合理的な設計モデルを構築することが可能である.

先に挙げた「コート掛け問題」の場合, この段階において行わなければならないことはトラスのトポロジの決定[6] である. いったんトポロジを定めてしまえば, 設計解の生成と評価に必要な全ての構造特性の数理的な定式化を直ちに行うことができる. この場合, モデルに含まれるパラメータとして節点位置, 部材断面積, 材料定数といったものが挙げられるが, これらの中から設計変数となるべきものを選択することになる.

数理的な最適化手法の適用を考える場合, 基本設計は次に述べる詳細設計と密接に結び付いている. 利用可能な解析や最適化のための**ツール群や計算機の能力を視野に入れたモデル**の構築を行うことが肝要である. また, 設計モデルに含まれる, どのパラメータを設計変数とするかということも重要な問題である. 多くのパラメータを設計変数として採用すれば, 実行可能な設計解の範囲, すなわち設計可能領域 (feasible design region) が広くなるため, 解の最適性という観点からは有利である. しかしながら, 設計変数の数の安易な増加は, 詳細設計を行う際の計算コストの急激な増大にもつながりやすく, 数理的な最適化処理が実質的に不可能となる場合もあるので注意が必要である.

詳細設計

この段階の目的は「全ての設計変数を最適に決定すること」である. 設計モデルと設計変数に大きく依存するが, 効果的な最適設計のためには 目的に応じた適切な 最適化手法を選択することが肝要である.

近年では, 多くの最適化ツールが開発され, 実用化されているが, **最適化手法は万能ではない**. 一般に, 最適化の精度と必要とされる記憶容量や計算時間はトレードオフの関係にある. また, 最適化手法には何らかの初期値からの繰り返し計算により最適解を求めるものが多く存在するが, このような場合, 初期値を適切に設定することにより計算コストの軽減がなされることが少なくない. 何れにしても, 最適化手法や最適化ツールの特性 をよく把握した上で利用することが必要である. 代表的な最適化手法については 5 節で概説する.

以上, 設計過程の各段階と最適設計について概観したが, 設計活動は上流から下流への一方向の流れではない. 各段階で得られた結果が不十分なものであった場合には, 過程をさかのぼってやり直すことが必要となる.

---

[5] もちろん, ここに挙げた例のほかにも, 未知であるものも含めた数多くの構造様式が存在するであろう.

[6] 実際には, トラスのトポロジをもパラメータとして含めたモデルを構築し, 数理的な定式化を行うことも可能である. しかしながら, 全ての可能なトポロジを設計解モデルの中に含めることはできないため, 何らかの形で想定するトポロジの範囲を定めることになる (4.1 節「その他の表現」参照).

3　設計評価規範

本稿ではここまでやや漠然と最適という言葉を用いてきた．では「最適」とは何を意味するだろうか．国語辞典 [8] によれば,「最適」とは「いちばん適していること」とあるが, 工学的な「最適」を議論するためには,「何をもって適しているとするか」という点を明確にしておく必要がある．設計の最適性を評価するための判断基準を設計評価規範 (design criterion) あるいは単に評価規範と呼ぶ.

何らかの形で評価規範を決定しなければ, 得られた設計解を評価することもできず, したがって最適設計を実施することは不可能である．設計評価規範の決定は, 最適設計における最も重要な作業であるといっても過言ではない.

ここでは, 最適設計のための評価規範について概説する。

3.1　主観的・定性的評価規範

最適設計を行うためには, 2 節で述べた設計過程における製品企画の段階で, 設計仕様と同時に設計評価のための判断基準を選択する必要がある．この段階では, これは例えば表 1 に示すようなものとなるであろう．これらの判断基準は, 一見してわかるように, 最適設計のための評価規範として用いるにはやや主観的・定性的なものであり, その数理的取扱いは困難である．しかしながら, これらの評価規範こそが, 設計目的に直結する根本的なものであることに注意する必要がある．また, 特に主観的な判断に基礎を置く概念設計の段階までは, これらの規範をこのままの形式で用いることもできる.

3.2　客観的・定量的評価規範

基本設計・詳細設計の段階で数理的最適化手法を適用するためには, 前節で述べた主観的・定性的な評価規範は, そのままの形では用いることができない．数理的な最適化のためには, 何らかの手段を用いて評価規範を客観的・定量的な取扱いが可能な様式に改める必要がある．表 2 に客観的・定量的な設計評価規範の一例を示す．これらの評価規範は比較的取扱いが容易であることから, 現在, 最適設計を考える際に広く用いられているものである．また, 元来は主観的・定性的な評価規範であったものを, 直接, 客観的・定量的な規範として取扱うための手法も研究されている [9],[10].

客観的・定量的評価規範は便利ではあるが, それ故に本来の最適化の意図を見失わないように注意しなければならない. 例えば, 移動性 (表 1) を評価規範とする最適設計過程における基本設計・詳細設計の段階で, 数理的最適化のための規範として製品の質量 (表 2) を採用する, といったことが考えられる．しかし, たしかに質量は移動性を評価するための一つの側面ではあるものの, これら二つの評価規範は決して等価なものではない．場合によっては, 質量を評価規範とする最適化の結果得られた製品形状や部品配置が, 結果的に移動性を損ねてしまうこともありうる.

数理的最適化手法は有用であるが非情である．定式化された側面については (計算機の処理能力の及ぶ範囲で) 極限までの最適化を達成することができる反面, 定式化されなかった側面については全く考慮されることがない．設計者は, このような最適化の結果を鵜呑みにすることなく, 常に批評的な目で見ることが重要である．定量的な評価規範にもとづく数理的最適化は, あくまでも手段であって目的ではない．本来の設計目的と, その根本となっている評価規範のことを常に念頭に置いておくべきである.

3.3　複数評価規範の取扱い

設計の際に考慮しなければならない判断基準は多くの場合単一ではない．例えば, 経済性と機能性 (表 1) の一方を無視した設計は有り得ないであろうし, 質量と強度 (表 2) は構造設計において常に考慮されるべき項目であろう．また, 一般

表 1: 「よい設計」の判断基準の例 [1]

判断基準	説　明
機能性	要求される機能を満足していること.
経済性	価格が低いこと.
生産性	作りやすさ, 能率のよさなど.
移動性	ハンドリングが容易であること.
運用性	使いやすく, すぐにこわれないこと.
廃棄性	使命を終えた後処理が容易であること.
美観性	形, 構成が立派で美しいこと.
その他	……

表 2: 客観的・定量的な評価規範の一例

判断基準	説　明
質量	構造全体の質量
強度	各種最大耐荷重
剛性	荷重作用点の変位
速さ	要求機能の実行時間
動特性	固有振動数, ダンピング特性
精度	位置精度, 時間精度
燃費	消費燃料や消費電力
その他	……

に, これらの評価項目は互いに対立することも多い. 最適設計を行う際には, このような複数の判断基準の取扱いについても吟味する必要がある.

副評価規範の制約化

複数の評価規範を取扱うためのもっとも単純な考え方は, そのうちの一つを実際に最適化する評価規範 (主評価規範 (main criterion) と呼ぶ) として選択し, 他の評価規範 (副評価規範 (sub criteria) と呼ぶ) を制約に置き換えてしまうことである. 例えば, 構造設計において剛性と質量がともに設計評価規範として与えられているとき, 剛性の下限を定めてその範囲内で質量の低下をはかる場合と質量の上限を定めてその範囲内で剛性の向上をはかる場合がこれに相当する.

どの規範を主評価規範として選択するか, 副評価規範にもとづく制約条件をどのように設定するか, といった問題は設計の状況に大きく依存する. 例えば, 客観的・定量的評価規範として複数の規範が存在する場合に, それらをこの考え方にもとづいて取扱うのであれば, 前述したように, 本来の設計目的やその根本となっている評価規範に立ち戻って検討することが重要となるであろう.

評価規範の統合

複数の規範を何らかの手段によって統合し, それらの全てを勘案した新たな単一の規範を構築し, これによって設計を評価することも考えられる. 例えば, 複数の主観的・定性的な評価規範にもとづく概念設計過程では, 設計者は積極的に意識することなくこのような統合を行い, それにもとづいて設計案を評価していると言える.

基本設計, 詳細設計の段階において客観的・定量的な複数の評価規範を統合する場合には, 明確に定式化された統合の様式が用いられる. これについては 4.2 節で述べる.

3.4　暗黙の評価規範

定量的評価が可能な (単一の/あるいは統合された) 規範が構築されれば, 様々な最適化手法が適用可能となる. 次節以降, 主として数理的最適化手法の活用を中心に述べるが, その前に, 最も古典的かつ (ある意味では) 実用的な手法について簡単に述べておきたい. それは「根性最適化」とも言うべき手法である. これは, 適当な初期設計から始めて, 設計者の勘と経験と嗜好にもとづく設計変数の改変と再評価を, 気力と体力と納期の許す限り繰り返し, もっともよい評価値を得た設計を採用するものである.

このような手法により得られた設計解の最適性は何ら保証されないし, 一般的には非効率的な手法である[7]. しかしながら, 最終的に得られた設計解は, 少なくとも評価を試みた解候補の中では "最適" なものと言うことができる. また, 根性最適化では, 設計者が特に意識しておらず, 陽な形で表れていない規範であっても設計変数の改変の際に勘案されうる. このような規範を暗黙の評価規範 (implicit criterion) と呼ぶ. 根性最適化による解と数理的最適化による解を比較した場合に, 明示的に評価に用いた規範の観点からは後者が優れていても直観的には前者が優れていると判断できる場合, 数理的最適化の際に考慮されなかった暗黙の評価規範が設計者の判断基準として存在する可能性がある.

暗黙の評価規範は, 設計者個人の経験によって裏付けられていることも多く, 明示的な評価規範や制約条件を補うものとなっている場合もある. こういった規範は多くの場合, 主観的・定性的なものであり, その数理的な取扱いは困難である. しかしながら, 根性最適化により設計解が得られている場合であれば, それを何らかの指標, 例えば設計変数の基準値として用いることにより, 数理的最適設計問題を調整することもできる.

4　最適設計問題の定式化

全ての最適設計問題は, 概念的には以下の形式で定式化することができる.

$$\begin{aligned} &\text{Minimize} \quad g = g(\boldsymbol{x}) \quad \text{with respect to } \boldsymbol{x} \\ &\text{subject to} \begin{cases} \boldsymbol{h}(\boldsymbol{x}) &= \underline{\boldsymbol{h}} \\ \boldsymbol{c}(\boldsymbol{x}) &\leq \underline{\boldsymbol{c}} \end{cases} \end{aligned} \tag{1}$$

ここで, $\boldsymbol{x}$ は設計変数, $g(\boldsymbol{x})$ は評価規範に対応する目的関数 (objective function), $\boldsymbol{h}(\boldsymbol{x}) = [h_1(\boldsymbol{x}), \cdots, h_{N_h}(\boldsymbol{x})]^T$ および $\underline{\boldsymbol{h}} = [\underline{h}_1, \cdots, \underline{h}_{N_h}]^T$ は等式制約条件 (equality constraint condition), $\boldsymbol{c}(\boldsymbol{x}) = [c_1(\boldsymbol{x}), \cdots, c_{N_c}(\boldsymbol{x})]^T$ および $\underline{\boldsymbol{c}} = [\underline{c}_1, \cdots, \underline{c}_{N_c}]^T$ は不等式制約条件 (inequality constraint condition) を表す. ここでは, 主として数理的最適化を想定した場合のそれらの選択, 構築, 表現について概説する.

---

[7] とはいえ, 経験を積んだ技術者は最適化の鍵となる設計変数の改変を瞬時に見抜くだけの能力を持っていることも多く, その効率や精度は設計者個人の資質に大きく依存する.

4.1 設計変数

設計変数 $\boldsymbol{x}$ は, 設計解を決定するパラメータをまとめたものとして定義される. 例えば, 前述したコート掛け問題における詳細設計段階では, 設計変数は 2 本の部材の断面積 x_1, x_2 をまとめたベクトル $\boldsymbol{x} = [x_1, x_2]^T$ となる. 設計変数の表現形式として以下のようなものが挙げられる.

連続 (実数) 変数ベクトル

設計変数の構成要素が全て実数変数の場合である. 製品設計の際に決定しなければならない, 種々の部品の配置位置や寸法等をまとめたものがこれに相当する. トラス構造の設計問題を例とすれば, 前述した例に挙げた部材断面積に加えて節点位置を設計変数とする場合もこれに相当する.

離散 (整数) 変数ベクトル

設計変数の構成要素が全て離散値をとる変数の場合である. なお, この場合, 一般性を損なうことなく, 全ての変数が整数であるとして設計問題を再構築することもできる.

構造材の材質の決定や, 既製の製品群の中からの部品選択の場合がこれに相当する. また, 前述したような部材断面積を設計変数とする場合であっても, 既製品である構造材を使用する際などは, 実際には離散変数となることもある.

形状関数

設計変数が関数となる場合である. 例えば, 以下のように表される.

$$\boldsymbol{x} = \boldsymbol{x}(t) \qquad (0 \leq t \leq 1) \tag{2}$$

連続体構造の形状を設計する場合がこれに相当する. この場合, べき多項式や三角多項式等[8] の, 必要な境界条件を満足する適当な多項式系で関数を表現することにより, 設計変数の実数変数ベクトル表現を得ることもできる. 例えば設計変数が, ある基準線からのオフセット値に相当するスカラ値関数 $x(t)$ である場合,

$$x(t) = a_0 + a_1 \cos 2\pi t + a_2 \sin 2\pi t + a_3 \cos 4\pi t + \cdots + a_{2n-1} \cos 2n\pi t + a_{2n} \sin 2n\pi t \qquad (0 \leq t \leq 1) \tag{3}$$

として $2n+1$ 項の三角多項式によって形状関数を表せば, 実数設計変数ベクトル $\boldsymbol{x} = [a_0, a_1, \cdots, a_{2n}]^T$ が得られる.

その他の表現

その他にもこれらの組合せをはじめとして様々な設計変数の表現が可能である.

例えば, 予め格子状に配置した節点に対して多数のトラス部材を想定した構造 (これを背景構造 (ground structure) と呼ぶ) の部材断面積を設計変数とする手法がある. この場合, 直接操作する設計変数は部材断面積ベクトルであるが, 最適化処理の過程で不要と判断された部材断面積をゼロとして消去してゆくことにより, 仮想的に, トラス構造のトポロジや形態を設計変数として扱っていると言うこともできる (図 3).

4.2 評価規範

評価規範に対応する目的関数 $g(\boldsymbol{x})$ は, 設計変数 $\boldsymbol{x}$ のスカラ値関数である. その表現様式には多種多様なものが存在するが, 本稿では, それらのうちの代表的なものについて概説する.

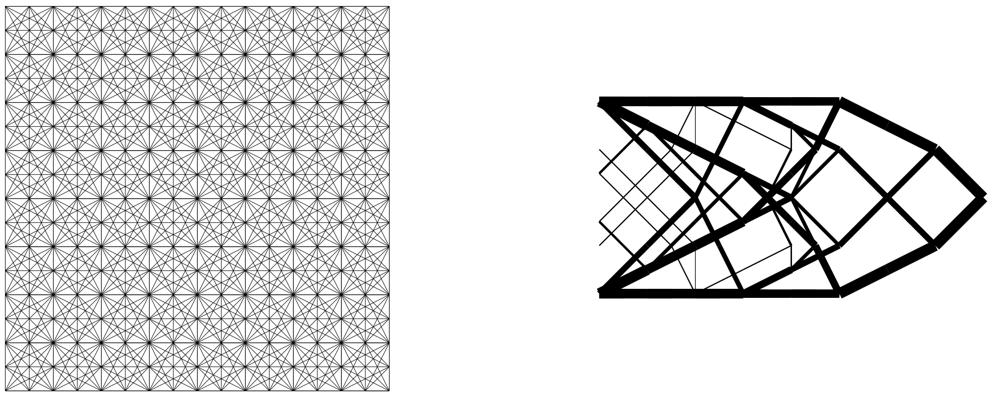
線形評価規範

設計変数が実数あるいは整数 (離散値) のベクトルであり, 目的関数が以下のような線形関数で表現される場合である.

$$g(\boldsymbol{x}) = \boldsymbol{s}^T \boldsymbol{x} = \sum_i s_i x_i \tag{4}$$

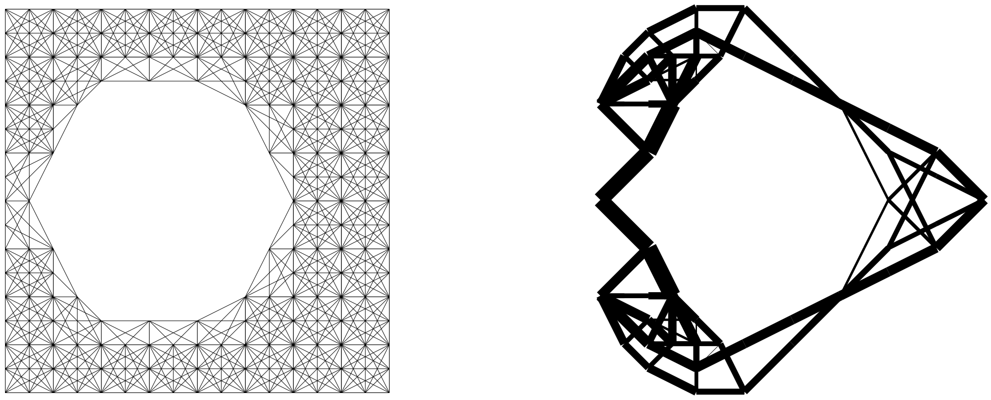
ここで, $\boldsymbol{s}$ は個々の設計変数の要素に対する係数ベクトル, あるいは重みベクトルである. 例えば, トラス構造設計において設計変数が部材断面積のみであるときの全体の質量がこれに相当する.

[8] この他にもルジャンドル多項式, チェビシェフ多項式等さまざまなものがある. 一般には直交関数系 (11) を用いるほうが定式化や計算の観点からは便利であることが多い.



背景構造　　最適解

(a) 幾何形状制約無し



背景構造　　最適解

(b) 幾何形状制約有り

図 3: 仮想的にトポロジを扱ったトラス構造の設計例 (12)

設計変数が実数変数ベクトルである場合, 線形評価規範の同一評価値に対応する解空間は設計変数空間の超平面となる. また, 一般に, この場合には最適設計解は設計可能領域の境界に存在する.

二次形式評価規範

設計変数が実数あるいは整数 (離散値) のベクトルであり, 目的関数が以下のような設計変数の二次形式で表現される場合である.

$$g(\boldsymbol{x}) = \frac{1}{2}(\boldsymbol{x}-\boldsymbol{r})^T \boldsymbol{N}(\boldsymbol{x}-\boldsymbol{r}) \tag{5}$$

ここで, $\boldsymbol{r}$ は定数ベクトル, また, $\boldsymbol{N}$ は二次形式規範を規定する正方行列であり, 一般には, 正定値対称行列である. 例えば, $\boldsymbol{r}$ として設計変数のある種の基準値を与え, $\boldsymbol{N}$ として適当な対角行列を用いれば, この規範は設計変数の基準値からの偏差を重みつきで評価するものとなる.

汎関数評価規範

設計変数が形状関数であり, 目的関数がその積分形式で汎関数として表現される場合である. 例えば, 設計変数がスカラ値関数 $x(t)$ $(0 \leq t \leq 1)$ である場合, その一般形は以下のように表される.

$$g[x] = \int_0^1 F(t, x, x')dt \tag{6}$$

ここで, x' は x の導関数であり, F は t, x, x' の関数である.

このような目的関数を持つ古典的な最適設計問題として等周問題が挙げられる．例えば Dido 問題[9] として知られている,「与えられた周囲長によって囲むことのできる面積が最大となる図形を求めよ」というのはその典型である．

多目的評価規範

複数の評価規範にもとづく最適設計では, 3.3 節で述べたように, 副評価規範を制約にする手法の他に, 全ての規範を統合する手法がある．複数評価規範に対応する目的関数 $g_1(\boldsymbol{x}), \cdots, g_{N_g}(\boldsymbol{x})$ を考える．この場合, 評価規範の統合は, 概念的には, 以下のようなスカラー値関数 $\hat{g}$ を決定することにほかならない．

$$g(\boldsymbol{x}) = \hat{g}(g_1(\boldsymbol{x}), \cdots, g_{N_g}(\boldsymbol{x})) \tag{7}$$

統合関数 $\hat{g}$ として様々な形式のものが考えられるが, 最も多く用いられているのは, 次式のように表される線形結合によるものである．

$$\hat{g}(g_1(\boldsymbol{x}), \cdots, g_{N_g}(\boldsymbol{x})) = w_1 g_1(\boldsymbol{x}) + \cdots + w_{N_g} g_{N_g}(\boldsymbol{x}) \tag{8}$$

ここで, $w_1, \cdots, w_{N_g}$ は個々の目的関数に対する重み係数 (weight coefficients) である．

多目的評価規範にもとづく最適設計においては, これらの重み係数を適切に設定することが重要である．そのためには「評価規範**を**評価する規範」が必要となるが, そのようなものが活用できる場面はほとんどないであろう．この場合にも 3.3 節で述べたように, 本来の設計目的とその根本となっている評価規範に立ち戻って重み付けを検討することが求められる．また, 評価規範の数が比較的少ない (例えば N_g が 2 ないし 3 である) 場合には, 重み係数を様々に変化させて最適解の集合 (パレート (Pareto) 最適解) を求め, 得られた解集合の中から, 設計者が自身の選好基準 (preference) にもとづいて最終的な設計解を決定することもできる．

その他の評価規範

おそらく, 実際的な意味で最も多く用いられる評価規範の表現様式は, 有限要素法等に代表される高度に数値的な手法にもとづくものであろう．この場合, 最適設計処理の過程で行われる改変により, 設計変数の値が計算処理プログラムの適用範囲から外れたり, 取扱い困難なものとなる[10] 可能性があるため, 特に注意が必要である[11]．

数値計算処理の結果得られた多数の情報をスカラ値に統合することにより, 目的に応じた新たな評価規範を構築することもできる．また, 後述するが, 近年ではニューラルネットワーク等を用い, 定量化が困難な評価規範を学習によって獲得することにより, 目的関数を構築する手法も提案されている (9)．

4.3 制約条件

制約条件は, 制約条件関数 $\boldsymbol{h}(\boldsymbol{x})$, $\boldsymbol{c}(\boldsymbol{x})$ および制約値ベクトル $\underline{\boldsymbol{h}}$, $\underline{\boldsymbol{c}}$ によって構成される．ここでは, 等式制約と不等式制約について概説し, それらの制約の取扱いについて簡単に述べる．

等式制約

製品が果たすべき機能に関する制約等が等式制約として与えられる．例えば, 前述のコート掛け問題の場合, コートを支持する位置に関する制約は, 等式制約となる．

等式制約は常に全てが有効 (active) である．したがって, 設計変数 $\boldsymbol{x}$ が実数ベクトルであり, 等式制約条件数 N_h がその次元と等しい場合, 設計解は制約条件関数の逆関数を用いることにより, $\boldsymbol{x} = \boldsymbol{h}^{-1}(\underline{\boldsymbol{h}})$ として評価規範とは無関係に定められる場合がある[12]．

不等式制約

製品の性能に関する制約等が不等式制約として与えられる．例えば, 前述のコート掛け問題の場合, 要求される耐荷重は不等式制約となる．また, コスト (材料コスト, 製造コスト等) に関連する制約も不等式制約となることが多い．

[9] 紀元前 900 年ごろ, 残忍な兄から逃れたフェニキアの王女 Dido は, 後にカルタゴになるアフリカの地にたどり着いたとき, 土地の購入を牛一頭分の皮で囲めるだけしか認められなかった．そこで彼女は皮をできるだけ細長く切ってつなぎあわせて一本の長い皮ひもを作り, それによって最大の面積を囲むようにして土地を手に入れた．

[10] 例えば, 設計変数である要素の厚みや密度を小さくした結果, 剛性マトリクスが特異になってしまう場合などがこれに相当する．

[11] この点に関しては数値的な処理を用いる制約条件関数も同様である．

[12] ただし, 等式制約の種類によってはそれを満たす設計解が複数存在する場合もある．また, 形式的にはこのように書くことができるが, 実際の計算が可能かどうか, 処理時間が現実的かどうか, といった点は制約条件関数 $\boldsymbol{h}(\boldsymbol{x})$ に依存する．

最適設計問題 (1) では制約値として上限のみを考えているが, 上下限双方に制約値を持つ場合もある．この場合は下限に関する制約条件に負号 (−) を付与することにより, 問題 (1) に帰着することができる．また, 設計変数の範囲を直接制限する不等式制約

$$\underline{\boldsymbol{x}}_L \leq \boldsymbol{x} \leq \underline{\boldsymbol{x}}_H \tag{9}$$

は, 適用する最適化手法によっては極めて容易に扱える場合がある[13]．このような場合, 不等式制約をこの形式に帰着させることにより, 処理量を大きく軽減できる可能性がある．

等式制約とは異なり, 不等式制約は全てが有効であるとは限らない．例えば, 2 つの制約条件, $x \leq 3$ および $x \leq 5$ がある場合, 後者は常に無効 (inactive) な制約であり, 無視することもできる．

不等式制約よりも等式制約のほうが扱いは容易であることから, 有効な不等式制約を等式制約に置き換えるというアプローチも可能ではある．しかしながら, 当該問題の最適化に関連する有効な不等式制約を先験的に検出することは, 一般には困難な作業である．

最適化問題における制約条件の取扱い

等式制約条件の場合, これを取扱う際の最も基本的な考え方は, 制約条件式を用いて制約数と同数の設計変数を消去し, 最適化問題を制約を含まない形に書き換えることである．ただし, このような手法は, 全ての設計変数が対等なものである場合であってもそのうちのいくつかを消去すべき変数として半ば恣意的に選択することでもあり, やや一般性, 合理性に欠けると言える．後述するラグランジュ乗数法 (Lagrangian multiplier method) を適用すれば, このような場合にも一般性のある体系的な取扱いが可能となる．

等式制約のみならず不等式制約を含めて取扱いが可能で, 多くの最適化手法に適用できる, 広く用いられているアプローチとしてペナルティ関数法 (penalty function method)[14] がある．これは, 最適設計問題 (1) を以下のような制約無し最小化問題として解く手法である．

$$\text{Minimize} \quad p = p(\boldsymbol{x}) = g(\boldsymbol{x}) + \sum_{i}^{N_h} \gamma_i^h (h_i(\boldsymbol{x}) - \underline{h}_i)^2 + \sum_{i}^{N_c} \gamma_i^c \max(0, c_i(\boldsymbol{x}) - \underline{c}_i)^2 \quad \text{with respect to } \boldsymbol{x} \tag{10}$$

ここで, γ_i^h および γ_i^c はそれぞれ i 番目の等式制約および不等式制約に関するペナルティ係数 (penalty coefficient) である．ペナルティ関数法は, 制約の満たされない部分に対し, ペナルティ係数を乗じた値を目的関数に加えることにより, これを優先的に最小化する手法である．

ペナルティ関数法には, ペナルティ項の構築の仕方等によって様々な考え方がある．詳細については文献 [13] 等を参照されたい．

5　最適化手法概観

最適設計問題がひとたび定式化されれば, その表現様式に適した最適化手法を用いてこれを解くことになる．全ての観点において秀でている, 万能の最適化手法は存在しない．例えば, 3.4 節でふれた根性最適化は, 原理的にはあらゆる問題に適用可能であるが, その得られた解の最適性の検証は非常に困難である．

適切な最適化手法を選択するには, その特徴を把握することが必要不可欠である．最適化手法には既に多種多様なものが存在し, 現在も新たな手法が次々と開発されている．ここでは, それらの最適化手法のうちのいくつかを選んで概説する．

5.1　関数極値法

17 世紀にニュートンとライプニッツが開発した微分法は, 最も基本的な最適化手法と言うことができる．例えば, 設計変数 x がスカラである, 以下のように定式化された制約無し最適設計問題

$$\text{Minimize} \quad g = g(x) \quad \text{with respect to } x \tag{11}$$

は, $dg/dx = 0$ となる x を求める問題に帰着される[15]．これに端を発する, 数学的に関数の極値を求めるいくつかの手法は, 目的関数や制約条件の性質がよい場合には現在でも広く用いられている．

---

[13] それどころか, 特にヒューリスティックな手法の場合には, こういう制約の存在を前提としているものも少なくない．

[14] 「罰金法」と呼ばれることもある．最小化すべき目的関数をコストであると見なし, 侵した制約に見合う高額の罰金をコストに追加することにより, 罰金を払わないような解を探索させる, という考え方である．

[15] 目的関数 $g(x)$ が微分可能であることが条件である．また, 実際には, 得られた x が最小値 (極小値) であるためには, 二次導関数 d^2g/dx^2 の符号を調べる必要がある．

ラグランジュ乗数法

設計変数 $\boldsymbol{x}$ が実数ベクトルであり, 等式制約条件が付与されている, 以下のように定式化された最適設計問題を考えよう.

$$\begin{aligned} \text{Minimize} \quad g = g(\boldsymbol{x}) \quad & \text{with respect to } \boldsymbol{x} \\ & \text{subject to } \boldsymbol{h}(\boldsymbol{x}) = \underline{\boldsymbol{h}} \end{aligned} \tag{12}$$

ラグランジュ乗数 (Lagrangian multiplier) と呼ばれるパラメータ $\boldsymbol{\lambda}$ を導入すれば, この問題を次のような制約無し停留問題に書き換えることができる.

$$\text{Make stationary} \quad f = f(\boldsymbol{x}, \boldsymbol{\lambda}) = g(\boldsymbol{x}) + \boldsymbol{\lambda}^T (\boldsymbol{h}(\boldsymbol{x}) - \underline{\boldsymbol{h}}) \quad \text{with respect to } \boldsymbol{x}, \boldsymbol{\lambda} \tag{13}$$

この問題の解は, 新たな目的関数 f を $\boldsymbol{x}$ および $\boldsymbol{\lambda}$ で微分することにより得られる.

$$\frac{\partial f}{\partial \boldsymbol{x}} = 0 \tag{14}$$

$$\frac{\partial f}{\partial \boldsymbol{\lambda}} = \boldsymbol{h}(\boldsymbol{x}) - \underline{\boldsymbol{h}} = 0 \tag{15}$$

ここで, 式 (15) が問題 (12) の制約条件式であることに注意しよう. このことから, 連立方程式 (14) および (15) を解いて $\boldsymbol{x}$ と $\boldsymbol{\lambda}$ を求めれば, $\boldsymbol{x}$ はもとの問題 (12) の解となる[16] .

例えば, 4.2 節で紹介した二次形式評価規範 (5) と線形等式制約で構成される, 以下の最適設計問題

$$\begin{aligned} \text{Minimize} \quad g = \frac{1}{2}(\boldsymbol{x} - \boldsymbol{r})^T \boldsymbol{N} (\boldsymbol{x} - \boldsymbol{r}) \quad & \text{with respect to } \boldsymbol{x} \\ & \text{subject to } \boldsymbol{H}\boldsymbol{x} = \underline{\boldsymbol{h}} \end{aligned} \tag{16}$$

の解はラグランジュ乗数法を用いて次のように求めることができる.

$$\boldsymbol{x} = \boldsymbol{H}^+ \underline{\boldsymbol{h}} + \left(\boldsymbol{I} - \boldsymbol{H}^+ \boldsymbol{H} \right) \boldsymbol{r} \tag{17}$$

$$\boldsymbol{H}^+ = \boldsymbol{N}^{-1} \boldsymbol{H}^T \left(\boldsymbol{H} \boldsymbol{N}^{-1} \boldsymbol{H}^T \right)^{-1} \tag{18}$$

ここで得られた $\boldsymbol{H}^+$ は, $\boldsymbol{N}$ をノルム行列とする $\boldsymbol{H}$ の一般化逆行列[17] である. なお, 問題 (16) の, より一般的な形式が文献 (15) で扱われている.

変分法

変分法は, 微分法による関数の極値問題を汎関数に拡張したものである. 詳細については割愛するが, 例えば, 4.2 節で紹介した汎関数評価規範 (6) による最適設計問題

$$\text{Minimize} \quad g = g[x] = \int_0^1 F(t, x, x') dt \quad \text{with respect to } x(t) \tag{19}$$

を考えれば, この停留条件は, 変分記号 δ を用いて $\delta g = 0$ と表すことができる. このとき, 形状関数 $x(t)$ は以下に示すオイラー方程式 (Euler's equation)[18] を満たさなければならない (16).

$$\frac{d}{dt} \left(\frac{\partial F}{\partial x'} \right) - \frac{\partial F}{\partial x} = 0 \tag{20}$$

式 (20) を満足する関数 $x(t)$ に境界条件をあてはめることにより, 問題 (19) の解が得られる.

5.2 計算的手法

実際的な最適化問題を解くために, コンピュータの数値計算能力を活用した様々な計算的手法が開発されている. それらのうちの代表的なものを簡単に紹介する.

[16] 正確には, これは最小化問題の解ではなく極値問題の解にすぎない.

[17] 通常の逆行列は (正則な) 正方行列に対してのみ定義されるが, このような一般逆行列を用いれば, 正方でない行列による線形方程式の解の一つを逆行列を用いた場合と同様の形式で求めることができる. 厳密な数学的定義等の詳細については文献 (14) 等を参照せよ.

[18] オイラー・ラグランジュの微分方程式 (Euler-Lagrange differential equation) とも呼ばれる.

線形計画法

設計変数が実数ベクトルであり, 目的関数および全ての制約条件関数が線形であるとき, 最適設計問題 (1) は線形計画問題 (linear programming problem) に帰着できる. この場合, 最適化問題は以下のような形式で定式化される[19].

$$\text{Maximize} \quad g = \boldsymbol{s}^T \boldsymbol{x} \quad \text{with respect to } \boldsymbol{x}$$
$$\text{subject to} \begin{cases} \boldsymbol{C}\boldsymbol{x} & \leq & \underline{\boldsymbol{c}} \\ \boldsymbol{x} & \geq & 0 \end{cases} \tag{21}$$

この問題を解くための代表的な手法としては, シンプレックス法 (simplex method) がよく知られている.

線形計画問題 (21) の様式は高い汎用性を持つため, これを解くための数多くの計算機プログラムがこれまでに開発されている. 計算機環境によっては, それらのプログラムの何れかを直ちに利用可能である場合も少なくない. また, 目的関数や制約条件関数が非線形であっても, 設計変数空間上の適当な点を基準とする線形化が可能であれば, 局所的な線形計画問題の構築による解の探索を繰り返す, いわゆる逐次線形計画法 (sequential linear programming method) が適用できる場合もある.

非線形計画法

設計変数が実数ベクトルであり, 目的関数が非線形である場合の数理的最適化手法を, 一般に非線形計画法 (nonlinear programming method)(13) と言う. その中でも代表的なものに種々の降下法 (descent method) がある (図 4). これは, 適当な初期値 $\boldsymbol{x}^{(0)}$ から解の探索を始めて

$$g(\boldsymbol{x}^{(k+1)}) < g(\boldsymbol{x}^{(k)}) \quad k = 0, 1, 2, \cdots \tag{22}$$

となるように,

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha^{(k)} \boldsymbol{d}^{(k)} \tag{23}$$

として点列 $\{\boldsymbol{x}^{(k)}\}$ を生成する手法である[20]. ここで, $\boldsymbol{d}^{(k)}$ は $\boldsymbol{x}^{(k)}$ において目的関数を減少させる方向を表すベクトルであり, $\alpha^{(k)}$ はステップ幅である. 方向ベクトル $\boldsymbol{d}^{(k)}$ として目的関数の勾配

$$\boldsymbol{d}^{(k)} = -\nabla g = -\frac{\partial g}{\partial \boldsymbol{x}} \tag{24}$$

を用いる場合を特に勾配法 (gradient method) と呼ぶこともある. よく知られている最急降下法 (steepest descent method) では, 方向ベクトル $\boldsymbol{d}^{(k)}$ として $-\nabla g$ を用い, 直線探索により降下度が大きくなるようにステップ幅 $\alpha^{(k)}$ を決定する.

目的関数の勾配 ∇g が計算可能である場合には, 降下法は比較的効率の良い最適化手法である. ただし, 一般に, 降下法で得られる解は初期値に依存する, いわゆる**局所最適解**の問題 (後述) がある. また, 制約を扱うには, 前述したペナルティ関数法や種々の射影法 (projection method)(17) が用いられる.

目的関数の勾配が計算できない場合の手法としてシンプレックス法 (simplex method)[21] を挙げておく (18). これは, N_x 次元設計変数空間上の $N_x + 1$ 個の点によって構成される単体 (シンプレックス) を用いる. シンプレックスを構成する個々の点における目的関数値の比較を行い, その結果によってシンプレックスを逐次的に改変し, 設計変数空間を探索してゆく手法である. シンプレックス法は勾配法に比して効率が良い手法であるとは言えないが, 目的関数の値のみから最適化が行えるなど比較的汎用性の高い手法である.

局所最適解と大域的最適解

非線形計画問題において, 制約によって定まる設計可能領域が凸でない場合や目的関数が多峰性を有する場合, 局所最適解 (local optimum) の問題が存在する (図 5). 一般に, 降下法に代表される繰り返し最適化手法では, 適当な初期値からの近傍探索により解の改善を行い, 近傍での解の改善が不可能となった時点で探索を終了する. このようにして得られた解は非線形計画問題の真の最適解 (大域的最適解 (global optimum)) の必要条件を満たしてはいるが, 大域的最適であるという保証はない.

一般に, 非線形計画法によって得られた解が大域的最適解であるかどうかを判定するのは非常に困難である. 局所最適の問題は, 最適化手法で得られる解の初期値依存性の問題にほかならない. 実際的な対処方法としては, 最適化処理を複数の初期値から行い, 得られた解を比較すること等があげられる.

厳密な最適解の探求という学問的な観点からは, 局所最適をどのように克服して大域的最適解を得るかという問題は非常に重要である. しかしながら, 最適設計の実際的な側面で考えれば, 目的関数や制約条件関数の非線形性の度合にも依

[19] 線形計画問題は, 通常, 最大化問題として定式化されるため, このように記述している.
[20] 要するに, 設計変数の空間を, とにかく目的関数が小さくなるように下ってゆくわけである. なお, 目的関数の最大化問題の場合には, 同様の手法が「山登り法」と呼ばれる場合がある.
[21] 前述の線形計画法におけるシンプレックス法と名称の由来は同じであるが, 直接的な関係はない.

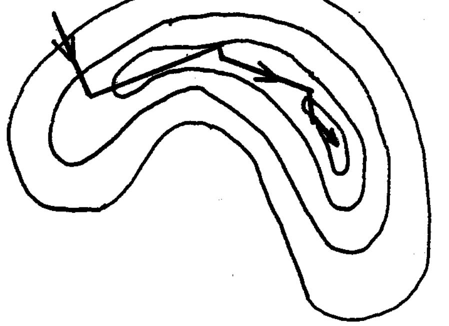


図 4: 降下法による最適解の探索

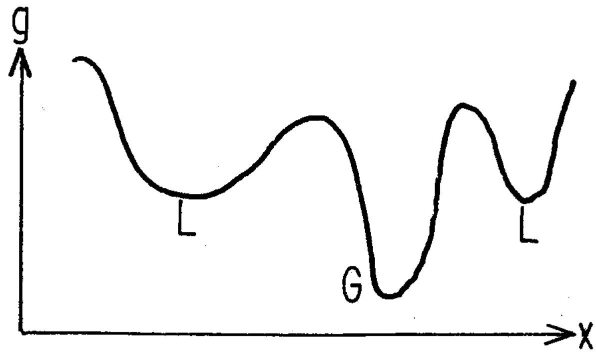


図 5: 目的関数の多峰性と局所最適解
(局所最適解 (L) と大域的最適解 (G))

存するが, 得られた解がよほど悪い局所最適解でない限り最適化の目的は ある程度 達成されていると考えられる. 前述した初期値の複数化等を活用し, 局所最適の問題を常に認識しつつ, しかし局所最適を過度に恐れずに最適化に挑むべきであろう.

5.3 ヒューリスティック手法

数理的最適化手法は有用ではあるが, 問題の種類によっては計算処理時間が実用的でなくなる場合がある. また, これらの手法を適用するためには目的関数や制約条件を数理的に明確な形で定式化する必要があり, 主観的・定性的な評価規範を直接取扱うことができない.

このような背景の下で, 大規模化・複雑化する最適化問題に対し, 正確ではなくとも実用に耐えうる, あるいは目安となる最適解を実際的な処理時間で求めるための手法がいくつか開発されている. これらの中心的なものは自然界の様々な現象にヒントを得たものであり, 総じてヒューリスティック手法 (heuristic method) と呼ばれる[22]. ここでは, ヒューリスティック手法のうち, ニューラルネットワークについて簡単に紹介する.

ニューラルネットワーク

ニューラルネットワーク (neural network) は, 脳神経回路網をモデル化した情報処理手法である. 実際の脳では, 膨大な数の神経細胞 (ニューロン) が複雑に結合することによって情報処理が行われているが, 工学問題の解決にはやや単純化したモデルが用いられる. ここでは, 階層形ニューラルネットワークを用いた最適化問題の解法[19]について解説する.

図 6(a) に階層形ニューラルネットワークを示す. これは, 図 6(b) に示す多入力一出力のニューロンを層状に配置したものである. 個々のニューロンのふるまいは次式で表される.

$$v_i^k = f(u_i^k), \quad u_i^k = \sum_j w_{i\,j}^{k\,k-1} v_j^{k-1} \tag{25}$$

ここで, u_i^k および v_i^k は第 k 層の i 番目のニューロンの内部状態およびその出力, $w_{i\,j}^{k\,k-1}$ は第 k 層の i 番目のニューロンに対する第 $k-1$ 層の j 番目のニューロンの結合の重みを表す. また, f はニューロンの特性関数であり, 図 6(c) に示すようなシグモイド状のものを用いるのが一般的である. ニューラルネットワークの入出力関係は, 個々のニューロンのふるまいを統合したものとして, 以下のような形式で考えることができる.

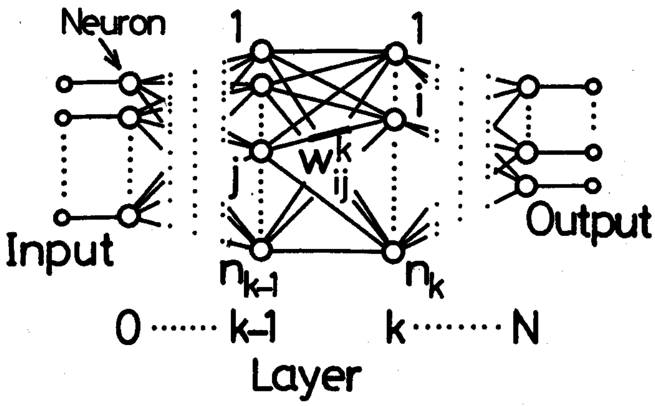
$$\boldsymbol{v}^N = \boldsymbol{v}^N(\boldsymbol{v}^0, \boldsymbol{w}) \tag{26}$$

ここで, $\boldsymbol{v}^0 = [v_i^0]$ および $\boldsymbol{v}^N = [v_i^N]$ は入力値ベクトルおよび出力値ベクトル, $\boldsymbol{w} = [w_{i\,j}^{k\,k-1}]$ は入出力特性を決定する結合の重みをまとめたものである.

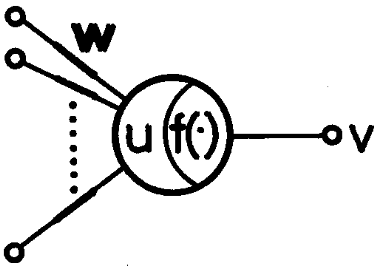
望ましい入出力関係を学習により構築する手法として誤差逆伝播学習 (error back-propagation learning)[20] が提案されている. これは, ある入力に対する望ましい出力 $\underline{\boldsymbol{v}}^N = [\underline{v}_i^N]$ との誤差

$$E = \frac{1}{2} \sum_i (v_i^N - \underline{v}_i^N)^2 \tag{27}$$

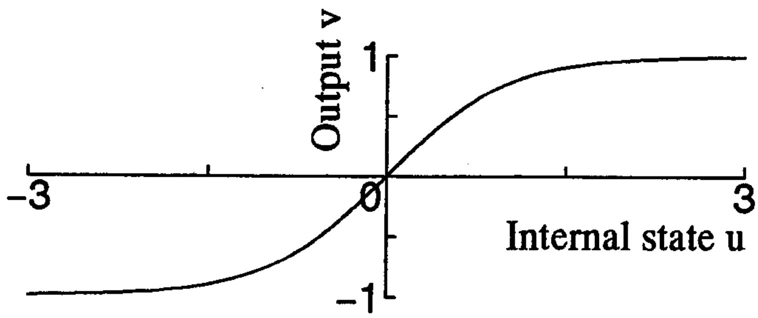
[22] この "heuristic" という言葉はギリシャ語の「発見」を意味する *heuriskein* に由来することから, 「発見的手法」と訳されることもある. なお, *heuriskein* の過去形は, アルキメデスが浮力の原理を発見した時に叫んだ, 有名な「ユーレカ!(*eureka!*)」である.



(a) 階層形ニューラルネットワーク



(b) ニューロン



(c) ニューロンの特性関数

図 6: 階層形ニューラルネットワーク

を小さくするように以下の重み結合の修正を反復的に行うものである.

$$w_{i\,j}^{k\,k-1} \leftarrow w_{i\,j}^{k\,k-1} - \Delta w_{i\,j}^{k\,k-1}, \quad \Delta w_{i\,j}^{k\,k-1} = \epsilon \frac{\partial E}{\partial w_{i\,j}^{k\,k-1}} \tag{28}$$

ここで, ϵ は適当な微小定数であり, $\Delta w_{i\,j}^{k\,k-1}$ は出力側の誤差に対する修正量が伝播されてくる形で, 次のようにして得られる.

$$\begin{aligned} \Delta w_{i\,j}^{k\,k-1} &= \Delta v_i^k \cdot f'(u_i^k) \cdot v_j^{k-1} && (29)\\ \Delta v_i^k &= \sum_j \Delta v_j^{k+1} \cdot f'(u_j^{k+1}) \cdot w_{j\,i}^{k+1\,k} && (30)\\ \Delta v_i^N &= \epsilon \frac{\partial E}{\partial v_i^N} = \epsilon(v_i^N - \underline{v}_i^N) && (31) \end{aligned}$$

この逆伝播過程に注目すれば, 入力側では修正量として $\boldsymbol{\Delta v}^0 = \epsilon(\partial E/\partial \boldsymbol{v}^0)$ が得られていることがわかる.

階層形ニューラルネットワークに入出力関係として最適化問題の目的関数 $g(\boldsymbol{x})$ を学習させた場合を考える (すなわち $v^N = g$, $\boldsymbol{v}^0 = \boldsymbol{x}$). 学習終了後は重み $\boldsymbol{w}$ を固定とし, 式 (28) による学習は行わないものとする. ここで, 新たな誤差関数としてニューラルネットワークの出力値それ自身, すなわち学習により獲得した目的関数の値を用いる ($E = v^N = g$). 式 (30), (31) にしたがって誤差逆伝播を行えば, ニューラルネットワークの入力では

$$\boldsymbol{\Delta v}^0 = \epsilon \frac{\partial E}{\partial \boldsymbol{v}^0} = \epsilon \frac{\partial g}{\partial \boldsymbol{x}} = \epsilon \nabla g \tag{32}$$

として <u>目的関数の勾配</u> が得られることから, これを利用した降下法による最適化処理 (前節参照) を行うことができる.

この手法は, 学習によって目的関数を獲得しているため, 数理的な取扱いが困難な評価規範に対しても適用可能である. また, 学習後は目的関数を計算する必要がないため, 学習に必要なデータさえ生成することができれば, 多大な計算コストを要する目的関数による最適化問題であっても比較的短時間で解くことも可能である. ただし, 基本的には降下法であり, 局所最適解の問題は存在する.

ここでは階層形ニューラルネットワークによる最適化手法について解説したが, 相互結合形ニューラルネットワークが持つ, ネットワークの 'エネルギ' を最小化する特性を活用した最適化手法 (いわゆるホップフィールド・モデルやボルツマン・マシン)[21] も広く研究されている.

ニューラルネットワーク以外のヒューリスティック手法としては, 生物の進化を参照した遺伝的アルゴリズム (genetic algorithm: GA), 金属材料の焼きなまし現象を参照したシミュレーテッドアニーリング (simulated annealing: SA), 細胞の挙動を参照したセルラ・オートマトン (cellular automata: CA) 等が挙げられる [22].

6 おわりに

現在では様々な最適化ツールが開発されており, それらの活用によって比較的容易に最適設計が行える場合もあるであろう. ただし, 最適化処理を行う過程それ自体はブラックボックス的な取扱いが可能であるとしても, そのツールが最適設計問題の定式化をどのように行っているのか, という点について設計者は知悉しておかなければならない. すなわち, 設計変数として何を選択し, どのような制約条件の下で, どの評価規範にもとづいて最適化を行おうとしているのか, という点を把握しておく必要がある. また, 得られた最適設計解の特性についても知っておかなければならない. その解に対する有効な制約がどれであり, どの設計変数がその制約や評価規範に関して大きな影響を持つか, といった点は設計解の具現化の際に考慮しておかなければならない事柄である.

本稿では, 最適設計を考える上での基本的な事項について, 設計過程から最適化手法までを概観した. 最適設計は非常に幅が広く奥の深い学問分野であり, 紙面の都合や筆者の非力もあってやや偏った雑駁な議論となってしまったかも知れないと危惧している. 取り上げた内容の詳細については, 個々の文献を参照していただきたい. 本稿が最適設計の研究や実施の際の何らかの参考になれば幸いである.

参考文献

(1) 瀬口靖幸, "構造最適化問題", 日本機械学会誌, Vol.92, No.847, (1989), 85-91.
(2) 日本機械学会 (編), 構造・材料の最適設計, 技報堂, (1989).
(3) 赤木新介, 設計工学 (上)—新しいコンピュータ応用設計—, コロナ社, (1991).
(4) G. ポール・W. バイツ (設計工学研究グループ訳), 工学設計 (体系的アプローチ), 培風館, (1995), 3 章.
(5) 瀬口靖幸ほか 2 名編, 機械設計工学 2, 培風館, (1987), 1.4 節.
(6) Nigel Cross, Strategies for Product Design, 2nd Eds., John Wiley & Sons, (1994), Chap.2.
(7) 岸光男, システム工学, 共立出版, (1995), 1.3 節.
(8) 西尾実ほか編, 岩波国語辞典第 5 版, (1994).
(9) 田中正夫・花原和之, "適応トラスの動作規範獲得 (モジュール形ニューラルネットによる実現)", 日本機械学会論文集 (C 編), 58 巻 550 号, (1992), 1735-1741.
(10) 第 51 回システム制御情報講習会テキスト, 定量化が困難な情報の取扱い—経験・感覚・感性—, システム制御情報学会, (2002).
(11) 竹之内脩, フーリエ展開, 秀潤社, (1978), 2-4 節.
(12) Hanahara, K. and Tada, Y., "A Recursive Structural Design Approach", Proc. WCSMO-4, (2001), 87-88.
(13) 今野浩・山下浩, 非線形計画法, 日科技連, (1978).
(14) ラオ, C. R.・ミトラ, S. K. (渋谷 政昭・田辺 国士 訳) 一般逆行列とその応用, 東京図書, (1973).
(15) 花原和之, "可変形状トラス (VGT) の多彩な動作", システム/制御/情報, Vol.45, No.2, (2001), 82-89.
(16) 長尾智晴, 最適化アルゴリズム, 昭晃堂, (2000), 2 章.
(17) 文献 [13] の 11 章.
(18) 文献 [13] の 12.3 節.
(19) 田中正夫ほか 2 名, "モジュール形ニューラルネットワークモデルによるトラス形高冗長並列アームの姿勢制御", 日本機械学会論文集 (C 編), 57 巻 533 号, (1991), 189-195.
(20) ラメルハート, D. E. ほか (甘利俊一監訳), PDP モデル (認知科学とニューロン回路網の探索), 産業図書, (1989), 8 章.
(21) 麻生英樹, ニューラルネットワーク情報処理, 産業図書, (1989), 1.3-1.4 節.
(22) 日本機械学会 (編), 工学問題を解決する適応化・知能化・最適化法, 技報堂, (1996).