博士論文

逆問題のための遺伝的アルゴリズムを用いた多次元非線形数値最適化手法の設計・開発

平成13年3月

富永大介

目 次

第1章　はじめに

古代から、生命とは何かという問いは人々を悩ませ続けてきた。これに対する科学的なアプローチは難しかったが、実験技術の進歩や近年の分子生物学の発展により、生命のハードウェア、すなわち生体そのものの構造を明らかにしようとする研究が盛んになり、現在までに多大な成果を上げている。

その結果現在では、生体内で様々な化学物質が互いに反応し合うことで、生命活動が行われていると一般的に言われるようになってきた。多くの生体内化学反応は、酵素タンパク質によって触媒される反応であり、遺伝媒体である染色体中に存在する遺伝子は、そのタンパク質の設計図であることが分かっている。つまり生命活動とは、遺伝子からタンパク質が作られ (遺伝子の発現)、タンパク質が生体内化学反応や遺伝子の発現を制御するというしくみで行われていると認識されている [1]。そういう意味では、生命の本質に迫ったとも言える。しかし現在、ヒトゲノム計画がおおよそ完了したと言われるが、これによって判明したのは染色体上の塩基配列のみである。今後の研究により、この配列から膨大な数の遺伝子を特定していくことで、生体を形作る部品のリストを入手することができると考えられる。つまり現在は、生体の構造を解明していくためのスタート地点であると言える。さらに現在、これまで行われてきた膨大な実験データから、生体の構造の一部が少しずつ明らかにされつつあるが、生体内に存在するタンパク質などの物質の種類は膨大で、一つの物質が複数の反応に関与することも多い。また反応により生成された物質が他の反応を制御することで、カスケードやループなどの制御機構が非常にたくさんあることが知られてきており、実験データが増え、生体の構造が明らかになるにつれ、生体全体はさらに複雑であることが予想されている [1]。

明らかになった一部の構造からは、さまざまな疾病の原因や発生機構が推定され、生命科学の成果が医学に応用されるようになってきている。また生体内で行われる化学反応を利用した、工業的な応用も期待されている。そのため、生体の構造解明は、生命の本質に迫る研究であるという面からも、人類に直接に利益をもたらす研究であるという面からも、非常に有益かつ有望である。

生体は一般に、一つまたは複数の細胞の集合体であり、生物の基本単位を細胞とすることが多い。そのとき、その基本単位を構成する部品は遺伝子を含むさまざまな化学物質で

あり、その動作機構は、複雑に絡み合った化学反応である。それぞれの化学反応は、ほんの数種類の物質による単純な反応 (素反応) の組み合わせである。ある反応による生成物が他の反応の基質 (生成物の原料となる物質) になったり、触媒になることで、化学反応どうしが制御しあうしくみにもなっている。この化学反応系のしくみが全て明らかになれば、細胞が活動するための機構、すなわち生命活動のしくみの解明が、飛躍的に進むであろう。

生体内反応系の構造を実験的な手法で解明するとき、反応系の一部を取り出し、さまざまな条件下のさまざまな入力に対して、その反応系がどう応答するかを調べ、系の構造を推定し、決定していく (システム同定)。つまり実験データを研究者が直接解析して系の構造を決めるわけだが、これは系を構成する要素 (反応にかかわる各化学物質の量、pH、温度、圧力など) の数が増えていくにつれて、困難になっていく。たとえば、全ての遺伝子が特定されている細菌や単細胞の真核生物の遺伝子の数は数千個であり、人間の場合は数万から 10 万個であろうと言われている。これらについて、その全体を一度に解析するのは非常に困難である [28][29]。

人為的に設計された工業的な系では、系に含まれれる要素 (またはそれを記述する状態変数) の値を任意の時刻で観測できることが多い。それらは可観測な変数と呼ばれる。また状態変数の個数も、取扱い可能な範囲に抑えることができるため、20 世紀後半の急速な計算機の発展にともなって、小規模な系に対しては、系の構造を知るため、または系の状態を制御するための多種多様な最適化法が開発されてきた [30][32]。しかしこれらは比較的構造が単純で、安定な系に対して開発されたものであり、全く事情が異なる自然界に存在する系では、有効でないものがほとんどである。生体系では、ほとんどの状態変数が可観測ではない。また物質の生成、分解が非常に速い反応も数多く含まれ、非常に不安定な系も少なくない。同じ理由から、可観測な状態変数の実測値には大きな誤差が含まれ、精密な測定が難しい。そのため、たとえば一つの細胞という系の中でも、構造が明らかになっているのは、そのほんの一部でしかない。また細胞は多くの系の集合体であり、それら小規模な系が相互作用しているものであるとも捉えられるが、それらの一つ一つの系についても同様である。

そういった生体内反応系は、生命活動を直接に担っていると言われており、その構造解明が強く望まれている。そしてそれは始まったばかりだが、その容易ではない。それは手がかりが少ないためである。さまざまな実験により得られるデータは、系を構成する要素 (状態変数) は何であるかというリストの一部、および特定の実験条件下での、一部の可観測である状態変数の経時変化 (時系列データ) のみであることが多い。状態変数の個数が少ない場合には、限られた実現可能な範囲の回数の実験でその構造を推測、決定することができる。またある程度なら状態変数が多くなっても、その系を複数に分割して、

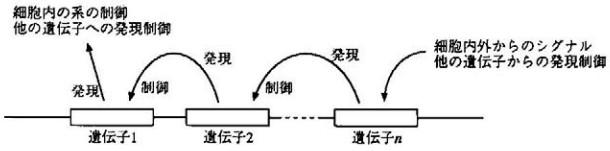

図 1.1: 遺伝子ネットワークの概念。細胞の活動を引き起こす遺伝子の発現は、他の遺伝子の発現、あるいは細胞内からのシグナルによってによって制御される。細胞外からのシグナルは、信号の伝達系を経て遺伝子に伝わり、遺伝子の発現、ひいては細胞の活動をひきおこし、また制御する。

逐次的に構造を推測、決定していく方法もある。ヒトゲノム計画の成果により、これからヒトの染色体にはどういった遺伝子が含まれているのかが明らかになろうとしているが、これらの遺伝子の発現が複雑に絡み合う相互的な制御構造 (遺伝子ネットワーク、図 1.1) では、状態変数 (各遺伝子の発現量) の数は数万から 10 万にのぼると言われている。こういった系のデータをすべて人間が直接吟味し、構造を推測していくのは不可能である。このため現在、計算機による自動的な構造推定の方法の開発が待たれている。

遺伝子ネットワークや生体内の代謝系などの生命活動の機構を、系に含まれる要素が互いに作用しあうネットワーク構造であると捉えたとき、系の時系列データからネットワーク構造を完全に決定しようとすることは、系が大規模であれば、上述のように事実上不可能であるが、ごく小さな、要素が数個しかない系でも不可能であることが多い。これは逆問題と呼ばれる問題だからである。逆問題とは、与えられた拘束条件からは、唯一の最適解を求めることができない問題である。ネットワーク構造を決定しようとする場合に、全く同じ、またはほぼ同じ時系列データを示すようなネットワーク構造は、一般に複数存在するため、それらは全て解であり、その中から唯一の最適解を選び出す、または解の個数を絞るためには、他の拘束条件を適用する必要がある。他の条件には、他の実験による時系列データや系に対する既知の情報などがあるが、生体内反応系の場合は、その構造は複雑であるが系を構成する要素同士の相互作用 (化学反応) はそれぞれ単純であり、ある物質はほかの数種類の物質としか直接には反応しないこと、特に酵素には基質特異性があり、それらが触媒するのは 1 種類またはごく少数の化学反応であること、また物質によっては既知の化学反応があることなどがあり、これらを利用して解を絞っていくことができる。

生体内反応系の構造解明のためには、実験で観測された時系列データと一致する挙動を示すネットワーク構造を決定するための最適化手法と、逆問題を解決するための手法、またはこれらの性質をあわせ持った手法が必要になる。ある条件下で実測された時系列データがあるときに、ネットワークを記述する数理モデルを考え、実験と同じ条件を計算機内で再現し、その下でモデルをシミュレートして時系列データを得る。これらの時系列デー

タが一致すれば、モデルの構造時計の構造が一致している可能性があり、これを検証するために、別の実験条件で観察されたデータに対して同様に、得られた全ての時系列データを再現するモデルを構築し、構造を推定していく。最適化手法とは、モデルから計算される時系列データが、実験で得られたデータと一致するように、モデルを構築、修正していくための計算機アルゴリズムである。しかしこの問題が逆問題であることから、こうして得られるモデルには複数の構造があり得るが、そのなかから唯一の解を決定する、あるいは唯一の解が求められるように最適化法を改良するのが、逆問題のための手法である。しかし必ずしも唯一の解を決定しなくても、有望な複数かつ少数のネットワーク構造を実験科学者に提示することができれば、それはネットワークの構造解明を大幅に促進する一助になると考えられる。また常に単一のアルゴリズムで構造を完全に決定してしまうと、系の性質によっては誤った構造を決定してしまうことが増えることも考えられ、判断の余地を残しておく方がむしろ有益である。つまり生体内反応系の最適化における逆問題のためのアプローチは、常に唯一の解を決定することではなく、最適化により得られる膨大な数の解を、分子機構的に見ても妥当で、また現実のネットワーク構造と一致する可能性が高い少数の解に絞るという戦略をとるべきである。本研究では以上の考えに従い、将来、生体内反応系や遺伝子ネットワークのシステム構造の解明を行うための生化学的な逆問題、すなわち状態変数の時系列データのみに基づいた系の構造推定のための最適化手法の開発を行った。

第2章　最適化と逆問題

2.1　最適化

一般に最適化問題とは、与えられた拘束条件を満たす多次元実数空間内の点の集合のうち、目的関数、または評価関数と呼ばれるの値を最小、あるいは最大にするもの (最適解) を求めよという問題 (多変数関数の最適化)、もしくは拘束条件を満たし目的関数を最小または最大化する関数を求めよ、という問題である [30]。ここで最小化と最大化は、目的関数の符号を変えることで同じ問題となるので、以下では最適化とは最小化のことであるとする。

多変数関数の最適化とは、生化学反応系を例に取ると、ある時点での状態変数値や、反応系の構造を決定することなどである。関数を求める問題では、ある状態変数に対する操作を決定することなどがある。どちらの場合も拘束条件として、濃度や圧力といった状態変数に対してはその値が正であること、また質量保存の法則により物質の量が制限を受けることなどがある。

生体内反応系の構造を推定しようとすることは、系内の要素間の相互作用の形式を推測することであり、これは多変数最適化である。多変数最適化は、目的関数の性質によって非常に簡単なものから事実上不可能なものまで様々であるが、その困難さをまず最初に分けるのは、目的関数の記述に最適化したい変数が陽に含まれている (陽関数) か否 (陰関数、関数の関数) かである。目的関数が陽関数の場合は、20 世紀後半、計算機の発展にともなって、解析的に最小化する手法が開発されてきた [30]。現在、目的関数が二次形式の場合には最小化できることが保証される様々な方法があり、そうでない場合にも極小値を求める方法が数多く開発されている。また導関数を用いることなどで最小値への収束を加速する方法もある [15][30][34]。しかし全ての関数に有効な最適化法は存在せず、特に陰関数や、陽関数でも記述が複雑で極値が多数あるもの、拘束条件が複雑なもの、非常に大きな系などは最適化が困難である。生体内反応系の構造最適化はほとんどの場合、陰関数の最適化であり、系は非常に大規模である。

2.1.1 解析的最適化

たとえば多変数関数を解析的に最適化するもっとも簡単な例の一つとして、以下のようなケースがある。ある与えられた容積Vを持つ、高さh、半径rの円筒形の容器を作る際に、表面積Sを最小とする$h=h^*$と$r=r^*$を求めよ、という問題である。この場合表面積は

$$S(h,r)=\pi r^2+2\pi rh \tag{2.1}$$

となる(底はあるが、ふたはないものとしている)。これが目的関数となる。この関数は二つの独立変数rとhを持つ多変数関数である。独立変数がある十分に小さな領域内にあり、その領域の境界上にない場合には、目的関数の偏導関数の値がすべて0になる独立変数値が、関数の極小点、極大点あるいは鞍点である。ここで$S(h,r)$の偏導関数は

$$\frac{\partial S(h,r)}{\partial h} = 2\pi r \tag{2.2}$$

$$\frac{\partial S(h,r)}{\partial r} = 2\pi(r+h) \tag{2.3}$$

となる。しかし$\frac{\partial S(h,r)}{\partial h}=0$となる点は$r=0$だけで、この時式(2.3)$h=0$になってしまう。これは最小値が境界上にあることを示しており、最適化のためには拘束条件を目的関数に組み込まなければならない。拘束条件は、rとhのどちらも物理的な長さであるから正の値を取ること($r\geq 0$、$h\geq 0$)、および容積が一定であることである。円筒の体積Vは$V=\pi r^2h$と表されるので式(2.1)のSは

$$S(r)=\pi r^2+\frac{2V}{r} \tag{2.4}$$

と変形され、独立変数はrのみになる。この問題は$S(r)$を最小にする$r=r^*$を求めよ、という問題になる。これであれば、$S(r)$はrの二次式でありr^2の項の係数は正なので、$S(r)$を微分して導関数を求め、それがゼロになるrの値を求めれば、それが$S(r)$を最小にするrの値、すなわち解r^*である(図2.1)。

$$\frac{\partial S(r^*)}{\partial r} = 2\pi r^* - \frac{2V}{(r^*)^2} = 0$$

$$r^* = \left(\frac{V}{\pi}\right)^{\frac{1}{3}}$$

$$h^* = \frac{V}{\pi(r^*)^2} = \left(\frac{V}{\pi}\right)^{\frac{1}{3}} = r^* \tag{2.5}$$

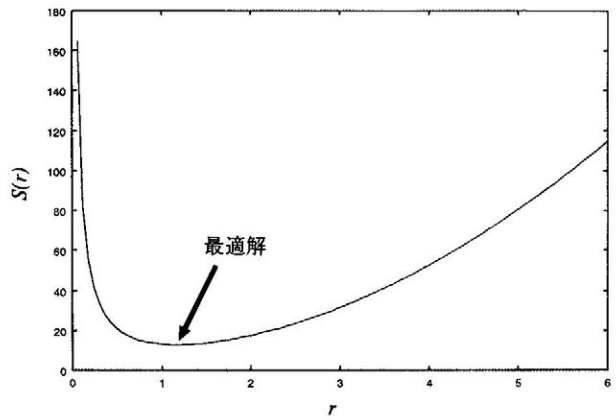

図 2.1: 例における目的関数 $S(r) = \pi r^2 + \frac{2V}{r}$ の、$V = 5$ としたときの形状。最適解は $r = (\frac{5}{\pi})^{(\frac{1}{3})} \approx 1.1675$ である。

目的関数の各独立変数に対して、目的関数の値が単調増加または単調減少であれば、最適化は簡単である。探索範囲の境界上を調べれば、それで最適化できる。また上述の例のように、極値を一つだけ持つ場合も同様である。極値が複数ある場合でも、導関数の値がゼロになる点が解析的に全て求められれば、その点が目的関数が極点あるいは鞍点であり、二階導関数の値から極値か鞍点か、極値なら極大か極小かが判別でき、それらの点を全て調べることで、最小点が得られる。拘束条件がある場合でも同様である [30]。

2.1.2 解析的探索法

しかし一般には、独立変数の数が増え、関数の記述が複雑になるにつれて、目的関数を直接解析して極小点を求めることがむずかしくなる。そういった場合には、逐次的な探索法を用いて、独立変数ベクトルと拘束条件が作る空間内を、様々な方向に極小点を探して進んでいく。これはまず、任意に探索の出発点となる独立変数ベクトルと探索方向ベクトルを決め、その方向に沿った直線上で目的関数値を極小にする点を求め、その点の目的関数値、または目的関数値と導関数値から次の探索方向を解析的に決定し、さらに直線探索を行っていくことを繰り返す。これにより全ての方向について目的関数が極小になる点を探していく方法である。解析的な探索法は二次関数を最小化することは保証しており、二次関数ではない関数についても、極小点の近傍では二次関数で近似できることから、探索の出発点を極小点に近いところに取れば、極小点は求められる。極小点が複数ある場合には、探索の出発点の位置により求まる点が決まる。また場合によっては極小点に収束し

ないこともある [30][34]。

単純かつ高速な方法には最急降下法、共役方向法などがある。これらは目的関数の導関数を用いて、初期あるいは前回の探索方向と直交あるいは共役な探索方向を決定し、その方向に沿って直線探索法を用いて目的関数を最小化する。前者は探索を繰り返すごとに最小値に漸近していき、後者は独立変数の個数が n 個 (探索空間の次元が n) であるときに、n 回の直線探索で最小値を得る [30][34]。

二次関数以外にも適用できる方法として、修正パウエル法 [12]、共役勾配法 (詳細は第 4 章参照)[34] などがある。前者は目的関数値のみを用い、初期探索方向ベクトルは任意の一時独立なベクトルとし、探索が進むに従って次第に共役なベクトルとしていく方法、後者は目的関数の偏導関数を用い、探索の開始から共役な方向に探索していく方法である。どちらも目的関数が従属変数の二次の関数であれば、最適解に到達し、そうでなければ、探索が進むにつれて探索点が極小値に次第に収束していく。[30][34]。

いずれの多次元探索法も、それぞれの探索法とは独立の直線探索法を必要とする。直線探索法とは、独立変数が一つの目的関数を最小化するアルゴリズムであり、これには勾配を利用するニュートン法、勾配を用いない黄金分割法、フィボナッチ探索、ローゼンブロック法 (Rosenbrock 法、成功と失敗のルーチンともよばれる) などがある [30]。勾配を用いない直線探索法は、目的関数の値を様々な点で計算しながら、最小値のある領域を絞っていくものである。したがって目的関数の形状を問わない。また極値を一つしか持たない目的関数の場合には最小化できる [30]。つまり、解析的な探索手法が有効であるかどうかは、目的関数に対して直線探索が有効であるかどうかが非常に大きく影響する。目的関数が非常に多くの極小値を持っている場合や、独立変数の定義域が入り組んでいる場合などは、一次元探索が難しいため、解析的な探索法は有効でないことが多い。

2.1.3 確率的探索法

探索空間内で解析的に方向を決め、逐次的に探索しても極小点が見つかりにくい場合、探索点を解析的にではなく、確率的に決定していく方法がある。これを確率的探索法という。

焼きなまし法

もっとも初期に開発された確率的探索法は、焼きなまし法 (シミュレーティッド・アニーリング) である [10]。これには現在様々な改良アルゴリズムがあるが、もっとも基本的なメトロポリスのアルゴリズムについて、概要を述べる。

このアルゴリズムの最初のステップでは、解析的方法と同様にまず探索の出発点を決め、目的関数値を計算する。そして次の探索点を出発点を平均とする正規分布乱数で決め、そこでの目的関数値が出発点よりも小さければ、探索点をそこに移し、次の探索点はその点を平均とした正規乱数を用いて生成する。そうでなければ、もとの出発点から新たに探索点を生成する。これは高い目的関数値を与える点(出発点)から、暗闇の中を手探りでより低い目的関数値を与える点を探し、徐々に坂を下りていく様子にたとえられる。しかしここで、目的関数値が小さくならなくても、熱力学でのボルツマンの確率分布による確率

$$P \approx \exp(-\frac{f}{kT}) \tag{2.6}$$

にしたがって(ここでfは目的関数値、kはボルツマン定数。Tは温度に相当する量だが、探索を制御するためのパラメータとなる)、そこに探索点を移動する。これはつまり、坂が登る可能性を残しておくことになる。これによって、出発点により決定される特定の極小値に捕らわれてしまう可能性を、低くすることができる[15]。また目的関数値fが大きなときは、よりランダムに探索を行い、最適化が進むにしたがって次第に坂を下る方だけを探索するようになる。つまり最初は、探索範囲全体を考慮した大域探索的な性格が強く、次第に探索点の周囲だけに注目する局所探索的になっていくことになる。この傾向を強くするため、パラメータTは探索の進行にともない値を小さくしていく。これが、灼熱した金属をゆっくりと冷やして最小のエネルギー状態に落ち着いていく様子と似ていることから、焼きなまし法と呼ばれる。Tを急激に小さくしていくと、すぐに局所探索になり、あまり目的関数値が小さくない極小値の周辺だけをいつまでも探索することになる可能性がある。反対にTを大きなままにしておくと、ランダムな点に対して目的関数の値を計算し、偶然にその値が小さな点が見つかるのを待つということになり、極小点への収束が非常に遅くなる。

探索点は極小点に次第に近づいていくが、極小点と一致する確率は低い。そのため、連続な目的関数に対する高精度な最適化はできない。焼きなまし法は開発当初、代表的な組み合わせ最適化問題である巡回セールスマン問題を実用的な範囲で解くことができる、ということで注目された[10][15]。この問題は、状態変数が離散値をとり、勾配を求めることができず、最適解を求めるためには全ての解候補(全ての独立変数ベクトル)に対して目的関数値を計算する(全点探索、または総当たり法)しかないため、解析的な最適化法は全く適用できない。特に巡回セールスマン問題は、系の規模が大きくなるにつれ爆発的に探索空間が大きくなる、NP完全問題と言われる種類の問題であり、系が大きくなるとすぐに実用的な時間内では最適解を求めることができなくなる。確率的探索法でも、全点を探索してしまうまでは最適解が得られる保証はないが、ほぼ最小と見なしてよい極小が、全点探索に比べてわずかな時間で得られ、また問題の規模の拡大にともなう探索時間

の増加も劇的に抑えられる。

さらに、探索方向はランダムに決定されるため、目的関数値が計算できさえすれば、ほかには何も必要ない。したがって解析的探索法に比べ、適用できる問題の範囲が非常に広い。これは以下に述べる遺伝的アルゴリズムでも同様である。

遺伝的アルゴリズム

焼きなまし法に次いで開発され、1980年代以降、急速に発展した最適化手法に、遺伝的アルゴリズム (Genetic Algorithm, 以下 GA) がある [2][3][5][6][31]。この方法も焼きなまし法と同様に、組み合わせ最適化に劇的な能力を発揮して有名になった方法である。しかし焼きなまし法とちがって、同時に多数の探索点を生成する。GAでは、各探索点を生物になぞらえ個体と呼ぶ。多数の探索点は集団または世代と呼ばれ、ある集団から次の集団を生成することを世代交代と呼ぶ。

初期世代の個体はそれぞれ、全くランダムに生成されるが、集団から次の世代の集団を作るには遺伝的操作と呼ばれる方法を用いる。これには交叉、突然変異、淘汰の3種類があり、一般的にはこれら全てを適用することで世代交代を行う。交叉では、二つの個体から解析的に一つ、あるいは二つの個体を生成する。これを繰り返して、次の世代に必要な数の個体を得る。新しい個体は、親となる二つの個体間の、内挿的な位置に生成することが多い。つまり補間による局所探索的な探索となる。しかしこれだけでは、確率的な性格が生じないため、個体(探索点、一般にはベクトル)の要素の一部を、ある確率(突然変異率)にしたがってランダムに変更する。これが突然変異である。これは用いる乱数の性質や突然変異率の大きさにより、大域探索的性格を持たせることができる。そして、こうして生成した集団の各個体に対して目的関数値を計算し、目的関数値から各個体を評価するための関数(評価関数)を使い、適応度を計算する。この適応度をもとに、新しく生成した個体集団ともとになった集団の個体とから、次の世代を構成する個体を選び出す。このとき、個体から計算した適応度が高い個体ほど、次世代に残る個体として選び出される確率を高くする。これが淘汰である。これにより、より目的関数値が小さな点の周囲に、個体を集めることになり、局所探索的になる [3][5]。

この3種類の操作により、大域探索、局所探索の両方を同時に行う。また同時多点探索であるため、解析的な探索法や焼きなまし法に比べると、最小点、あるいはよりよい極小点が求まりやすい。反面、生成する探索点の総数、つまり目的関数を計算する回数が非常に多いため、最適化により長い時間を要する。つまり最小値が求まる可能性と適用できる目的関数の幅広さを得る代わりに、探索速度が非常に遅くなる [3]。

2.2 逆問題とは

最適化問題における目的関数は、探索空間を形成する、独立変数の組であるベクトル(多次元空間内の点)の集合からスカラー値である目的関数値への写像である。目的関数値の集合に、ただ一つの最小点が含まれている場合は、様々な最適化手法で高速、あるいは高精度に最適化できるか、または最悪の場合、探索空間内の全ての点に対して、目的関数値を計算する総探索(総あたり法)により最適化できる。しかし目的関数によってこの最小点へ写される独立変数ベクトルが複数存在し、与えられた条件からは唯一の解を特定できないことがある。これを逆問題という。

逆問題は不良設定問題とも呼ばれる。これは、唯一の解を特定するための設定(拘束条件)が足りないことから、そう呼ばれる。図2.2に示すように、有限個のサンプリングポイントにおける状態変数の観測値が与えられ、それに一致する挙動を示す系のモデルを作成せよという最適化問題が与えられたとき、観測値に一致する挙動を示すモデルは複数存在し(最適解の集合。図2.2ではたとえば$f_1(t)$と$f_2(t)$の両方が最適解)、最適化により得られるモデルはそのうちの一部である。最適解の集合の要素を全て求めることは、多くの場合非常に困難である。集合が連続な領域である場合は、問題を解析的に解いてこれを求めることができなければ、前述した解析的、あるいは確率的最適化法でこれを求めることはできない。そのため最適解の集合を求めるためには、最適化のための条件(探索出発点など)を変えて複数回の最適化を行う、複数の種類の最適化法を適用するなどといった方法が必要である。

最適解の集合から唯一の解を選ぶための判断材料は、問題には含まれておらず、最適解の集合に含まれる要素の個数を減らす、あるいは唯一に特定するためには、拘束条件を問題に追加する必要がある。たとえば実験を複数回行って観測データを増やし、複数の時系列データの組に対して、その全てに一致するモデルの集合は、そのうちの一組のみに一致するモデルの集合よりも小さいはずである。またモデルの形式に制限を加える(その構造を制限する、モデルを記述するパラメータの値のとりうる範囲を制限するなど)ことによっても、最適解の集合を絞ることができる。

2.3 生化学反応系における逆問題

生化学反応系は、巨大なネットワークを形成している。ネットワークの各要素、またはそれを表す状態変数は、化学反応種の濃度や遺伝子の発現量、温度、pH、座標などの物理量であり、それらの相互作用の結果として、各要素の値が決まっていく。

様々な実験をすることにより、生化学反応系にはどのような状態変数が要素として含ま

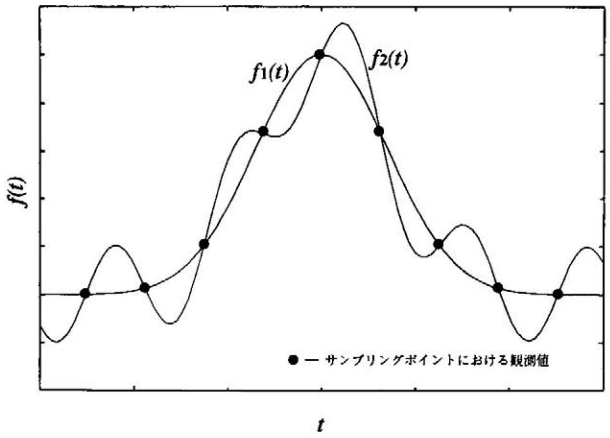

図 2.2: 逆問題の概念。グラフの横軸を時間 t とし、$t = t_i$ $(i = 1, 2, \ldots, n)$ の有限個のサンプリングポイント t_i における状態変数の観測値が与えられ、それと一致する応答を示す系を求めよという問題が与えられたとする。このとき最適化により $f_1(t)$ と $f_2(t)$ の二つの関数が求められ、どちらもサンプリングポイントにおける関数値は観測値と一致した場合、どちらを最適な関数とするかは、観測値との一致、不一致を見るだけでは決定できない。

れ、各状態変数は経時的にどのように変化していくことかを観察することができる。近年の技術革新により、観察できる対象は拡大しつつある。今まで巨大な生化学反応系ネットワークの未知の部分に対して、その挙動を観察することでそのシステム構造が部分的に推定されてきており、求められた構造から全体像を組み立てることも行われている。しかし生体内の反応系はまだ未知の部分が多い。

観察された挙動から反応系の構造を推測する、という問題は逆問題である。反応系の構造は、質量作用則などの数理モデルで表現されるが、前節に述べたように、サンプリングポイントにおいて状態変数が同じ値を示すが構造は異なるような数理モデルは、一般的に複数存在するため、唯一の数理モデルを決定することができないためである。これに対し、実験条件を変えて異なった挙動を観察し、複数の実験データを満足する (拘束条件の追加) ようにモデルを最適化することや、数理モデルを記述するパラメータの取りうる範囲を制限すること、またパラメータの個数を制限することでモデルの構造を制限するなどして、解の数を絞っていくことができる [24]。

代謝系などの反応系に対し、これまでは全て人手で実験、データ収集、構造特定が行われてきた。しかし近年、実験技術の大幅な進歩により、遺伝子ネットワークなどに対して膨大な実験データが自動的に得られるようになり、反応系の構造解析のために供給されるようになっている。構造解析は、得られた実験データと既知の反応系に対する知識から、

系の構造を推測し、それを検証するための実験を行うことを繰り返す、非常に時間と労力のかかる作業である。そこで、実験により観測されたデータから、自動的にネットワーク構造の推定をする計算機アルゴリズムの開発が望まれている。つまり実験で得られた様々な状態変数の時系列データから、それを再現するネットワークモデルの推定の自動化である [20][21][22][23][24][39]。

第3章　生化学反応系の最適化

代謝系などの生体内反応系や、遺伝子ネットワークなどの構造を明らかにしようとすることは、前述したように、実験で得られる系の挙動(系を構成する各要素、すなわち状態変数の値の時系列データ)と同じ挙動を示すように、系を表す数理モデルを最適化することである。これは逆問題であるが、ここではそれについては触れず、最適化法そのものについて述べる。

最適化を計算機よって行う場合、まず系を数理モデルで表現することが必要である。これは系を有限個の実数パラメータの集合で表すということである。モデルの記述はできるだけ簡潔で、パラメータの個数は少ない方がよい。最適化ではこのパラメータの値を(逐次的ではなく)同時に決定する必要があることが多く、その場合はパラメータの個数がそのまま探索空間の次元数となり、大規模な系を最適化しようとすると、パラメータ個数の増加にともない爆発的に探索空間の体積が大きくなる。生化学反応系一般では、質量作用則により連立微分方程式をたてることでモデリングが行われるが、これは反応系によって形式が全く異なってくるので、これを一般化し計算機で取り扱えるようにした一般質量作用則(Generalized Mass Action Law, GMA)[25]が考案されている。また遺伝子ネットワークでは、遺伝子が発現または抑制されているかどうかが重要であり、発現量そのものは現在では即的が困難であり、あまり重要視されていないことから、0か1の2値を取る要素からなる行列(2値論理ネットワーク)が用いられることが多い[8][9][23]。

また、系の数理モデルはシミュレートしやすいことも要求される。つまり、実験により系の挙動を観察したときと同じ条件を計算機内部で再現し、モデルが表現する系が現実に存在したらどのような挙動を示すかが推定しやすいということである。GMAの場合は微分方程式を解くことでモデルの挙動が計算により得られる。2値論理ネットワークでは、発現と抑制の依存関係が行列に表現されているので、逐次的に発現か否かを決定していける。

数理モデルの形式とシミュレート法が決まれば、最適化における目的関数が計算できることになる。つまり実験により得られている系の挙動と、モデルが示す挙動の相似性を反映する目的関数を設定すればよい。これには、実験データの各サンプリングポイントにおける相対二乗誤差の和などが用いられることが多い。さまざまな最適化法で、目的関数

値を最小、あるいは最大にするように、モデルを定義するパラメータ値を探していくことで、系の構造を探っていく。生体内反応系はほとんどの場合、系に含まれる要素数は複数で、場合によっては非常に多い。そのため探索空間は二次元以上の多次元空間である。また最適化すべきパラメータが、直接には目的関数を記述しないこともある。たとえば相対二乗誤差の和を用いる場合は、目的関数にはモデルをシミュレートすることで得られる系の状態変数の値と実験で得られた観測データがあるのみで、モデルを記述するパラメータは陽に現れないが、目的関数値に影響を及ぼす。こういった関数を汎関数と呼び、生体内反応系の構造を求めようとする最適化は、多変数の汎関数の最適化であると言える。

以下では、まず生化学反応系を表現するモデルの形式、次にそのシミュレート法、続いてモデルの最適化法について述べる。

3.1 系の数理的表現

計算機で自動的に反応系の構造を最適化するためには、まず反応系の数理モデルが必要である。つまり実数値および整数値の集合で反応系を表現する記述法である。化学反応一般では、連立微分方程式である質量作用則が用いられる。また遺伝子ネットワークでは、各遺伝子の発現または抑制が表現され、かつ簡潔な記述である2値論理ネットワーク(ブーリアンネットワーク)が用いられる。さらにここでは、簡潔さと十分な記述力を両立するS-システムと呼ばれるモデルについて述べる。

3.1.1 一般質量作用則

質量作用則(Mass Action Law)は、もっとも一般的に用いられる、化学反応系の数理モデルである。べき乗則と呼ばれる形式をもつ連立微分方程式で、独立変数は時間、従属変数は系の状態変数で、多くの場合、反応に関与する物質の濃度である。もっとも正確なモデルを作ることができ、酵素反応などの特殊な形式の反応速度式は、質量作用則による数理モデルを近似することで得られる[17]。さまざまな反応速度論のもっとも基礎となるモデルである。

例えばA、B、Cの3種類の化学物質があるときに、これらが

$$\mathrm{A} + 2\mathrm{B} \xrightarrow{k} \mathrm{C}$$

という、反応速度定数がkの化学反応に関与しているとする場合、この反応を表す質量作

用則に基づく数理モデルは以下のようになる。

$$\frac{d[\mathrm{C}]}{dt} = k[\mathrm{A}][\mathrm{B}]^2 \tag{3.1}$$

ここで[A]は物質Aの濃度を表す。つまり生成物Cの増加速度は基質A、Bの濃度の指数乗の積となるということである。この場合、つまり生成物Cの増加速度と基質Aの減少速度は等しく、基質Bの減少速度はその2倍である。

$$\begin{aligned} \frac{d[\mathrm{A}]}{dt} &= -\frac{d[\mathrm{C}]}{dt} \leq 0 \\ \frac{d[\mathrm{B}]}{dt} &= -\frac{1}{2}\frac{d[\mathrm{C}]}{dt} \leq 0 \end{aligned} \tag{3.2}$$

が成り立つ。

生成物Cが複数の反応により生成される場合には、式(3.1)の右辺に複数の項が現れる。つまり反応系が

$$\begin{aligned} \mathrm{A} + 2\mathrm{B} &\xrightarrow{k_1} \mathrm{C} \\ 3\mathrm{D} + \mathrm{E} &\xrightarrow{k_2} \mathrm{C} \end{aligned}$$

という反応系であれば、

$$\frac{d[\mathrm{C}]}{dt} = k_1[\mathrm{A}][\mathrm{B}]^2 + k_2[\mathrm{D}]^3[\mathrm{E}] \tag{3.3}$$

というモデルになる。Cを基質として消費する反応があれば、その反応によるCの減少速度を減じる項が現れることになる。たとえば

$$\begin{aligned} \mathrm{A} + 2\mathrm{B} &\xrightarrow{k_1} \mathrm{C} \\ 3\mathrm{C} + \mathrm{D} &\xrightarrow{k_2} \mathrm{E} \end{aligned}$$

という反応系であれば、

$$\begin{aligned} \frac{d[\mathrm{C}]}{dt} &= k_1[\mathrm{A}][\mathrm{B}]^2 - \frac{d[\mathrm{E}]}{dt} \\ &= k_1[\mathrm{A}][\mathrm{B}]^2 - \frac{1}{3}k_2[\mathrm{C}]^3[\mathrm{D}] \end{aligned} \tag{3.4}$$

このように質量作用則による数理モデルは、各状態変数についてその変化速度を記述する微分方程式であり、系に含まれる各状態変数ついての微分方程式を連立させることで、その系の数理モデルとなる。上記の例からわかるように、質量作用則によるモデルでは、

反応形式が異なれば微分方程式の形式が異なる。こういった数理モデルは、データ構造としてツリー構造やリスト構造などを利用することで計算機で取り扱うことが可能ではあるが、実験で得られた時系列データに基づく最適化を考える場合、ツリーまたはリストで表されたモデルに対して、その構造と要素を同時に決定しなければならず、非常に困難な問題であり、現在その研究は始められたばかりである [36]。

したがって現在、計算機で取り扱う数理モデルは、どんな反応系でも同じ形式で記述できるような形式であることが望ましい。そこで質量作用則による微分方程式を一般化した、一般質量作用則 (Generalized Mass Action Law) が考え出された [25]。これは以下の形式で表される。

$$\frac{dX_i}{dt} = \sum_{k=1}^{A} \alpha_{ik} \prod_{j=1}^{n} X_{jk}^{g_{ijk}} - \sum_{k=1}^{B} \beta_{ik} \prod_{j=1}^{n} X_{jk}^{h_{ijk}} \tag{3.5}$$

ここに X_i は状態変数であり、化学物質の濃度や温度などの物理量を表す。n はその系内での総数である。A は X_i を増加させようとする作用 (合成反応など) の数、α_k はその速度定数、B は X_i を減少させようとする作用 (分解反応など) の数、β_k はその速度定数である。g_{ijk} は一般的な意味としては、X_i を増加させる反応 k において状態変数 X_j が X_i に及ぼす影響、h_{ijk} の意味は同様に、X_i を減少させる反応 k において状態変数 X_j が X_i に及ぼす影響である。$g_{ijk} > 0$ は、状態変数 X_j は反応 k において、X_i の生成を促進する作用を持っていることを表す。つまり X_i の生成反応の基質であったり、触媒であるようなことを表す。逆に $g_{ijk} < 0$ であれば、X_j は反応 k において、X_i の生成を阻害する働きを持つ。$g_{ijk} = 0$ であれば、X_j は X_i に直接には作用しない。h_{ijk} は X_i の分解について、同様の意味を持つ。最適化すべき系について、A、B、n を定めておけば、モデルの最適化とはモデルを記述するパラメータ α_{ik}、β_{ik}、g_{ijk}、h_{ijk} の値を決定するという問題になる。ある状態変数 X_i について、α_{ik} と β_{ik} の総数はそれぞれ A と B、g_{ijk} と h_{ijk} の数はそれぞれ An と Bn である。したがって系を記述するパラメータの総数は $(A+B)n(n+1)$ であり、このパラメータ全ての値を決定することで、系の数理モデルを定義することができる。つまり、数理モデルの決定は最適化法による $(A+B)n(n+1)$ 次元実数ベクトルの決定であると考えることができ、これが探索空間の次元数になる。状態変数の個数 n は、系にどんな物質が含まれているか、または可観測な状態変数の個数などの既知の情報により見積もることができるが、未知の系に対しては、その内部で行われている化学反応を限定することができないため、A および B については見積もりが難しいことがある。これを大規模な系 (状態変数の個数 n が多い系) に対しても、個々の物質が関与する反応の数はある少数の範囲内であるとすると、パラメータ数のオーダーは $O(n^2)$ であるが、反応数がおおよそ n に比例して増えていくとすると、パラメータ数のオーダーは $O(n^3)$ となり、大規模な系に対する最適化が急激に困難になると言える。

3.1.2 2値論理ネットワーク

2値論理ネットワークは、状態変数が0または1のいずれかの値しか取らないとする、グラフ理論に基づくモデルである[8][9]。このモデルはグラフにより表現され、グラフ中の各ノードはそれぞれ一つあるいは複数の要素からなるグループであり、これは多階層有向グラフと呼ばれる。状態変数が離散値であると同時に、モデルにおける独立変数である時間も離散値であり、タイムステップと呼ばれる。各タイムステップはその前後関係を表すだけで、実時間を反映しない。状態変数が連続量をとらないため、化学反応一般を表すことはできないが、遺伝子ネットワークを表現するモデルとしてもっともよく用いられる。

遺伝子ネットワークでは、DNAマイクロアレイ(ジーンチップ、Gene chip)と呼ばれる実験技術により要素数が数万の単位の実験データが得られる。これは対象とする各遺伝子について、その発現の有無を知ることができる。発現量を知ることはできず、またサンプリング時には、系(多くの場合は細胞)を破壊することで反応の進行を止めてから測定するため、サンプリング時刻を高精度で特定することも難しい。

遺伝子に関しては、セントラルドグマと呼ばれる法則がある。遺伝子であるDNAの情報はRNAに伝えられ、それをもとにタンパク質が作られる、つまり遺伝情報の流れる方向はDNA$\longrightarrow$RNA$\longrightarrow$タンパク質という向きであるという法則である。つまり遺伝子からmRNA、mRNAからタンパク質が作られ、ある遺伝子からその遺伝子が表すタンパク質が作られたとき、その遺伝子は発現した、という。この遺伝子の発現はさまざまなタンパク質によりその発現するタイミングや発現量が制御されているが、その制御しているタンパク質も、そのタンパク質の遺伝子の発現により作られたものであり、つまり遺伝子の発現は他の遺伝子により制御されているということになる。この制御の様子は代謝系の反応などと同様に、カスケードやループを作っており、全体として制御ネットワークを形成していると考えられている[1]。そしてジーンチップにより、この制御により発現が促進されているか抑制されているかを、各遺伝子ごとに調べることができる。ジーンチップによる実験結果から、2値論理ネットワークによる数理モデルを得るには、以下のような手順による[8][9]。

実験結果と遺伝子間の関係

解析対象となる遺伝子ネットワークを形成する、N個の遺伝子を集合$S = a, b, c, \ldots$と表し、遺伝子間の制御関係を(a, b)と表す。遺伝子aが遺伝子bの発現を直接に制御していれば、そこには関係(a, b)が存在する、とする。

対象としている遺伝子ネットワークについて、何も手を加えない状態(ワイルドタイプ)

で各遺伝子の発現の有無を調べる。次に、この中のある遺伝子を強制的に発現させるか、またはその発現を抑えた状態で S 中の全ての遺伝子の発現の有無を同時に観測する。発現を強制的に調節された遺伝子を a とするとき、もし他の遺伝子 b についてワイルドタイプの場合と発現の有無が変化していたら、それは遺伝子 a が遺伝子 b の発現を直接、または間接に制御を行っていると考えられ、遺伝子間の関係を表す行列 R^0 の (a,b) 成分 $R^0_{a,b}$ を 1 とする。この強制発現あるいは抑制の実験を全ての遺伝子に対して行い、関係が認められなかった R^0 の成分は 0 とすることで、行列 R^0 を完成する。

このとき、もし $R^0_{a,b} = R^0_{b,a} = 1$ であれば、どちらの遺伝子がどちらの遺伝子に影響をあたえているのか、影響は双方向か単方向かは分からない。こういった関係にある遺伝子は、同値類としてまとめられ、一つの遺伝子グループとして捉えられる。そしてこの方法によるアプローチでは、全ての同値類を特定し、同値類間の影響の方向を調べることで、遺伝子ネットワークの構造を特定することを考える。

R^0 から、以下の手順により R^* を得る。

$$R^* = \bigcup_{n=0}^{\infty} R^n \begin{cases} R^0(i,j) = \begin{cases} 1 & (i=j) \\ 0 & (i \neq j) \end{cases} \\ R^{n+1} = R^n \circ R \end{cases} \tag{3.6}$$

ここに用いた演算子 $\circ$ は、以下のように定義されるものである。

$$(G \circ F)(i,j) = \min\left(1, \sum_k G(i,j)F(i,j)\right)$$

式 (3.6) で示す R^* の (a,b) 成分 $R^*_{a,b}$ は、遺伝子 a から R^0 の成分をたどっていって、遺伝子 b に最終的に影響があるかどうか、を示す。ここで $R^*_{a,b} = R^*_{b,a}$ であれば、遺伝子 a と遺伝子 b は制御ループにあることを示す。

次に、同値関係を調べるために行列 ER を計算する。これは以下のように定義される。

$$\begin{aligned} ER(a,b) &= \begin{cases} 1 & (R^*(a,b) = 1 \wedge R^*(b,a) = 1) \\ 0 & (R^*(a,b) \neq 1 \vee R^*(b,a) \neq 1) \end{cases} \\ \{a\}_{ER} &= \{b | ER(a,b) = 1\} \end{aligned} \tag{3.7}$$

ここで $\wedge$ は論理積、$\vee$ は論理和を表す。遺伝子 a から遺伝子 b に影響があり、同時に遺伝子 b から遺伝子 a にも影響がある場合に $ER_{a,b}$ が 1 になり、そうでない場合は $ER_{a,b} = 0$ となる。ER は対称行列である。ここで遺伝子 a に対して、ER の遺伝子 a を表す行において $ER_{a,b} = 1$ となる b の集合が、$\{a\}_{ER}$ であり、a を含む同値類に含まれる遺伝子の集合である。同じ同値類内の遺伝子間では、制御の上下関係は決定できない。したがって以下では、同値類間の制御関係を得るための演算となる。

ここで、次のような関係CRを定義する。

$$CR = \{(\{a\}_{ER}, \{b\}_{ER}) | xR^*y, \exists x \in \{a\}_{ER}, \exists y \in \{b\}_{ER}\} \tag{3.8}$$

ここで$\exists x \in \{a\}_{ER}$は、xが$\{a\}_{ER}$に含まれていることを表す。これはaからbへの制御関係の有無を表す。つまり、二つの同値類に属する遺伝子間に直接、または間接の制御関係があることを表す。R^*は対称行列とは限らないので、CRも対称行列とは限らない。そしてこの制御関係CRから、同値類間の直接の制御関係だけを取り出したものが、ネットワーク構造を表す。これを集合MRとし、以下のように定義する。

$$MR = \{(a,b) | a \neq b, (a,b) \in CR, \neg(\exists x \in S,\ (a,x) \in CR,\ (x,b) \in CR,\ x \neq a,\ x \neq b)\} \tag{3.9}$$

これにより定義される関係MRに(a,b)が存在すれば、遺伝子aを含む同値類が遺伝子bを含む同値類に影響を持つことを意味する。影響を持つとは、発現を促進、または抑制する制御関係を持つことである。

3.1.3 S-システム

上述した二種類のネットワークの数理モデルには、それぞれ問題点がある。一般質量作用則による数理モデルは、モデルを定義するパラメータ数が多く、また最小限必要なパラメータ数を正確に決定することができないことであり、また2値論理ネットワークでは、連続値を扱えないため、適用対象が非常に限られることである。そこで、パラメータ数を抑え、かつ連続量を扱うことのできるS-システム[17]と呼ばれる数理モデルを導入する。これは以下の形式で表される連立微分方程式である。

$$\frac{dX_i}{dt} = \alpha_i \prod_{j=1}^{n} X_j^{g_{ij}} - \beta_i \prod_{j=1}^{n} X_j^{h_{ij}} \tag{3.10}$$

各添字の意味は式(3.5)と同じで、X_iは物質の濃度や温度、圧力などの状態変数、nは状態変数の総数、g_{ij}はX_jがX_iを増加しようとする影響の大きさ、h_{ij}はX_jがX_iを減少しようとする影響の大きさを表す。α_i、β_iはそれぞれ、X_iを増加あるいは減少しようとする作用の速度定数である。これらは0または正の値を持つ。g_{ij}およびh_{ij}は正、負、または0の実数である。$g_{ij} > 0$の場合、X_jはX_iの増加を促進する働きを持ち、$g_{ij} < 0$の場合はX_iの増加を抑える働きがある。これはX_iを減少する働きとは区別され、これは$h_{ij} > 0$で表される。同様に$h_{ij} < 0$はX_iの減少を抑制する働きの大きさを表す。$g_{ij} = 0$であることはX_jはX_iの増加には影響を持たないことを表す。$h_{ij} = 0$であることはX_j

表 3.1: 図 3.1 に例示する遺伝子ネットワークを表現する S-システムパラメータ。このように、実数パラメータの値を設定することで数理モデルを定義することができる。図 3.1 中の $pool_1$ および $pool_4$ は一定値であり、X_1 と X_4 の生成プロセスが一定の速度で行われていることを表す。これらはそれぞれ α_1 および α_4 で表されている。これらのプロセスに対し、X_3 と X_5 がそれぞれ抑制、促進の作用を及ぼす。

i	α_i	g_{i1}	g_{i2}	g_{i3}	g_{i4}	g_{i5}	β_i	h_{i1}	h_{i2}	h_{i3}	h_{i4}	h_{i5}
1	15.0	0.0	0.0	1.0	0.0	-0.1	10.0	2.0	0.0	0.0	0.0	0.0
2	10.0	2.0	0.0	0.0	0.0	0.0	10.0	0.0	2.0	0.0	0.0	0.0
3	10.0	0.0	-0.1	0.0	0.0	0.0	10.0	0.0	-0.1	2.0	0.0	0.0
4	8.0	0.0	0.0	2.0	0.0	-1.0	10.0	0.0	0.0	0.0	2.0	0.0
5	10.0	0.0	0.0	0.0	2.0	0.0	10.0	0.0	0.0	0.0	0.0	2.0

は X_i の減少に影響を持たないことを表す。$g_{ij} = h_{ij} = 0$ であれば、X_j は X_i の合成、分解過程に直接には影響を与えないことになる。

一般質量作用則による数理モデル (式 (3.5)) では、状態変数 X_i を増加する作用 (化学反応による生成など) が A、減少する作用 (分解など) が B あるとするが、生体内反応系において、個々の物質は、直接には数個のごく少数の物質との反応に関与しているだけであることが多い。つまり式 (3.5) における A および B は 2〜3 であるようなことが多いということである。すると状態変数 X_i に対しそれを増加させようとする反応が数個、減少しようとする反応が数個で、X_i の値のとりうる範囲を限定すると、これらの反応をそれぞれ一つにまとめて近似することが考えられる。S-システムはこうした考えから考案された [17][32]。

この近似により S-システムでは、パラメータ総数は、要素数を n とするとき、$2n(n+1)$ となり、式 (3.5) に比べると $1/(A+B)$ とすることができる。また A と B の値を見積もることを省くことができるため、n さえ決定しておけばモデルの形式は完全に固定され、モデルの最適化は $2n(n+1)$ 個の実数値を要素の持つベクトルの探索となる。このため、一般作用則に比べてモデル作成、定義の作業および最適化時間を大幅に短縮することができ、また動特性の解析などがしやすくなる [32]。

図 3.1 に示す遺伝子ネットワークを S-システムで表現した例を表 3.1 に示す。この遺伝子ネットワークが実数パラメータの集合のみで定義される。このパラメータを式 (3.10) に代入することにより得られる微分方程式を解くことで、この系の時系列データ (図 3.2) を得ることができる [24]。

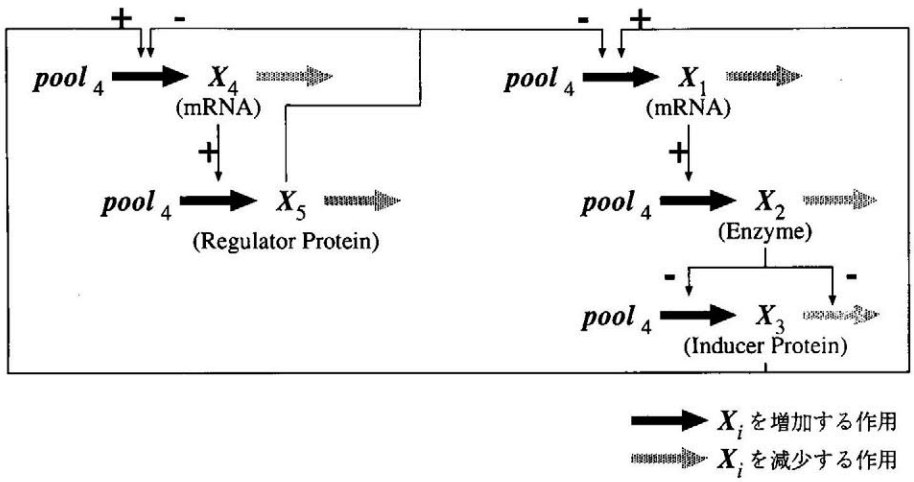

図 3.1: 遺伝子ネットワークの例。ここでは二つの遺伝子から X_1 と X_4 の二つの mRNA が作られ、そこからそれぞれタンパク質 X_2、X_5 が作られる。X_1 と X_4 の生成される速度は $pool_1$ および $pool_4$ で表される。これらは一定の値であり、他から制御されなければ、X_1 と X_4 は一定速度で生成されることを表す。X_2 は別のタンパク質 X_3 の生成、分解をともに抑制し、X_5 は二つの遺伝子の発現を抑制する。タンパク質 X_3 は二つの遺伝子の発現をともに促す。これを表す S-システムモデルを定義するパラメータは表 3.1 で表される。

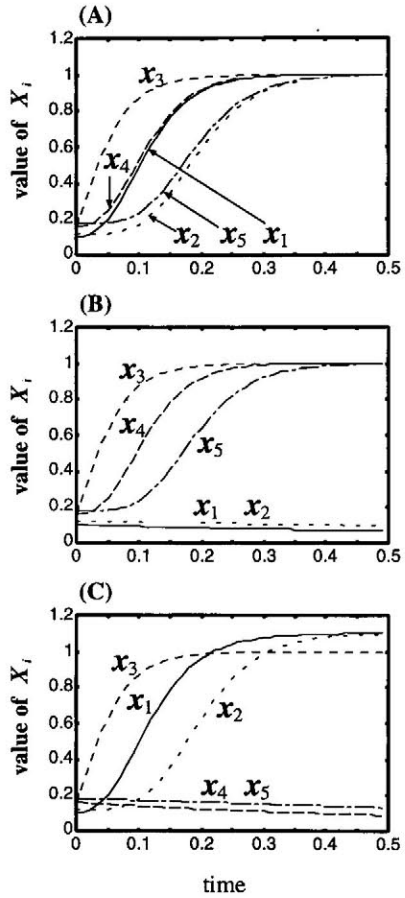

図 3.2: 図 3.1 示す遺伝子ネットワークを表現する S-システムモデル (表 3.1) から計算される時系列データ。(A) 表 3.1 に示すパラメータを式 3.10 に代入し、微分方程式を解くことで得られる時系列データ。(B) 表 3.1 に示すパラメータのうち、α_1(すなわち $pool_1$) を 0.0 に設定して計算される時系列データ。これは図 3.1 示す遺伝子ネットワークに対して、タンパク質 X_2 の遺伝子を破壊するなどして、その発現を強制的に抑制して、系の挙動を観察する実験に相当する。(C) 表 3.1 に示すパラメータのうち、$\alpha_4(pool_4)$ を 0.0 に設定して計算される時系列データ。(B) と同様に、これはタンパク質 X_2 の遺伝子の発現を強制的に抑制する実験に相当する。

3.2 微分方程式の数値解法

S-システムで表される数理モデルを使ってシミュレーションを行うことは、微分方程式を積分し、与えられる条件の元で状態変数の時系列データを得ることである。積分を行い原関数の値を求めることを一般に、微分方程式を解くとも言うが、この微分方程式の解法には解析的なものと、数値的なものがある。解析的な方法とは、積分により原関数を求めることである。これが理想ではあるが、上述の一般質量作用則もS-システムも、解析的に解くことはできない。したがって数値計算により、解こうとする独立変数の範囲全体にわたって、ある点を初期値として原関数の近似値を逐次的に求めていくことになる[15][33]。現在までに様々な数値解法が開発されているが、ここではもっとも基本的な解法であるオイラー法、もっとも代表的で広く用いられているルンゲ-クッタ法、S-システムに特化したアーヴィン (Irvine) とサバジョー (Savageau) の方法[7]について述べる。

3.2.1 オイラー法

ここで解きたい微分方程式は、一般質量作用則もS-システムも、いずれも一階の連立微分方程式である。形式的には

$$\begin{aligned} \mathbf{y}' &= \mathbf{f}(\mathbf{x},\mathbf{y}) \\ \mathbf{y}(\mathbf{a}) &= \mathbf{y}_0 \end{aligned} \tag{3.11}$$

と書ける。$\mathbf{y}_0$ は $\mathbf{x}=\mathbf{a}$ での $\mathbf{y}(\mathbf{x})$ の値であり、数値解法の初期値となる。原関数 $\mathbf{y}(\mathbf{x})$ は解析的に求められる場合もあるが、そうでない場合もある。また解そのものが存在する場合と存在しない場合がある。ここでは解が存在するものとし、$\mathbf{f}(\mathbf{x},\mathbf{y})$ は十分に微分可能であるとする。すると解 $\mathbf{y}(\mathbf{x})$ は二次の項までのテイラー展開により

$$\mathbf{y}(\mathbf{x}+\mathbf{h}) = \mathbf{y}(\mathbf{x}) + h\mathbf{y}'(\mathbf{x}) + \frac{h^2}{2}\mathbf{y}''(\mathbf{x}+\theta\mathbf{h}) \tag{3.12}$$

と書ける。すなわち

$$\frac{\mathbf{y}(\mathbf{x}+\mathbf{h})-\mathbf{y}(\mathbf{x})}{h} = \mathbf{y}'(\mathbf{x}) + \frac{h}{2}\mathbf{y}''(\mathbf{x}+\theta\mathbf{h}) \tag{3.13}$$

となる。微分方程式(式(3.11))を満足する解を $\mathbf{y}$ とするとき、ここでの目的は、$\mathbf{x}=\tilde{\mathbf{x}}$ での原関数値 $\mathbf{y}(\tilde{\mathbf{x}})$ を求めることである。そこで解を計算する範囲 $[a,\tilde{\mathbf{x}}]$ を N 等分し、

$$\begin{aligned} h &= \frac{\tilde{\mathbf{x}}-\mathbf{a}}{N} \\ \mathbf{x}_0 &= \mathbf{a}, \\ \mathbf{x}_n &= \mathbf{a}+n\mathbf{h} \end{aligned}$$

とする。$\mathbf{y}(\tilde{\mathbf{x}})$ を式 (3.11) の解とすると、

$$\begin{aligned} \mathbf{y}(\mathbf{x}_0) &= \mathbf{y}_0, \\ \mathbf{y}'(\mathbf{x}_0) &= \mathbf{f}(\mathbf{x}_0, \mathbf{y}_0) \end{aligned} \tag{3.14}$$

これをテイラー展開して

$$\begin{aligned} \mathbf{y}(\mathbf{x}) &= \mathbf{y}_0 + (\mathbf{x} - \mathbf{x}_0)r\mathbf{y}'(\mathbf{x}_0) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^2\mathbf{y}''(\xi_0) \\ &= \mathbf{y}_0 + \mathbf{f}(\mathbf{x}_0, \mathbf{y}_0) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^2\mathbf{y}''(\xi_0), (\mathbf{x}_0 < \xi_0 < \mathbf{x}) \end{aligned} \tag{3.15}$$

ここで $\mathbf{x} = \mathbf{x}_1 = \mathbf{x}_0 + \mathbf{h}$ とし、$\mathbf{h}^2$ の項は値が無視できるほど小さいと見なして省略し、$\mathbf{Y}_0 = \mathbf{y}_0$ とおくと、$\mathbf{y}(\mathbf{x}_1)$ の近似値として

$$\mathbf{Y}_1 = \mathbf{Y}_0 + \mathbf{h}\mathbf{f}(\mathbf{x}_0, \mathbf{Y}_0) \tag{3.16}$$

が得られる。この $\mathbf{Y}$ は $\mathbf{y}$ の数値解法により得られる近似値となる。解の真の値は、以下の式 (3.17)

$$\mathbf{y}(\mathbf{x}_2) = \mathbf{y}(\mathbf{x}_1) + \mathbf{h}\mathbf{y}'(\mathbf{x}_1) + \frac{1}{2}\mathbf{h}^2\mathbf{y}''(\xi_1) \tag{3.17}$$

で計算されるが、これを式 (3.16) で得られる $\mathbf{Y}_1$ を用いて計算した値

$$\mathbf{Y}_2 = \mathbf{h}\mathbf{f}(\mathbf{x}_1, \mathbf{Y}_1) + \mathbf{Y}_1 \tag{3.18}$$

で近似する。この方法はもっとも基礎的な数値解法で、オイラー法と呼ばれる。ここで $\mathbf{h}$ を刻み幅という。

オイラー法では、各ステップでの誤差 $\frac{1}{2}\mathbf{h}^2\mathbf{y}''(\xi_1)$ が蓄積して、真の解と近似解が大きく異なってしまう場合がある。これを修正しながら計算を勧めていく方法がルンゲ-クッタ法である。

3.2.2 ルンゲ-クッタ法

オイラー法に真の解を代入したときの誤差

$$\tau_{n+1} = \frac{\mathbf{y}(\mathbf{x}_{n+1}) - \mathbf{y}(\mathbf{x}_n)}{h} - \mathbf{f}(\mathbf{x}_n, \mathbf{y}(\mathbf{x}_n)) \tag{3.19}$$

を、オイラー法における打ち切り誤差と呼ぶ。これは一ステップ分の計算による誤差なので、局所打ち切り誤差とも呼ぶ。これは、$\mathbf{Y_n}$が真の値$\mathbf{y}(\mathbf{x}_n)$に等しかった場合の、Y_{n+1}とy_{n+1}の差を表す。オイラー法の場合は

$$\tau_{n+1} = \frac{1}{2}h\mathbf{y}''(\xi) = O(h) \qquad (\mathbf{x}_n < \xi < \mathbf{x}_{n+1}) \tag{3.20}$$

となり、真の解について

$$\mathbf{y}(\mathbf{x}_{n+1}) = \mathbf{y}(\mathbf{x}_n) + \mathbf{h}\mathbf{f}(\mathbf{x}_n, \mathbf{y}(\mathbf{x}_n)) + h\tau_{n+1} \tag{3.21}$$

と書け、$\tau_{n+1} = O(h)$となる。このとき、オイラー法は精度1であると言う。一般に、

$$\mathbf{Y}_{n+1} = \mathbf{Y}_n + h\mathbf{\Phi}(\mathbf{x}_n, (\mathbf{Y}_n)) \tag{3.22}$$

という計算を考えたときに、これに真の解を代入すると

$$\mathbf{y}(\mathbf{x}_{n+1}) = \mathbf{y}(\mathbf{x}_n) + \mathbf{h}\mathbf{\Phi}(\mathbf{x}_n, \mathbf{y}(\mathbf{x}_n)) + O(h^{p+1}) \tag{3.23}$$

となるとき、この計算法は精度pである、という。hは1以下の値に取ることが多いので、精度pが高ければ、それだけ局所打ち切り誤差も小さいことになる。ルンゲ-クッタ法は、精度4の方法である。

$\mathbf{Y}_n$をn回目の計算ステップで得られた原関数の近似値($n=0$での値は初期値として与えられる)、$\mathbf{x}_n$をその時の独立変数の値とすると、ルンゲ-クッタ法は

$$\begin{aligned}
\mathbf{k_1} &= \mathbf{f}(\mathbf{x}, \mathbf{Y}(\mathbf{x})) \\
\mathbf{k_2} &= \mathbf{f}(\mathbf{x} + \frac{h}{2}, \mathbf{Y}(\mathbf{x}) + \frac{h}{2}\mathbf{k}_1) \\
\mathbf{k_3} &= \mathbf{f}(\mathbf{x} + \frac{h}{2}, \mathbf{Y}(\mathbf{x})) + \frac{h}{2}, \mathbf{Y}(\mathbf{x}) + \frac{h}{2}\mathbf{k}_2) \\
\mathbf{k_4} &= \mathbf{f}(\mathbf{x}, \mathbf{Y}(\mathbf{x}) + h\mathbf{k_3})
\end{aligned} \tag{3.24}$$

(3.25)

とし、

$$\mathbf{Y}_{n+1} = \mathbf{Y}_n + \frac{h}{6}(\mathbf{k_1} + 2\mathbf{k_2} + 2\mathbf{k_3} + \mathbf{k_4}) \tag{3.26}$$

として計算する。式(3.22)において、$\mathbf{\Phi}(\mathbf{x}_n, \mathbf{y}(\mathbf{x}_n)) = \frac{h}{6}(\mathbf{k_1} + 2\mathbf{k_2} + 2\mathbf{k_3} + \mathbf{k_4})$となっている。

オイラー法とルンゲ-クッタ法は$\mathbf{Y}_{n+1}$を得るために、その1ステップ前での値$\mathbf{Y}_n$と$\mathbf{x}_n$だけしか用いない、一段階法と呼ばれる方法に属する。一段階法では、刻み幅hを小さく

していくと真の解に収束していくこと、また精度が高い方法ほどその収束が早いことが証明されている。ルンゲ-クッタ法は、複数ステップ前からの計算結果を用いる多段解法に比べ、初期値を求める手間が少ないこと、単純なアルゴリズムでありながら精度が高いことから、幅広く用いられている。

3.2.3 アーヴィンとサバジョーの方法

上述したS-システムは、連立微分方程式である。各式は二つの項を含んでおり、それぞれべき乗数の積になっている。

$$\frac{dx_i}{dt} = \alpha_i \prod_{j=1}^{n} X_j^{g_{ij}} - \beta_i \prod_{j=1}^{n} X_j^{h_{ij}} \quad (i = 1, 2, ..., n) \tag{3.27}$$

このため微分係数 $\frac{dX_i}{dt}$ を計算する途中で非常に大きな値がたびたび現れ、これが桁落ちを起こすことなどで計算精度に影響を与え、またオーバーフローなどの原因にもなる。また計算機での数値計算では、べき乗の計算は、四則演算に比べると5倍から10倍の時間がかかる。そこで、式(3.27)の両辺の対数を取ることで得られる微分方程式を解き、得られた数値解を、指数乗を計算して原関数の値を得る、という方法がアーヴィンとサバジョーによって開発された[7]。

まず、時刻 t における式(3.27)の左辺の値を $X_i^{(1)}$ と表し、原関数値 X_i の対数を $Y_i(t)$ とすると、これはテイラー展開を用いて

$$Y_i(t+h) = Y_i(t) + \sum_{m=1}^{p} \left(\frac{Y_i^{(m)}(t)}{m!} \right) h^m \quad (i = 1, 2, ..., n) \tag{3.28}$$

となる。ここではテイラー展開の高次の項($p+1$ 次以上の項)を切り捨てている。この式は以下のように変形できる。

$$\begin{aligned} Y_i^{(1)} &= \frac{X_i^{(1)}}{X_i} \\ &= \alpha_i \prod_{j=1}^{n} X_j^{g'_{ij}} - \beta_i \prod_{j=1}^{n} X_j^{h'_{ij}} \quad (i = 1, 2, ..., n) \\ &= \alpha_i \exp\left(\sum_{j=1}^{n} g'_{ij} Y_j \right) - \beta_i \exp\left(\sum_{j=1}^{n} h'_{ij} Y_j \right) \\ &= A_i^{(1)} - B_i^{(1)} \end{aligned} \tag{3.29}$$

ここで、$g'_{ij} = g_{ij} - \delta_{ij}$、$h'_{ij} = h_{ij} - \delta_{ij}$ である。δ_{ij} はクロネッカーのデルタ ($i = j$ のとき $\delta_{ij} = 1$、それ以外の時 $\delta_{ij} = 0$) である。

ここで α_i、β_i、g'_{ij}、h'_{ij} は全て定数なので、 式 (3.29) は簡単に微分できる。

$$\begin{aligned} Y_i^{(2)} &= A_i^{(1)} \left(\sum_{j=1}^{n} g'_{ij} Y_j^{(1)} \right) - B_i^{(1)} \left(\sum_{j=1}^{n} h'_{ij} Y_j^{(1)} \right) \\ &= A_i^{(1)} G_i^{(1)} - B_i^{(1)} H_i^{(1)} \\ &= A_i^{(2)} - B_i^{(2)} \end{aligned} \tag{3.30}$$

ここで $A_i^{(2)} = \sum_{j=1}^{n} g'_{ij} Y_j^{(1)}$、$B_i^{(2)} = \sum_{j=1}^{n} h'_{ij} Y_j^{(1)}$ である。これを一般化して、

$$\begin{aligned} Y_i^{(m)} &= A_i^{(m)} - B_i^{(m)} \\ A_i^{(m)} &= \sum_{q=1}^{m-1} F_{mq} A_i^{m-q} G_i^{(q)} \\ B_i^{(m)} &= \sum_{q=1}^{m-1} F_{mq} A_i^{m-q} H_i^{(q)} \\ G_i^{(q)} &= \sum_{j=1}^{n} g'_{ij} Y_j^{(q)} \\ H_i^{(q)} &= \sum_{j=1}^{n} h'_{ij} Y_j^{(q)} \end{aligned} \tag{3.31}$$

となる。ここで $F_{mq} = \frac{(m-2)!}{(q-1)!(m-q-1)!}$ である。さらに、少しでも計算時間を短縮するために、Y_i の代わりに $\tilde{Y}_i = \frac{Y_i}{m-1}$ を用いることにすると、

$$\begin{aligned} \tilde{Y_i^{(m)}} &= \tilde{A}_i^{(m)} - \tilde{B}_i^{(m)} \\ \tilde{A_i^{(m)}} &= \frac{\sum_{q=1}^{m-1} \tilde{A}_i^{m-q} \tilde{G}_i^{(q)}}{m-1} \\ \tilde{B_i^{(m)}} &= \frac{\sum_{q=1}^{m-1} \tilde{B}_i^{m-q} \tilde{H}_i^{(q)}}{m-1} \\ \tilde{G_i^{(q)}} &= \sum_{j=1}^{n} g'_{ij} \tilde{Y}_j^{(q)} \\ \tilde{H_i^{(q)}} &= \sum_{j=1}^{n} h'_{ij} \tilde{Y}_j^{(q)} \end{aligned} \tag{3.32}$$

という式が得られる。これは、以下の関係による。

$$\begin{aligned}
\tilde{Y_i^{(m)}} &= \frac{Y_i^{(m)}}{(m-1)!} \\
\tilde{A_i^{(m)}} &= \frac{A_i^{(m-q)}}{(m-q-1)!} \\
\tilde{B_i^{(m)}} &= \frac{B_i^{(m-q)}}{(m-q-1)!} \\
\tilde{G_i^{(q)}} &= \frac{G_i^{(q)}}{(q-1)!} \\
\tilde{H_i^{(q)}} &= \frac{H_i^{(q)}}{(q-1)!}
\end{aligned} \tag{3.33}$$

これにより二階微分を計算するときの乗算の回数を $\frac{1}{n}$ に、三階以上の微分係数については $\frac{1}{n(2m-5)}$ にすることができる。加算の回数は変わらない。一般的なテイラー展開では、その係数を計算するとき Y_i の m 階の微分係数を $m!$ で除した値が必要であるが、修正した式 (3.33) によるテイラー係数を用いれば、これを m で除するだけで得られる。つまり、既に計算した低次の (m がより小さい) 係数を用いれば、Y_i を $m!$ で除した値が簡単に得られることになる。

連立の次数を n とするとこの方法は、乗算の最低限回数はルンゲ-クッタ法の $12n^2+36n$ に対し $10n^2+37n$、べき乗計算の最低限回数はルンゲ-クッタ法 $12n$ に対し $2n$ になっている。式 (3.34) に示す連立微分方程式に対してルンゲ-クッタ法とこの方法を比較した場合、同じ精度ではこの方法が 228 倍高速であることが示されている [7]。

$$\begin{aligned}
\frac{dX1}{dt} &= 8X_1 - 8X_1X_2^{0.5} \\
\frac{dX2}{dt} &= X_1^{0.5}X_3^{-0.5} - X_2 \\
\frac{dX3}{dt} &= X_2^{0.5} - X_3 \\
X_1(t_0) &= \frac{1}{1024} \\
X_2(t_0) &= \frac{1}{16} \\
X_3(t_0) &= \frac{1}{4} \\
t_0 &= 0.0 \ \text{(計算開始点)}
\end{aligned} \tag{3.34}$$

$$t_f \quad = \quad 12.0 \ \text{(計算終了点)}$$

3.3 系の最適化法

微分方程式を解くことで得られた結果と、実験により観測された時系列データを比べることで、モデルがどの程度、現実の系をよく表現しているかが評価できる。この評価が高くなるようにモデルを最適化していくことになるが、ここではまず、非線形多次元関数の一般的な最小化アルゴリズムについて述べる。

3.3.1 一次元探索法

一次元探索法とは、独立変数が一つの目的関数を最小化、あるいは最大化するアルゴリズムである [30]。最小化とは、目的関数値が最小になる独立変数の値、すなわち最適解を決定することであり、これを逐次的な繰り返し計算により決定する場合には、探索とも呼ばれる。関数 $f(x)$ の最大化はすなわち、$-f(x)$ を最小化することであるので、以下では最小化についてのみ述べる。

関数の最小化は、解析的に一通りの計算で行える場合もある。例えば目的関数が独立変数の二次関数 $f(x) = ax^2 + bx + c$ の場合には、$a \neq 0$ なら $x = -b/2a$ となる点が $f(x)$ を最小化する点である。しかし一般的な応用問題では目的関数の形式が定まっておらず、独立変数が目的関数の記述の中に陽に含まれないことがある。つまり目的関数の記述形式の中に、独立変数値が直接には入っていないが目的関数の値を制御するといった、関数の関数 (汎関数) と呼ばれる形式である。実験で得られた時系列データと一致する挙動をする数理モデルを得ようとする場合はこれに相当する。つまり独立変数は数理モデルを記述するパラメータであり、目的関数は、そのモデルから微分方程式を解いて得られる時系列データと実験データとの二乗誤差などであるため、二乗誤差を計算する式の中には独立変数値は出てこない。関数の形式が多項式であれば、最小二乗法による回帰で、解析的に時系列データと一致する関数を決定することができるが、最小二乗法が適用できない場合には、その場合には、まずある独立変数値を出発点として任意にとり、そこから目的関数値や微分係数を利用して次の独立変数値を決定し、次第に最適解に近づいていくという方法、すなわち探索を行うことで関数を最適化していく [15][30]。

任意の多次元非線形関数の最適化のための多次元探索法では、その内部で一次元探索法が使われるが、これはその探索空間内である特定のベクトル (探索方向ベクトル) に沿った一次元空間において、目的関数を最小化する点を探す方法である。このことから、一次元探索は直線探索とも呼ばれる。一次元探索にはさまざまな方法があるが、全ての目的関

数、最適化条件に対して高速かつ高精度に最適化する普遍的な方法というのは、存在しない。また一次元探索能力によって多次元探索の成否や速度、精度に直接影響が出るため、目的関数の性質に応じた適切な一次元探索法を選ばなければならない。

以下では代表的な一次元探索法として黄金分割法、フィボナッチ探索法、ローゼンブロック法について述べるが、前二者のアルゴリズムは、どちらも探索範囲が有限である場合に用いられ、その探索範囲内に極小値が一つだけ存在する場合には、その極小点が求められるが、複数ある場合には、どの極小値が求められるかはわからない。しかし最終的に、一つの極小点が求められる。これら二者に共通するアルゴリズムは以下のようになる。

まず初期探索範囲中に一つの探索点を取り、探索範囲の境界(両端)とその点の3点で目的関数値を計算する。ここで目的関数を $f(x)$ とし、境界をそれぞれ a と c、その間の点を b とする ($a < b < c$、図3.3)。ここでもし、$f(a)$、$f(b)$、$f(c)$ のうち $f(b)$ が最小であれば、閉区間 (a, c) の中に極小点が存在する。そして新しい点 x を閉区間 (a, b) もしくは (b, c) の間にとる。ここでは (b, c) に取ったとする。もし $f(x) < f(b)$ なら、閉区間 (b, c) の間に極小点があることになる。逆に $f(x) > f(b)$ なら、極小点は閉区間 (a, b) 内にある。どちらの場合も、極小を含む区間の両端の中点が、極小点の近似である。この操作を、区間の幅が十分に小さくなるまで繰り返すことにより、望む精度の近似解を得る。

計算機による数値計算では、常に精度が問題となり、この場合も区間の幅が十分に小さくなるまでとはどの程度のことを言うのかが問題となる。計算機での実数の表現では、一つの数値を表現するのに4バイトを使う単精度実数と、8バイトを使う倍精度実数がある。それぞれ $3.0e^{-8}$、および $1.0e^{-15}$ 程度以下の違いは表現できない。そのため、区間の幅や評価関数の値もこれ以上小さな値や差を求めても無意味であるが、実際にはここまでの精度は求められない。上述の値を ϵ と表し、b の近くで評価関数 $f(x)$ をテイラー展開して(つまり $x - b$ が十分に小さいとする)高次の項($x - b$ の3乗以上の項)を無視すると、

$$f(x) \approx f(b) + f'(b)(x - b) + \frac{1}{2} f''(b)(x - b)^2 \tag{3.35}$$

となるが、b が極小点であるとすると、その点での微分係数は0であるため、

$$f(x) \approx f(b) + \frac{1}{2} f''(b)(x - b)^2 \tag{3.36}$$

となる。区間の囲い込みが進んで式(3.36)の第二項の絶対値が小さくなっていき、第一項 $f(b)$ の ϵ 倍よりも第二項が小さくなると、式(3.36)の右辺の加算を行ってもその結果は $f(b)$ と同じになり、これ以上の繰り返し計算は無意味になる。この時 $(\epsilon \frac{1}{2} f''(b)(x - b)^2 < f(b))$

$$|x - b| = \sqrt{\epsilon} b \sqrt{\frac{2|f(b)|}{b^2 f''(b)}} \tag{3.37}$$

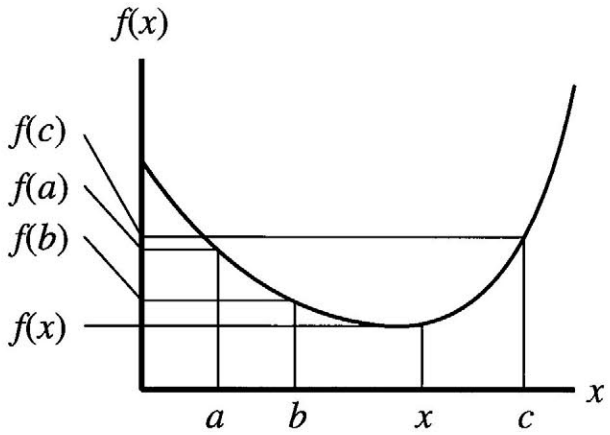

図 3.3: 探索範囲が決まっている場合の一次元探索法の概略図。極小値は点 a と点 c の間にあるものとし、まず点 b を a と c の間にとって a、b、c の 3 点で目的関数値 f を計算、そして閉区間 $[a,b]$ または $[b,c]$ のどちらかの内側に新しい点 x を取る (図では $[b,c]$ 内に取っている)。区間の内側に取った 2 点について目的関数値を比較し、$f(x) < f(b)$ であれば、極小値は区間 $[b,c]$ 内にあり (図の場合)、そうでなければ極小値は区間 $[a,b]$ 内である。前者の場合であれば、点 b、x、c を後者であれば点 a、b、x を新しい 3 点として、区間の幅が十分に小さくなるまで同じことを繰り返す。

である。式(3.37)中の$\frac{2|f(b)|}{b^2f''(b)}$はおおよそ1のオーダーである。なぜなら、極小値付近の目的関数$f(x)$の曲線を二次式Ax^2+Bx+Cで近似すると、$f''(x)=A$となり、

$$\frac{2|f(b)|}{b^2f''(b)}=\frac{2|Ab^2+Bb+C|}{b^2A} \tag{3.38}$$

であるから、影響の大きな二乗の項に注目すると、そのオーダーは1である。またbが非常に大きくなれば、この項は0に近づく。xとbを囲い込む区間の両端とすると、区間の幅を、注目している区間周辺の独立変数値の$\sqrt{\epsilon}$倍よりも小さくすると、式(3.36)の左辺は$f(b)$と等しくなってしまうことが多くなる。つまり探索精度の限界は$1.0e^{-4}$あるいは$1.0e^{-8}$ということになる。

黄金分割法

黄金分割法[30]とフィボナッチ探索[30]は、上述した共通したアルゴリズムと図3.3において、点bや点xのとり方が異なる。黄金分割法では、まず点a、cが与えられたとき、Wを規定の定数として

$$\frac{b-a}{c-a}=W,\frac{c-b}{c-a}=1-W \tag{3.39}$$

となる点bをとる(図3.4)。点bは点aから$c-a$のW倍だけ右に進んだ点である。また、定数Zを用いて

$$\frac{x-b}{c-a}=Z \tag{3.40}$$

となる点xをとる。点xはbから右に$c-a$のZ倍進んだ点である。すると、新しい囲い込みの区間の幅は、現在の幅$|c-a|$の$W+Z$倍か、$1-W$倍になる。そのうちの広い方にしか囲い込めなかった場合の区間の縮まり方を最大にするためには、$W+Z=1-W$となるようにZをとればよい。したがって

$$Z=1-2W \tag{3.41}$$

とする。このとき、新しい点xは区間内でbと対称な位置にくるため、区間(a,b)と(b,c)のうち広い方の中にとられる。次の探索のための新しい3点はa,b,xまたはb,x,cのいずれかになるが、どちらの場合でももとの3点a,b,cの位置関係を保存するようにするためには、

$$\frac{Z}{1-W}=\frac{W}{1} \tag{3.42}$$

でなければならない。この式と式(3.41)から、次の二次方程式

$$W^2-3W+1=0 \tag{3.43}$$

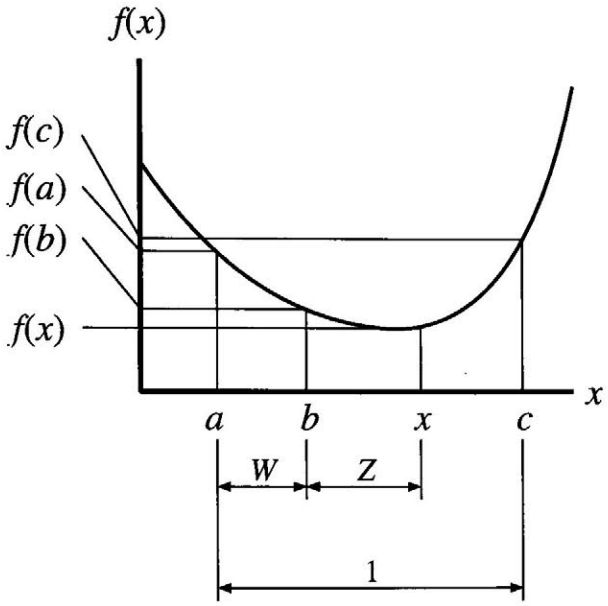

図 3.4: 黄金分割法。点 a、b、c は探索開始時に与えられるものとする。点 x は点 c から $c-a$ の 0.38197 倍だけ左に戻った点にとる。

が得られ、したがって

$$W = \frac{3-\sqrt{5}}{2} \approx 0.38197 \tag{3.44}$$

となる。つまり最初の囲い込み区間 (a,c) で、点 b は $ab : bc = 0.38197 : 0.61803$ に内分する点となるようにとるということである。この比は黄金分割比であるため、この方法は黄金分割法と呼ばれる。

黄金分割法は、探索の最初に与えられる 3 点 $(a < b < c)$ の次にとる点は、広い方の区間の中央の点ではない方の点 (a または c) から、中央の点 b に向かってもとの区間の幅 $|c-a|$ の 0.38197 倍進んだ点とするアルゴリズムである。

フィボナッチ探索法

フィボナッチ探索法 [30] は、黄金探索法で用いる黄金比の代わりに、フィボナッチ数列による比を用い、また繰り返し回数をあらかじめ制限しておく方法である。フィボナッチ数列 F_n は、$F_n = F_{n-2} + F_{n-1}$, $F_0 = F_1 = 1$ で定義される。

まず最大の繰り返し回数 N を決めておき、与えられた探索範囲 (a,c) を $F_N : F_{N-1}$ に

内分する点を b とする (逆に $F_{N-1} : F_N$ でもよい)。そして広い方の区間を $F_{N-1} : F_{N-2}$ に内分する点を新しい探索点 x とする。k 回目 $(k > 1)$ にとる新しい探索点は、極小値のある方の区間を $F_{N-k+1} : F_{N-k+2}$ に内分する点にとるようにする。N 回探索点をとった時点でフィボナッチ数列を使いきって終了する。このときに、区間が十分狭くなるように探索範囲の広さ $|c-a|$ から N を決めておく。同時にこの時には、$F_1 = F_2$ から、最後の二つの探索点は重なってしまうため、これらはその位置からできるだけ小さく離れた 2 点をとり、それを代わりに用いる。

1 回目の探索点をとったとき、それにより最初の探索区間は $F_{N-1} : F_{N-2}$ に内分される。最初の探索範囲を d_1、k 回目の探索点でできた極小点の存在する区間の幅を d_k とすると、

$$d_k = \frac{F_{N-k+1}}{F_{N-k+2}} d_{k-1} \tag{3.45}$$

となる。すると最後の区間との幅は、

$$\frac{d_1}{F_N} \tag{3.46}$$

となり、最終的に重なる 2 つの探索点のずれる、小さな幅を ϵ とするとこれは、

$$\frac{d_1}{F_N} + \epsilon \tag{3.47}$$

となる。この幅を任意の値 δ 以下にするためには、

$$F_N > \frac{d_1}{\delta - \epsilon} > F_{N-1} \tag{3.48}$$

となるように N をとればよい。

二次式内挿法

以上の探索法では、極小点を含むことが保証されている範囲を得ることができる。これを繰り返すことで、計算機による実数表現の精度の限界まで範囲の幅を狭めていくことができるが、これには計算時間がかかる。連続な関数では、微分係数が 0 となる点では二次関数で近似できることが、テイラー展開を用いることで分かる。そこで、黄金分割法やフィボナッチ探索法で得られる最後の 3 点の目的関数値を比較し、その差があらかじめ定めた範囲に収まっていれば、この 3 点から二次関数の係数を決定し、この二次関数で目的関数を近似し、二次関数の極小点を解とする。

二次関数を $y = f(x) = ax^2 + bx + c$ とし、3点を (x_1, y_1)、(x_2, y_2)、(x_3, y_3) とすると、

$$\begin{aligned} x_1^2 a + x_1 b + c &= y_1 \\ x_2^2 a + x_2 b + c &= y_2 \\ x_3^2 a + x_3 b + c &= y_3 \end{aligned} \tag{3.49}$$

である。これからクラメルの公式で二次関数に含まれる係数 a、b を求めると、

$$\begin{aligned} a &= \frac{\begin{vmatrix} y_1 & x_1 & 1 \\ y_2 & x_2 & 1 \\ y_3 & x_3 & 1 \end{vmatrix}}{\begin{vmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \end{vmatrix}} = \frac{x_3 y_2 - x_2 y_3}{x_2^2 x_3 - x_3^2 x_2} \\ b &= \frac{\begin{vmatrix} x_1^2 & y_1 & 1 \\ x_2^2 & y_2 & 1 \\ x_3^2 & y_3 & 1 \end{vmatrix}}{\begin{vmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \end{vmatrix}} = \frac{x_2^2 y_3 - x_3^2 y_2}{x_2^2 x_3 - x_3^2 x_2} \end{aligned} \tag{3.50}$$

となる。するとこの二次関数の導関数 $y' = 2ax + b$ が0となる $x = \tilde{x}$ が極小点である。

$$\tilde{x} = -\frac{b}{2a} = \frac{x_3^2 y_2 - x_2^2 y_3}{2(x_3 y_2 - x_2 y_3)} \tag{3.51}$$

ローゼンブロック法

このアルゴリズムは成功と失敗のルーチン [16] とも呼ばれ、探索範囲に制限を設ける必要がない。また多峰性の関数に対しては、探索開始点や初期ステップ幅などを変更することにより、得られる局所解が容易に変化する。このため、目的関数の形状が全く不明な場合に、まず極小値の存在する範囲を確定したい場合などに用いる。そしてその確定した範囲内を黄金分割法などで絞り込み、最終的に二次式内挿法で近似解を求めるということになる。

このアルゴリズムではまず、初期ステップ幅 h、拡大率 $\alpha(\alpha > 1)$ と縮小率 $\beta(0 < \beta < 1)$ を決めておく。初期出発点 x_0 に対し、$x_1 = x_0 + h$ となる点を取り、$f(x_1) < f(x_0)$ であれ

ば (成功)、$h = \alpha h$ とし、$x_0 = x_1$ とする。もし $f(x_1) > f(x_0)$ であれば (失敗)、$h = -\beta h$ とする。このとき、x_0 は更新しない。この操作を繰り返す。つまり、探索点を更新して目的関数値が小さくなれば、その方向への探索を続ける、そうでなければ逆方向に探索するという考えで、同方向に探索を続けていく場合にはステップ幅を大きくしていくことで、探索の加速を図る。そして目的関数値が大きくなれば、その二つ前の探索点と、目的関数値が大きくなった点との間に極小値が存在することになり、その範囲を黄金分割法などで探索すればよい (図 3.5)。一番最初の探索点から開始して、どちらの方向に進んでも目的関数値が大きくなる場合には、目的関数値が大きくなった二つの点の間、探索の出発点付近に極小値があることになる。またどちらに進んでも目的関数が小さくなる場合には、どちらにも極小値がある可能性があり、どちらを探索するかは任意に選ぶ。

拡大率 α と縮小率 β の値が 1 と大きく異なると、より広範囲を探索することになり、また極小値が一つしかない場合には、そこへの収束が速くなるが、多峰性の場合に、探索点を更新する距離の中に極小値が複数あると、かならずしも特定の極値に収束していくとは限らず、探索が混乱することがある。しかし拡大率 α と縮小率 β の値を 1 に近くすると、それだけ収束には時間がかかる。これらの値は、目的関数の性質に応じて経験的に決定するしかない。一般には、これらの値は黄金比にとって、探索範囲を決定して黄金分割法に移行する際に、改めて目的関数値を計算する手間を省くことが多い。

3.3.2 多次元非線形関数の最適化

一般的な多次元非線形関数の最適化法は、二つに大別される。一つは上述した一次元探索法を内部で利用する、逐次的かつ解析的な探索法、もう一つは乱数を用いた確率的、発見的な探索法である。前者は 20 世紀後半、計算機の発展とともに研究、開発されてきた、探索空間の中で探索点を逐次的に更新していき、極小点に収束していく方法であり、後者は熱力学にヒントを得た焼きなまし法 (シミュレーティッドアニーリング)[10] や 1980 年代から研究が盛んになった、遺伝的アルゴリズム (GA)[2][3][5][6] などである。一般に解析的方法は、確率的方法に比べると目的関数の値を計算する回数が非常に少なくて済み、それだけ高速で、高精度を得られやすい。反面、目的関数の性質により最適化の可否が大きく左右され、自然界の現象を記述する系にたいして有効でないことが多い。これは、自然現象の記述においては、探索空間の非線形性が強く、目的関数を二次関数で近似できる範囲が非常に狭いことが多いのと、探索空間の次元が高いことが多いことによる。対して確率的方法では、目的関数の値さえ計算できれば最適化できる方法であり、適用範囲が解析的方法に比べると広く、自然現象のモデルの最適化に用いる例も少なくない。また遺伝的アルゴリズムにおいては探索点を同時に多数発生させるため、関数の概形をつかみやす

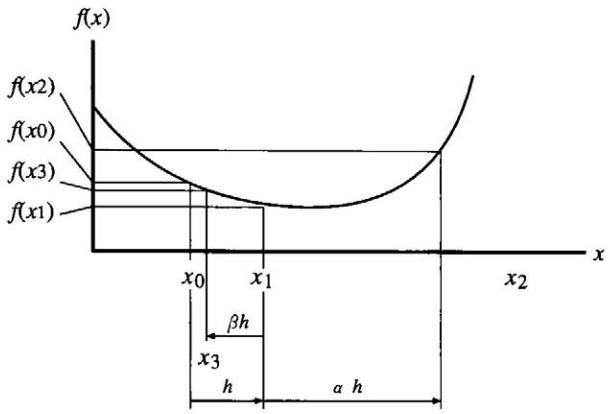

図 3.5: ローゼンブロック法による探索。この方法では、初期探索点は一つだけ決めればよい。まず初期ステップ幅h、拡大率αと縮小率βを決めておき、$\alpha > 1$、$\beta < 1$にしておく。まず$x_1 = x_0 + h$となる点をとり、$f(x_1) < f(x_0)$となれば、この方向への探索は成功であるとし、$h = \alpha h$として同様に$x_2 = x_1 + h$となる点をとる。ここで$f(x_2) < f(x_1)$となれば、この方向への探索は成功であり、同様の操作を繰り返すが、$f(x_2) > f(x_1)$となれば、x_1からx_2への探索は失敗であり、x_1から、x_2とは逆方向へ探索をすすめる。つまり$h = -\beta h$として、$x_3 = x_1 + h$となる点を次の探索点とする。これを繰り返し、探索点を更新しても目的関数値の改善量がごく少なくなった場合($|f(x_3) - f(x_2)| \leq \theta$、$\theta$は前もって決めておく定数)や、極小値を囲い込んだ範囲($|x_3 - x_1|$)が前もって決めておく定数よりも小さくなった場合に、探索を終了する。

く、極小点が多数ある場合に、より目的関数値の小さな極小点を探しやすい。しかしそのため、目的関数を計算する回数が前者に比べると非常に多く、これが最適化に要する時間を短縮する上でのボトルネックになる。また高精度を得るためには、要求精度が高くなるに従って非常な勢いで探索点数を増やしていく必要があり、時間と精度の面では解析的な方法が、適用範囲と探索空間の大きさ (次元) の限界では確率的な方法が優れていると言える。

まず、解析的な方法について述べる。

最小点には、その点を含む十分に小さい領域を考えたときに、その領域内で目的関数値を最小にする極小点 (局所的最小点) と、独立変数の取りうる値全てを含む領域内で、目的関数が最小の値を取る最小点 (大域的最小点) がある。以下では、極小点について述べる。

一変数の関数について、その極小点には以下の性質がある。

> 二階微分可能な一変数の目的関数 $f(x)$ が、ある十分小さい領域内の点 $\bar{x}$ で極小値をとるための必要条件は
>
> $$f'(\bar{x}) = 0 \text{ かつ } f''(\bar{x}) \geq 0 \tag{3.52}$$
>
> であり、十分条件は
>
> $$f'(\bar{x}) = 0 \text{ かつ } f''(\bar{x}) > 0 \tag{3.53}$$
>
> である。

必要条件が成立している場合、目的関数や拘束条件によって極小点である場合とそうでない場合がある。たとえば目的関数が $f(x) = x^3$ の場合、$f'(x) = 0$ となるのは $x = 0$ のときだけであるが、このとき目的関数は極小ではない。しかし独立変数のとりうる範囲を限定し、極小点がその境界上に存在する場合にはこの条件は正しくない。このときの必要条件は、極小点が領域の下限であれば $f'(x) \geq 0$、上限であれば $f'(x) \leq 0$ である。これを独立変数値に制限のない、多次元関数に拡張すると以下のようになる。

> 多次元の目的関数 $f(\bar{\mathbf{x}})$ が $\mathbf{x} = \bar{\mathbf{x}}$ で極小となるための必要条件は
>
> $$\mathbf{f}_x(\bar{\mathbf{x}}) = \mathbf{0} \text{ かつ } \mathbf{f}_{xx} \geq 0 \tag{3.54}$$
>
> となることであり、十分条件は
>
> $$\mathbf{f}_x(\bar{\mathbf{x}}) = \mathbf{0} \text{ かつ } \mathbf{f}_{xx} > 0 \tag{3.55}$$
>
> である。ここで、
>
> $$\mathbf{f}_x(\mathbf{x}) \equiv \left(\frac{\partial f(\mathbf{x})}{\partial x_1}, \frac{\partial f(\mathbf{x})}{\partial x_2}, \frac{\partial f(\mathbf{x})}{\partial x_3}, \ldots, \frac{\partial f(\mathbf{x})}{\partial x_n}\right)^T \tag{3.56}$$

$$[\mathbf{f}_{xx}] \equiv \begin{pmatrix} \frac{\partial f_x(\mathbf{x})^2}{\partial x_1^2} & \frac{\partial f_x(\mathbf{x})^2}{\partial x_1 x_2} & \cdots & \frac{\partial f_x(\mathbf{x})^2}{\partial x_1 x_n} \\ \frac{\partial f_x(\mathbf{x})^2}{\partial x_1 x_2} & \frac{\partial f_x(\mathbf{x})^2}{\partial x_2^2} & \vdots & \cdots \\ \vdots & \cdots & \ddots & \vdots \\ \frac{\partial f_x(\mathbf{x})^2}{\partial x_1 x_n} & \frac{\partial f_x(\mathbf{x})^2}{\partial x_2 x_n} & \cdots & \frac{\partial f_x(\mathbf{x})^2}{\partial x_n^2} \end{pmatrix} \tag{3.57}$$

であり、nは独立変数の個数である。また$\mathbf{0}$は全ての要素が0であるベクトルを、$[\mathbf{f}_{xx}] > 0$は、$\mathbf{f}_{xx}$の全ての要素$\mathbf{f}_{xx,ij}$について$\mathbf{f}_{xx,ij} > 0$であることを表す。

上述の$\mathbf{f}_{xx}$を関数$f(\mathbf{x})$のヘッセ行列、またはヘシアンという。

これは独立変数の取りうる範囲に制限がない場合であるが、これに制限がある場合には、上述の条件は、独立変数がその範囲内でかつ境界上でない場合であり、境界上での極小点の性質は、以下のようになる。

目的関数$f(\mathbf{x})$がある範囲Sの境界上の点$\mathbf{x} = \bar{\mathbf{x}}$で極小をとるための必要条件は、十分小さな値$t$に対して$\mathbf{x} + t\mathbf{y}$が$S$内に収まるような$\mathbf{y}$に対して

$$(i) \quad \mathbf{f}_x^T(\bar{\mathbf{x}})\mathbf{y} \geq 0 \tag{3.58}$$

$$(ii) \quad \text{もし } \mathbf{f}_x^T(\bar{\mathbf{x}})\mathbf{y} = 0 \text{ なら、} \mathbf{y}^T[\mathbf{f}_{xx}]\mathbf{y} \geq 0 \tag{3.59}$$

となることである。

3.3.3 焼きなまし法

以下に述べる焼きなまし法(シミュレーティッドアニーリング法、Simulated Annieling)と遺伝的アルゴリズムは、モンテカルロ法[37]と並ぶ代表的な確率的探索法である。

焼きなまし法[15]は前述したように、乱数を用いた逐次探索法である。多変数関数に対しても、その関数値さえ計算することができれば、最適化することができる。関数の連続性や導関数値の計算の可否などは問わない。しかし探索の点の更新は、正規分布乱数と式(2.6)に示す確率のみによって行うため、探索空間内で極小値を与える点には、確率的にしか到達し得ない。したがって解析的な方法に比べ極小点への収束が遅く、目的関数値の計算回数が非常に多くなり、要求精度が高い場合には、最適化に非常に長い時間がかかるという欠点を持つ。また逐次探索であるため、探索空間の次元が高くなるにつれて、急激に最適化に要する時間が長くなっていく。

生体内反応系の構造最適化にこれを適用する場合、実験で得られる系の時系列データとモデルから計算される挙動の差異、つまり時系列データの二乗誤差和などを目的関数と

することができる。すると式(2.6)のTを設定することで最適化を行い、観測データを再現する系を得ることができる。

3.3.4 遺伝的アルゴリズム

焼きなまし法による探索は、一つの探索点を更新していくことで最適化を進めていく逐次探索であるため、得られる目的関数に複数の極小点がある場合、どの極小点を得るかは、初期値に非常に大きく依存し、また極小点への収束も遅い。これを改善するために、探索を一点だけを使って逐次的に行うのではなく、常に複数の点での目的関数値を評価する、同時多点探索を導入したのが遺伝的アルゴリズム (Genetic Algorithm, GA)[3][5][6][11][31]である。

これにより初期値依存性を減らすことができ、また探索点を更新する際に、一つの探索点を複数の過去の探索点から決定することで、効率のよい探索をすることができる。また焼きなまし法同様に、目的関数はその値さえ計算できればよく、関数の連続性や導関数値の計算の可否などは問わない。

しかし遺伝的アルゴリズムを実際の最適化問題に適用するためには、最適化条件や解の表現方法など、設計あるいは設定せねばならない事項が非常に多い。このため現実的な問題として、最適化問題への適用開始から解決までの時間を考えると、解析的最適化または焼きなまし法のプログラムを作成して、非常に多数の初期値から繰り返し最適化を行って、得られた極小値のうち、もっともよいものを解とする方が早いこともある[3][5]。

生体内反応系の構造最適化では、モデルを記述するパラメータが多い、つまり探索空間の次元が非常に高い。また数理モデルとしてS-システムあるいは一般質量作用則を用いた場合、パラメータの値によっては微分方程式が数値的に解けない場合があり、探索空間中のそういった場所では、目的関数値が計算できない。どちらの数理モデルでも、パラメータの値を用いて状態変数の指数乗を計算し、しかもそれらの積を計算せねばならないため、有限桁の実数しか表現できない計算機上では、パラメータの値によってはしばしば桁あふれを起こし、やはり微分方程式が解けないことがある。つまり目的関数が未定義の範囲が、探索空間中に少なからず存在し、それらの範囲の境界で目的関数は不連続となる。

解析的探索法や焼きなまし法では、探索点が目的関数の未定義領域の境界に近づいてきた場合、探索点を大幅に移動して未定義領域から離れるということは行わない。このような場合に、同時多点探索を行っていれば、問題なく探索を続けることができる。そのような理由からここでは、確率的な同時多点探索として、近年非常に盛んに研究、応用されている遺伝的アルゴリズム (GA) を最適化法として導入した。

GAを生体内反応系の構造最適化に導入するにあたって、設定あるいは設計しなければならない事項には、個体(および個体に含まれる、数理モデルの実数パラメータ)の表現方法、個体の評価方法、淘汰、交叉、突然変異などの適用方法などがある。GAとは、最適化法としては概念的なもので、具体的な適用法によってアルゴリズムの詳細は変化し、探索の速度も大きく変わる。このため、上記の事項それぞれについてさまざまなアルゴリズムやデータ構造などからもっとも適当なものを選ばねばならないが、現在これらについて有効性や妥当性を判断する一般的な方法はなく、適用される最適化問題の性質によって、その都度評価、考察などを行って、試行錯誤的に決定せねばならない。ここでも、さまざまな方法を比較、検討した。

第4章　最適化アルゴリズム

これまでは、一般的な生化学反応系のモデリングおよび最適化について述べたが、ここから、本研究で開発した最適化手法について述べる。

一般的に逆問題を解こうとするとき、解析的に唯一あるいはただ一通りの解を見つけるには総探索を行うしかない。これは解空間の次元が高くなるとすぐに現実的な時間では解くことができなくなる。このため解空間の一部を探索し、最適解を試行錯誤的に探していくアプローチを取らざるを得ない。

探索空間が一次元空間である逆問題の場合には、黄金探索法、フィボナッチ探索法、ローゼンブロック法 (成功と失敗のルーチン) などの簡便な方法 (3.3.1) がある。これらは多次元空間を探索する場合にも用いられる、基礎的な方法であり、どの一次元探索法を用いるかで探索の効率が大きく左右される。また同様に、多次元探索法も解析的な方法では勾配を用いる方法と用いない方法、または発見的な手法と、どれを用いるかが大きな問題となる。いずれにしても、探索空間の特徴により、いかに適切なものを選ぶかという問題である。

ここで対象としている探索空間は、与えられた時系列データを再現するネットワークを記述する、数理モデルである。数理モデルを S-システム (式 (3.10)) で記述すると、モデルは実数パラメータの組で表現されるため、モデルの最適化問題は、モデルを記述する各実数パラメータの値を決定するということになる。ここで、ネットワークの構成要素の個数が n の時、モデルは $2n(n+1)$ 個の実数パラメータの組で表現されるため、最適化問題における探索空間は $2n(n+1)$ 次元となる。

解析的な多次元探索法は、探索空間の次元が高くなっていくに連れて、探索方向の一時独立性を失っていき、探索が困難になることが知られている。したがって、ここでは発見的探索法を取る。発見的探索法には遺伝的アルゴリズム (GA)、シミュレーティッドアニーリング、モンテカルロ法などがあるが、前二者は、既に分かっている範囲内での準最適解の周辺を探索するという点で、似た方法である。後者は、探索点をランダムに増やしていくにつれて探索範囲が徐々に絞られ、最適解に漸近していくことが必要であり、目的関数の作成が困難であるという欠点を持つ。このためここでは、GA を用いた。GA においても、目的関数の形状は最適化の可否、速度に決定的な影響を持つが、原理的にはどの

様な形状でも最適化は可能である。またGAで行われる最適化のための操作は非常に幅が広く、一度GAによる最適化プログラムを作成すれば、わずかなパラメータ値の変更でシミュレーティッドアニーリング的な性格を持たせることもでき、また操作を特徴づける最適化パラメータを変更することにより、局所探索や大域探索といった最適化の性質を動的に制御できる。これらは解析的な最適化法にはない、GAの利点である。

4.1 最適化対象と遺伝的表現

ここでの最適化対象はネットワークモデルであるが、これをどのように計算機内で表現するかは、目的関数の形状にも直接影響する、非常に重要な問題である。

計算機で取り扱うためには、モデルは実数または整数の数値の集合で表現されなければならない。一般にネットワークは、構成要素とそれらの間の相関と捉えられ、結合行列やグラフなどで表現されることが多い。ここでは、与えられた時系列データを再現するモデルを得ようとしている。そのため、ネットワークモデルをシミュレートして、モデルから時系列データが得やすい形式であることが求められる。時系列データに特に制約を求めず、任意の曲線であることを許すためには、ネットワークにもそれに応じた自由度が求められる。こういった連続量を表すための数理モデルとして、化学反応系を扱う場合には一般に、質量作用則に基づいた微分方程式が用いられる。しかし質量作用則によるモデルの形式は表現したい系の構造によって異なるため、計算機で取り扱うためにはその形式を一般化する必要がある。それが一般作用則とよばれる、以下の形式で表される連立微分方程式である[25]。

$$\frac{dX_i}{dt} = \sum_{k=1}^{A} \alpha_{ik} \prod_{j=1}^{n} X_{jk}^{g_{ijk}} - \sum_{k=1}^{B} \beta_{ik} \prod_{j=1}^{n} X_{jk}^{h_{ijk}} \tag{4.1}$$

ここにX_iは状態変数であり、化学物質の濃度や温度などの物理量を表す。nはその系内での総数である。AはX_i増加させようとする作用(合成反応など)の数、α_kはその速度係数、BはX_i減少させようとする作用(分解反応など)の数、β_kはその速度係数である。g_{ijk}はX_iを増加させる反応kにおいて状態変数X_jがX_iに及ぼす影響、h_{ijk}はX_iを減少させる反応kにおいて状態変数X_jがX_iに及ぼす影響である。ここで、A、Bおよびnが定まるとパラメータの総数が求まるが、これは$(A+B)2n(n+1)$となる。このAとBの値は、それぞれの状態変数に固有の値となり、それが質量作用則が反応系ごとに異なった形式を取る理由となる。これの形式を一般化するためには、AおよびBをそれぞれ、その形における最大値、または予想される最大値に取り、各状態変数について共通の値を用いるようにする。

しかし一般的には、ある状態変数X_iの値を増加、あるいは減少させるようなプロセス

の個数は、あまり大きくない数に限られているため、式(4.1)のモデルではg_{ijk}やh_{ijk}のほとんどが0であり(X_iの生成あるいは分解反応kにおいて、X_jが直接にはX_iの増減に影響を与えない)、α_{ik}やβ_{ik}についても同様である($A=10$のときにX_iの合成反応が2種類しかなければ、$\alpha_{i,3}$〜$\alpha_{i,10}$は0である)。したがって、状態変数を増加させる作用および減少させる作用をそれぞれ、一つのプロセスに近似して表現する方法が考えられる。それが以下に示す、S-システムと呼ばれる数理モデルである[17]。

$$\frac{dX_i}{dt} = \alpha_i \prod_{j=1}^{n} X_j^{g_{ij}} - \beta_i \prod_{j=1}^{n} X_j^{h_{ij}} \tag{4.2}$$

各変数及び添字の意味は、式(4.1)と同じである。このモデルでは、状態変数X_iを増加させるプロセスおよび減少させるプロセスをそれぞれ一つにまとめて記述する。これにより、モデルを記述するためのパラメータの総数は$2n(n+1)$にまで減らすことができる。つまりモデルを記述するパラメータの個数が$\frac{1}{A+B}$になったということであり、最適化問題の対象としてみると、探索空間の次元数が$\frac{1}{A+B}$になったということである。このモデルはネットワークモデルとして見ると、全結線型であり、要素(状態変数)X_iとX_jの間に関与するパラメータはg_{ij}、g_{ji}、h_{ij}、h_{ji}の4つである。これは有効グラフによるモデルの2倍であり、この冗長分は各要素について、それを増加させようとする作用と減少させようとする作用に分けて要素間相互作用を考えることによる。また、状態変数間の全ての相互作用について、その大きさを表す変数が含まれていることは、構造が未知のネットワークを記述しようとする際に、必要不可欠である。

4.1.1 実数の表現

ネットワークを計算機内でモデルとして表現するためには、まず実数値の表現形式を決めなければならない。一般にGAでは2進数の数字を必要な精度だけ並べて表す。つまり一般的な計算機での実数の内部表現である、浮動小数点形式(IEEE規格)をビット列として用いることが多い(図4.1)[5][31]。

この形式では、仮数部の右の方の桁の値がGAの遺伝的操作により値が変わっても、実数値そのものはあまり変化せず、また指数部が変わると大きく実数値が変わる。そのため局所探索を行いたい場合は右側の、大域探索を行いたい場合は左側の方に対してより高い確率で突然変異を起こすことで、探索範囲の広さを制御できる。また個々の実数値に対して交叉を行えるため、遺伝的な多様性を保ちやすい[14]。

しかしこの表現方法では、差の小さい二つの数で、その表現が大きく異なっている場合がある。たとえば整数では、10進数の7と8の差は1しか違わず、もっとも近い整数どう

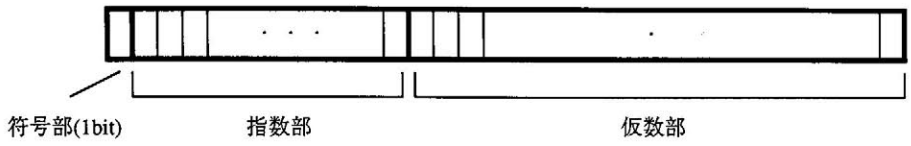

図 4.1: 実数の 2 進数を用いた浮動小数点形式による表現。符号部、指数部、仮数部の 3 つからなり、符号部は実数の正負を、指数部は実数の桁数を、仮数部は精度を表す。一般的な計算機では、一つの実数値を 32 ビットで表す単精度実数の場合、符号部は 1 桁 (1 ビット)、指数部は 8 桁、仮数部が 23 桁である。実数値は符号部×指数部×仮数部で表される。符号部は例えば 0 なら正、1 なら負であり、指数部はその最上位桁 (もっとも左のビット、Most Significant Bit、MSB) をその符号として、残りをそのまま 2 のべき乗数とする。仮数部は少数点以下を 2 進数で表したものである。したがって符号部 0、指数部 5(2 進数で 101)、仮数部 0.123 の場合、その表す実数値は $(+1) \times 2^5 \times 0.123 = 3.936$ である。

しであるが、2 進数ではそれぞれ 0111 と 1000 であり、全てのビットで異なっていて、もっともかけ離れたものとなっている。この異なるビットの個数をハミング距離という。浮動小数点実数ではもっと深刻なことになり、このユークリッド空間と探索空間でのハミング距離の違いが、数値最適化では局所探索の速度に大きな影響を与えるため、これらが比例する、符号なし整数を用いたスカラー表現を導入する [22][23][24]。つまり探索空間をある整数値で等分割し、その各点を整数値で表す。表現したい実数 r の範囲を $r_{min} \leq r \leq r_{max}$ とし、それを表現する符号なし整数 u を $0 \leq u \leq u_{max}$ とすると、

$$r = \frac{u}{u_{max}} (r_{max} - r_{min}) \tag{4.3}$$

となる。これにより実数空間での距離と探索空間での距離が比例することになる。したがって GA では、この符号なし整数は数値としてのみ扱い、その内部表現には全く触れないことにする。

4.1.2 モデルの表現

最適化対象となるモデルは、S-システムで記述されるネットワークの数理モデルである。S-システムモデルはそのパラメータセット $\alpha_i, \beta_i, g_{ij}, h_{ij}$ $(1 \leq i \leq n, 1 \leq j \leq n, n$ はモデルを構成する要素の個数) で定義されるため、計算機内でのモデルはこれらの値全てを持つ必要がある。またパラメータ値は実数であり、前節で述べた方法で表現される。GA を最適化に導入するにあたって、最適化対象はネットワークモデルであるから、GA

における個体をそれぞれ一つのネットワークモデルとする。ネットワークモデルは一組の実数パラメータで定義されるため、個体もS-システムの各パラメータを適当な方法で表現される実数として保持していればよい。詳細については第5章で述べる。

4.2 遺伝的アルゴリズムにおける遺伝的操作

S-システムの導入によりパラメータ総数を減らすことができたが、しかしその総数は系の要素数を n としたときに $2n(n+1)$ である。したがってモデルを決定するためには $2n(n+1)$ 個の実数パラメータを同時に決定しなければならない。このような多変数同時最適化問題には、最急勾配法などの逐次的な解析的方法を適用することがあるが、これらの方法は探索空間の次元が高くなるにつれ、探索方向の多様性を保つことが難しくなることに加え、内部で用いる一次元探索法の性質により、多峰性の関数に対しては最適解を得ることが保証されないことが知られている。つまり探索開始点によって決まる、ある一つの局所解のみしか得られない。

これは逆問題に対しては致命的な欠点である。したがって、ここでは発見的探索手法を導入することとし、もっとも代表的で一般的である遺伝的アルゴリズム (以下GA) に骨格構造化と呼ぶ方法を導入したアルゴリズムを用いることとした[22][23][24]。GAを適用するにあたって非常に多くの最適化パラメータを設定しなければならないが、それらを操作することにより大域探索性と局所探索性、探索のランダムさなどを制御することができ、シミュレーティッドアニーリングとほぼ同じような探索をすることもできる。また解析的な探索手法とも組み合わせて最適化することができるため、非常に自由度の高い探索手法でもある。

4.2.1 初期集団の生成

あらかじめ決められた数 P(以下個体数と呼ぶ) の個体を生成し、それをGAにおける個体集団とするが、生成される個体が持つS-システムパラメータ α_i、β_i、g_{ij}、h_{ij} に対し、それぞれ探索範囲をあらかじめ決めておき、その範囲内で一様乱数を発生しパラメータの値とすることで、全くランダムに個体を生成する。P の値は探索空間の次元および評価関数の形状に応じて、経験的に決定する。

4.2.2 評価

与えられた時系列データと、各個体から微分方程式を解くことで、それぞれ計算される時系列データの異なりの度合いを表す関数を決め、その関数値を各個体の評価値(適合度)とする。

$x_{i,t}$を、サンプリングポイントtにおける要素iの観測値、$X_{i,t}$を、サンプリングポイントtにおける要素iの、個体が表現するモデルから計算される値、Tをサンプリングポイントの総数、nを要素数とすると、相対二乗誤差の和eと評価値fは以下のように計算される。

$$\begin{aligned} e &= \sum_{t=1}^{T}\sum_{i=1}^{n}\left(\frac{x_{i,t}-X_{i,t}}{x_{i,t}}\right)^2 \\ f &= \frac{1}{e} \end{aligned} \tag{4.4}$$

4.2.3 終了条件

集団中でもっとも評価値の高い個体から計算される時系列データが、与えられた時系列データに十分に近いと判断される場合に、最適化を終了し、その個体を最適解とするのがもっとも自然な考え方である。

ここでは、現実に実験で得られたデータを適用する前に、数理モデルから計算された時系列データを用いて最適化を行う。したがってデータには測定誤差などは含まれない。したがって

高精度に解を求めようとすれば、与える時系列データと個体から計算される時系列データの二乗誤差が計算機での実数演算の精度で表現できる限りの最小値になるまで最適化を行えばよい。この判定法は解析的探索法では一般的な収束判定であるが、しかしGAでは確率的にしか局所解へ収束しないため、ほとんどの場合で収束が判定されず、非現実的である。また、実際に観測される実測値のデータでは10%からときには100%を越えるような大きな測定誤差が含まれることが少なくないため、観測データの再現精度をどの程度であれば最適化できたと判断するか、という基準を考えにくい。

そのためここでは、最適化の進行の様子を見て、収束したように思われるところで終了とする。つまり世代交代を行う毎にエリート個体の評価値を調べ、一定数の世代交代を行ってもエリート個体の評価値が変化しない場合に、最適化を終了することとする。

4.2.4 淘汰

淘汰とは、評価値の低い個体は消去してしまい、残った個体のうち評価値が高い個体ほど多くの子個体の親となる可能性が高くなるようにする操作である。一般的に、個体集団から評価値の高い個体を選び、生き残る個体とする。この生き残る個体の総数をあらかじめ決めておき、その数だけ個体集団からの選択を繰り返す。このため淘汰は、選択淘汰あるいは単に選択とも呼ばれる。広く用いられている淘汰の方法には、ルーレット戦略とランキング戦略がある。

ルーレット戦略では、各個体はその評価値に比例した確率で生き残る。つまり評価値の合計を F、個体 i の評価値を f_i としたとき、その個体 i が生き残りとして選ばれる確率は f_i/F である。生き残りとして選ばれなかった個体は消去される。評価値が高ければ複数回数選ばれる可能性があり、その場合は増殖したことになる。評価値の差が出にくいような最適化では、個体の多様性が保たれ (さまざまな個体が生き残る)、大域探索的な性質が保たれる。評価値を二乗誤差の逆数を使って計算するような場合は、誤差が小さくなるにしたがって評価値が急速に大きくなるため、集団としての遺伝的多様性が失われやすい。つまりエリート個体の周辺のみに個体が集中することになり、局所探索性が強くなる。

ランキング戦略では、個体は評価値の順に並べられ、その順位にしたがった確率で選択される。もっとも単純な方法は順位と選択確率を比例させることであり、これは線形ランキング戦略 [2] と呼ばれる。また指数関数を使う方法もある [11]。どちらでも、個体間で評価値に大きな差がある場合でも、その順位のみにしたがって選択確率が決定されるため、遺伝的多様性を保ちやすい。

線形ランキング戦略では、比例定数を変えることにより、いわゆる淘汰圧を制御することができる。つまり比例定数が小さければ、ランダムな選択に近くなり、比例定数を大きくとれば、評価値の高い個体がより選ばれやすくなる。この比例定数は、パラメータ η を使って表され、順位が i の個体が選択される確率 p_i は以下の式で表される。

$$p_i = \frac{1}{P}\left(\eta^+ - (\eta^+ - \eta^-)\frac{i-1}{P-1}\right) \quad (1 \leq i \leq P) \tag{4.5}$$

ここで P は総個体数、η^+ と η^- はそれぞれ、選択確率の最大及び最小期待値である。$\sum_{i=1}^{P} p_i = 1$ でなければならないことから、$1 \leq \eta^+ \leq 2$、$\eta^- = 2 - \eta^+$ である (図 4.2)。η の値を大きく取れば、すなわち選択淘汰における淘汰圧が強いことに相当し、順位の低い個体が抹消されやすくなる。

ここでの最適化問題は、時系列データに基づいた数理モデルの最適化である。つまり数理モデルを定義する実数パラメータを決定することが目的であるが、GA のような試行錯誤的なアプローチでそれを行う場合、探索中に発生する多くの探索点において、その点が表す数理モデルが、シミュレーションを行うことができないようなモデルであることがあ

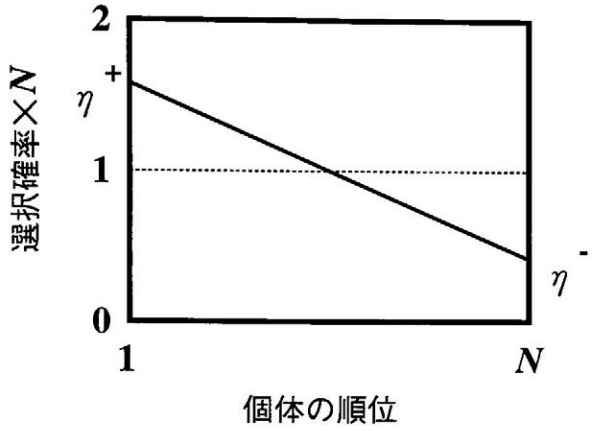

図 4.2: Baker の線形ランキング戦略を用いた場合の、個体の評価値順位とその個体の選択確率。$\eta^+(=2-\eta^-)$ が大きくなると、評価値の大きな個体は選ばれやすく、評価値の小さな個体は選ばれにくくなり、η^+ を小さくしていくと、どの個体も同じ確率で選ばれるようになっていく。

る。つまり、その探索点が表す数理モデルすなわち連立微分方程式が、数値的に解けないということである。この場合、その数理モデルに対しては、評価値を 0 にする。これにより、こういったモデルを表す個体を、評価に続く交叉、突然変異などといった遺伝的操作の対象外とする。

しかしこうすると、探索空間内において評価関数値が 0 である領域が複数存在することになり、遺伝的操作によりその領域に発生した個体は、全く探索の役に立たない。GA により大域探索が行われる場合には、突然変異、または探索空間内で遠くへだった個体同士の交叉による場合であり、新しく生成される個体の評価値が 0 になるかどうかの見通しは立たない。こういった状況を避けるためには、まず微分方程式が解ける個体をランダムな探索により発見し、その個体の周辺を集中的探索することで、高速な局所探索を行い、局所解を得るという操作を基本として、これを繰り返す、または GA の個体集団を複数に分けてそれぞれにおいて、この基本的な局所探索を行う、などして大域探索にすることが有効であると考える。

したがってここでは、より探索の局所性を高めるために、ランキング戦略を改良した方法で淘汰を行う。すなわち選択確率をエリート個体に対して非常に高く設定し、突然変異によりその周辺だけを探索することにする。選択確率 c_i は以下の式で計算する。

$$c_i = (1-p_i)\prod_{j=0}^{i-1} p_j \quad (i=1,2\ldots,P) \tag{4.6}$$

ここでp_iは式(4.5)で定義されるものであり、$p_0 = 1$とする。Pは総個体数である。これにより、ほぼエリート個体の周辺だけで探索が行われる。式(4.5)における$\eta^+(= 2 - \eta^-)$の大きさが淘汰圧を表し、この値が大きいほど、評価値の順位が低い個体が淘汰されやすくなるという特徴は同じである。

4.2.5 交叉

交叉とは、二つの個体から一つあるいは複数の個体を合成する操作である。もととなる個体を親個体、合成される個体を子個体と呼ぶ。子個体は、親個体の特徴を受け継ぐものであり、探索空間内では、二つの親個体を補間する位置付近に作られる。各個体を実数値の集合として考えると、それぞれは探索空間内の一つの点であると捉えられるが、交叉の操作により、二点の平均の点や、その近辺の点を生成する。一般的に親個体と遠くない個体を生成する操作であり、これは局所探索(あるいは近傍探索)に相当する。

GAによる数値最適化では、実数を浮動小数点形式によるビット列で表現した場合、そのビット列の途中にランダムに交叉点をとり、新しく合成される子個体の、交叉点よりも前の部分は、片方の親個体の交叉点よりも前の部分、子個体の交叉点よりも後ろの部分はもう片方の親個体の交叉点よりも後ろの部分からそれぞれ複製することで、子個体を合成する(図4.3)。交叉点は一つではなく、複数とってもよい。交叉点の数か増えるほど、遺伝的な多様性を増しやすい。つまり大域探索的な性質を強くできるが、それだけ収束しにくくなる。

ここでは一つの個体が多数の実数パラメータを含む構造を取っており、実数の表現形式に符号なし整数を用いると、各実数についてその中に交叉点を取ることができないため、異なる方法を取る。つまり子個体に含まれる実数パラメータの値は、二つの親個体の一方から選んだ値とするという方法である。各パラメータについてどちらの親から選ぶかは、ランダムに決める(図4.4)。またそれぞれ対応するパラメータから選ぶ。あるiとjについて、子個体のα_iは親個体のα_iから、子個体のβ_iは親個体のβ_iから選び、子個体のg_{ij}は親個体のg_{ij}から、子個体のh_{ij}は親個体のh_{ij}から選ぶ。これを全てのi、jについて行うことで交叉を完了し、子個体を合成する。交叉点の数はパラメータ総数-1(たとえば、パラメータ数が2の時は交叉点は1か所)、つまり要素数をnとするとき$2n(n+1)-1$点交叉とよばれる交叉である。

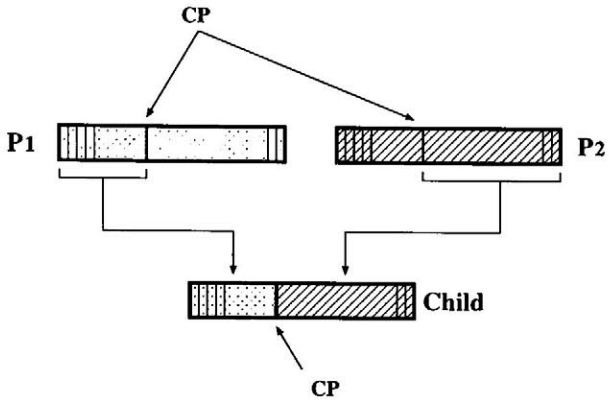

図 4.3: ビット列同士の交叉。二つの親となる個体、または親に含まれるビット列から、ランダムに決めた交叉点よりも前の部分と後ろの部分をそれぞれとりだし、取り出したものを連結して子個体、あるいは子個体に含まれるビット列とする。ビット列が浮動小数点形式の実数である場合、おおまかには交叉点が左端に近ければ近いほど、新たに作られる子個体が表す実数値、親個体が表現する実数値から大きく離れた値になると言える。

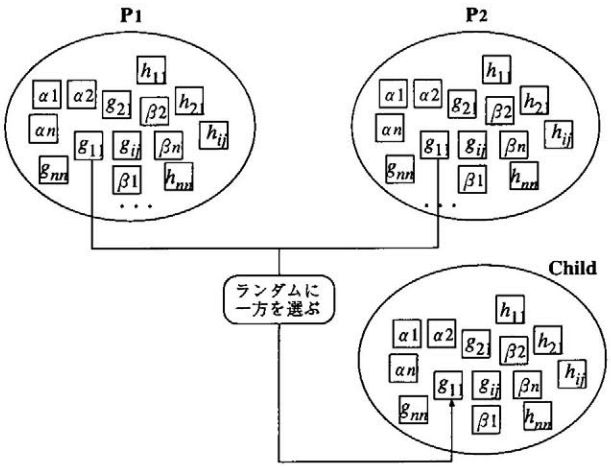

図 4.4: パラメータごとの交叉。子個体に含まれるパラメータの値は、親個体の対応するパラメータから選ぶ。その際、どちらの親から選ぶかは、等確率にランダムに決める。

4.2.6 突然変異

交叉だけでは、初期集団中に現れなかった実数値は、探索されないままである。そこで乱数により探索点を決定することで、親個体の集団から交叉のみによっては発生し得ない点を生成する。これは親個体とは遠く離れた点を生成する可能性を持っており、大域探索に相当する。

突然変異率が低い場合は、乱数による探索があまり行われないということであり、初期集団の周囲のみを集中的に探索する、局所探索的な性格が強い最適化法になる。目的関数が多峰性である時は、一つの局所解の周辺を集中的に探索するため、高速に局所解を発見することができる。

逆に突然変異率を高くすると、大域探索的な性格が強くなり、探索点が広範囲に分散することになる。そのため一つの局所解に収束しにくくなり、最適化に長時間かかることになるが、最適解発見の可能性が高くなり、収束した場合には、突然変異率が低い場合に比べ、より評価値の高い解を発見しやすくなる。

一般には探索範囲を限定しないために、一様乱数を用いる。しかし正規分布乱数を用いることで、探索の性質を制御することができる。つまり集中的に探索したい領域の中心を、発生する乱数の平均とし、局所的に探索したいときには乱数の分散を小さく、大域探索をしたいときには大きくすればよい。さらに、集団全体として評価値が上がっていかなくなった場合は、全ての個体がある評価値の高い集団の周囲に集まってしまい、局所解に捕らわれている場合がある。こういった場合には突然変異率を大きくして、大域探索に切り替えるとよい。またそれでよりよい個体が見つかった場合には、突然変異率を小さくして、よりよい個体の周辺を集中的に探索すると、極小点への収束が早くなる。

4.2.7 エリート戦略

各探索ステップにおいて、それまでに発見された最良解をエリート個体として保存しておく。そして淘汰、交叉、突然変異の各遺伝的操作によって作られた個体集団に、エリート個体を加える。これにより、交叉により生成される個体がこのエリート個体の周辺に多くなり、局所探索的な性質を持ち、最適化を高速化できる [4]。反面、個体の遺伝的多様性は失われやすくなるため、個体集団をエリート個体とそれとほぼ同じ個体が支配してしまうのを避けるために、淘汰、突然変異の方法に工夫が必要である。

4.3 骨格構造化

生体内反応系を表現する式(4.1)におけるg_{ijk}とh_{ijk}や式(4.2)におけるg_{ij}とh_{ij}は、全ての状態変数間の作用を記述し得るが、現実の生体内反応系では、各状態変数(反応に関与している物質の濃度または量)は他のわずかな個数の状態変数としか直接的には作用を及ぼさないことが多い。つまり各物質はそれぞれごく少数の化学反応にかかわっており、それぞれの化学反応もごく少数の物質同士の反応である。多数の反応にかかわる物質もあるが、その数は全体から見ると少数である。したがって、状態変数間の直接的な相互作用を表す変数(式(4.1)におけるg_{ijk}およびh_{ijk}、あるいは式(4.2)におけるg_{ij}とh_{ij})は、その値の大半が0である。

したがって、逆問題に対するアプローチとして二通りの方法が考えられる。一つは、式(4.1)および(4.2)における各i、つまり各状態変数について、非ゼロの値を持つ指数係数(式(4.1)におけるg_{ijk}、h_{ijk}または式(4.2)におけるg_{ij}とh_{ij})は、その個数を制限しておくという方法であり、もう一方は、ある閾値をあらかじめ決めておき、最適化の進行中に、値がその閾値より小さくなった指数係数は、値を0にするという方法である。前者の方法では、確実に簡素な数理モデルを得ることができる。しかし一般に最適化しようとする対象では、制限すべき個数が未知である場合がほとんどあり、これを不適切に設定してしまうと、構造最適化が全くできなくなってしまう可能性がある。したがってここでは、後者を導入する。GAにおける世代交代が既定回数繰り返される毎に、指定された閾値以下の値を持つ指数係数は、その値を0にする。これを骨格構造化と呼ぶことにする。

4.4 最適化アルゴリズム

以上のことをまとめると、本研究で設計、開発した最適化アルゴリズムは以下のようになる。

ここで最適化のために与えられる情報は、最適化対象となる系の状態変数の個数N、N個の状態変数の時系列データ(サンプリングポイント数T)である。つまりデータの点数はNT個である。そして最適化の前に、S-システムパラメータ(α_iおよびβ_i、g_{ij}およびh_{ij})に対してそれぞれ、探索範囲$R_{ab} \geq 0$、$R_{gh} \geq 0$を決めておく。すなわち探索において、$0 \leq \alpha_i < R_{ab}$、$0 \leq \beta_i < R_{ab}$、$-R_{gh} \leq g_{ij} < R_{gh}$、$-R_{hh} \leq h_{ij} < R_{gh}$とする。また、突然変異率初期値$m_0$、突然変異率変更の頻度$G_m$、突然変異率の更新倍率$k$、正規分布乱数の分散の初期値$d_0$、正規分布乱数の分散の変更後の値$d_g(d_g \geq d_0)$、骨格構造化の閾値$s$、骨格構造化の頻度$G_s$、終了判定のための世代交代回数$G_t$を決めておく。

(1) 初期化:　一様乱数を用いてモデルを生成する。モデルに含まれるパラメータのう

ち α_i と β_i は 0 以上 R_{ab} 以下、g_{ij} と h_{ij} は $-R_{gh}$ 以上 R_{gh} 以下となるようにする。モデルは P 個生成する。GA において、生成した各モデルが個体、その集合が集団、P が個体数である。

(2) 評価: 集団中の各個体について、その評価値を計算する。個体が持つ S-システムパラメータに基づいて連立方程式を定義し、それを数値的に解いて、観測値である時系列データとの、各サンプリングポイントにおける相対二乗誤差の和を計算し、その逆数を個体の評価値とする。

(3) 終了判定: すでに世代交代を行っていた場合、現在のエリート評価値が得られてから行われた世代交代の回数が既定の世代交代回数 G_t に達した場合、または世代交代の回数が既定の最大世代交代回数 G_{max} に達した場合に、最適化を終了する。どちらの場合も、その時点でのエリート個体を最適解とする。

(4) 淘汰: 集団から、重複を許して $P-1$ 個の個体を選び出す。個体を評価値の順に並べ、順位 i の個体が選ばれる確率は、式 (4.6) の $c_i(i,\eta^+,\eta^-)$ で計算される。選ばれた $P-1$ 個の個体にエリート個体を加えて、親集団とする。

(5) 交叉: 親集団から等確率で親個体となる 2 個体を選び出し、交叉により 1 個体を生成する。子個体に含まれる全ての S-システムパラメータについて、親個体の持つ対応するパラメータの値とする。どちらの親のパラメータを用いるかは、等確率に選ぶ。子個体は $P-1$ 個生成する。

(6) 突然変異: 子個体に含まれる全ての S-システムパラメータに対し、突然変異率 m の確率でその値を変更する。変更する値は、各パラメータの探索範囲内の値で、一様乱数あるいは正規分布乱数で決定する。正規分布乱数を用いる場合は、乱数の平均はもとのパラメータの値、分散 d は既定の初期値 d_i である。エリート個体の評価値が、既定の G_m 回の世代交代を行っても改善されない場合、局所解に捕らわれていると判断し、突然変異率 m を k 倍する。正規分布乱数を用いている場合にはこれと同時に、乱数の分散を d_g にする。世代交代によりエリート評価値が改善した場合には、m と d はそれぞれ初期値 m_0 および d_0 に戻す。

(7) 世代交代: (4)～(6) の操作を行って得られた $P-1$ 個の個体とエリート個体を合わせた P 個の個体で、集団を置き換える。これが世代交代となる。

(8) 骨格構造化: 世代交代を既定の回数 G_s だけ行ったとき、新しい集団を評価する前にすべての個体に含まれる指数係数について、その絶対値が既定の閾値 s よりも小さな場合には、その指数係数の値を 0 にする。これは G_s 回の世代交代の度に行う。そして骨格構造化を行った場合も行わなかった場合も (2) に戻る。

第5章　実装

前述のアルゴリズムを用いて、実際に最適化を行い系の構造を推測するために、計算機上でプログラムを作成した。プログラムの記述には ANSI 規格の C++ 言語を用い、計算機は OS として米国 Compaq 社の Digital UNIX version 4.0E を搭載するコンカレント・システムズ社の Tempest II(CPU: Alpha 21164A、クロック:600MHz、SPECfp95: 21.3、SPECint95: 18.6) を用いた。

5.1　最適化プログラムの概要

一般に計算機プログラムには、入力と出力が必要である。第4章で述べたアルゴリズムでは、入力はシステムを構成する要素数と各要素の時系列データ、および最適化を制御するパラメータであり、出力はS-システムパラメータで記述された系の構造である。プログラムは利用者からみると、時系列データを入力すると系の構造が出力されるブラックボックスであるとも言える。プログラム内部では、乱数で作成した系のモデルに対して遺伝的アルゴリズム (GA) と骨格構造化を適用し、次第に入力データと矛盾のないモデルへと最適化していく。その詳細を以下に述べる。

5.2　反応系の数理モデル

計算機内部では、GA の各個体は一つの系のモデルを表現する。ここではモデルに S-システムを採用したため、各個体は要素数を n とするとき $2n(n+1)$ 個の実数を持つ。またそのほかに、その個体の評価値を持つ。これらの値は全て実数である。したがって、まずその実数を計算機内で表現する方式、次に複数の実数値を保持する形式が必要となる。

5.2.1 実数表現 - 1. 浮動小数点表現

一般に計算機で実数を取り扱うためには、実数を2進数で表現しなければならない。またどんな桁の実数でも同じ精度を持たせ、表現できる範囲を広くするために、浮動小数点方式が考え出された。GAを用いた数値最適化でも、これを用いることが多い。

2進数の各桁はビットと呼ばれるため、実数を2進数で表現するということは、実数をビット列で表すということでもある。ビット列は3部に分けられる。1ビットからなる符号部、複数ビットを持つ指数部、やはり複数ビットを持つ仮数部である。符号部のビットは0か1の値を持ち、これは数値としてそれぞれ1か-1と解釈される。つまり符号ビットが0であればそのビット列が表す実数は正の数、1であれば負の数である。指数部は、そのまま2進数の整数として扱われる。たとえば指数部が3桁で、その各ビットがそれぞれ010であれば、これは10進数で2である。仮数部は1.0未満の実数である。仮数部の上位ビットからi番目のビットの値をb_iとし、仮数部のビット数はnであるとすると、仮数部の表す値は

$$\sum_{i=1}^{n} b_i \frac{1}{2^i}$$

である。たとえば仮数部が4桁でその値が1000であれば、これは

$$1 \times \frac{1}{2} + 0 \times \frac{1}{4} + 0 \times \frac{1}{8} + 0 \times \frac{1}{16}$$

であり、10進数で0.5である。符号部の値をs、指数部の値をe、仮数部の値をmとすると、そのビット列の表す実数値は$s2^e m$である。0.0は仮数部が全て0であることで表す。このとき指数部と符号部は無視される。

計算機内ではこれらのビットを全て並べて、一つのビット列とする(図4.1)。最上位(もっとも左側)ビットを符号部、それに続いて指数部、末尾に仮数部という並びが一般的である。上述の例(指数部010、仮数部1000)で符号部が1である場合は、そのビット列は10101000という8ビットの列になる。そしてこの表す実数値は

$$s2^e m = (-1) \times 10^{10} \times \frac{1}{10^{11}}(2\text{進数}) = (-1) \times 2^2 \times \frac{1}{2^1}(10\text{進数}) = -2$$

である。

指数部については、負の整数も表現する必要がある。しかし指数部の先頭ビットを符号部にすると0を表すビット列のパターンが2種類あることになり無駄であるため、バイアス表現を用いる。つまりnビットで0以上の整数を表すとすると、その範囲は0から$2^n - 1$であるが、ビット列が表す0以上の整数から、2^{n-1}を減じて指数の値とする方法である。

GAによる突然変異では、各ビット毎に突然変異を起こすことが考えられるが、このとき符号部以外の左側のビットほど、突然変異が起こったときの実数値としての値の変化が大きくなる。突然変異率をビットの位置によって変えることによって、探索の大域探索性や局所探索性を制御することができる。

これをC++言語のプログラム内で表現し利用するためには、クラスを作成しなければならない。本研究ではbsrealという名前のクラスを作成し、これを実現した。このクラスは単精度整数へのポインタを3つ持ち、それぞれ指数部へのポインタ***exp**、仮数部へのポインタ***man**、これらを連結して一つのビット列としてみたときの、そのビット列へのポインタ***str**である。***exp**と***str**はこのように異なる意味を持つが、実際に持つ値はは同じアドレスである。符号部は仮数部の先頭ビットを割り当てる。指数部をeビット、仮数部をmビットとするとき、クラスのインスタンスが生成されるときには$e+m+1$ビットのメモリを確保し、その先頭アドレスを***str**に格納する。そして***exp**は***str**に、***man**は***str**+eにセットされる。以下に示すコードは実際に用いたコードの一部である。**EXP_BITS**が上述のe、**MANTISSA_BITS**が上述の$m+1$に相当する。コンストラクタのコードは示してあるが、他に加減算と代入演算子、C++言語に備え付けのdouble型との型変換演算子を備えている。

```
#  define EXP_BITS       (3)
#  define MANTISSA_BITS  (5)
#  define ALL_BITS       (EXP_BITS + MANTISSA_BITS)
class bsreal {
  short *str;
  short *exp;
  short *man;
public:
  bsreal() {
    int i, j;
    str = (short *)calloc(ALL_BITS, sizeof(short));
    exp = str;
    man = str + EXP_BITS;
    for (i = 0; i < EXP_BITS; i++)       exp[i] = (int)(rand()%2);
    for (i = 0; i < MANTISSA_BITS; i++) man[i] = (int)(rand()%2);
  }
};
```

5.2.2 実数表現 - 2. 符号なし整数を用いた表現

浮動小数点方式は、広く用いられている表現方法ではあるが、第4.1章に述べたような問題がある。つまり二つの実数があるときに、探索空間内でのその二つの実数間の距離(ハミング距離)と、その表す実数のユークリッド空間での距離(ユークリッド距離)が大きく異なることがあるということである。ハミング距離とは、浮動小数点方式で表された二つの実数をaとb、それぞれの上位からi番目のビットをa_i、b_iとしたときの、$a_i \neq b_i$となるiの個数のことである。たとえば絶対値の大きな実数に対して、その符号部だけを変えた実数は、もとの実数からのハミング距離は1で、0以外の距離では最小であるが、ユークリッド距離はその絶対値の2倍であり、二種類の距離の値が大きく異なる。指数部ではその違いがもっと大きくなり得る。

これは探索において、局所探索に不都合を生じる。上述の例で実数値-0.5は10100010であると述べたが、-0.4375は10100111である。この二つの実数値の差は0.0625であり、もとの数0.5の12.5%である。比べて浮動小数点表現では、ハミング距離は4であり、もとの数10100010の持つ8ビットのうち、半数が異なっている。つまり実数空間でのユークリッド距離にたいして、ハミング距離が非常に大きい、と言える。ここで-0.5が最適値で-0.4375が探索点であるとすると、実数値としては最適値を得るためには、ほんのわずかの変化を要するだけだが、そのために探索空間では大きなハミング距離を適切な方向に移動せねばならない、ということである。

そこで、実数空間での二つの実数の違い(ユークリッド距離)と探索空間での違い(ハミング距離)の相関をできるだけ単純にするために、実数空間をスケーリングして、整数で表すことが考えられる。つまりアルゴリズム内で表される実数表現は、計算機があらかじめ用意している整数をそのまま用い、ビット列としては扱わないということである。計算機で用意している整数にはその表現範囲に限りがあり、また実数の精度を制御するため、表現できる実数の範囲をあらかじめ決めておく必要がある。さらに表現を簡潔にするため、整数は0以上しか用いない(これを符号なし整数と呼ぶ)。

表現される実数の最小値をR_{min}、その最大値をR_{max}、整数の最大値をnとすると、符号なし整数値iで表現される実数の値Rは

$$R = i\frac{R_{max} - R_{min}}{n} \tag{5.1}$$

となり、二つ表現の探索空間内での距離と、それらの表す実数値間の距離が比例する。

これはC++言語のプログラム内では、unsigned int型を用いて表現される。前説の浮動小数点方式と互換性を持たせるため、bsrealクラス内でC++言語のプリプロセッサのdefine構文を用いて、浮動小数点方式と符号なし整数方式を切り替えることにした。したがってプログラム内では、浮動小数点方式でも符号なし整数方式でもS-システムパラメー

タを保持する変数の型は bsreal 型である。以下のコード内にある real bs_upper_range と bs_lower_range は、それぞれ表現される実数の上限と下限、式 (5.1) に示す R_{max} と R_{min} である。また ULONG_MAX は C++言語の符号なし整数の表現できる最大値である。浮動小数点方式の場合と同様、ここに示すほかに加減演算子、代入演算子、C++言語の double 型との型変換演算子を備えるが、他に乗除演算子を備える。

```
extern real bs_upper_range;
extern real bs_lower_range;
class bsreal {
  unsigned long int num;
public:
  bsreal() {
    num = (unsigned long int)0;        // とりあえず中身を 0 で初期化
  }
  bsreal (real xx)                     // 引数があるときはそれを代入
  {
    if (xx > bs_upper_range) xx = bs_upper_range;
    if (xx < bs_lower_range) xx = bs_lower_range;
    num = (unsigned long int)
      ((xx-bs_lower_range)/(bs_upper_range-bs_lower_range)*ULONG_MAX);
  }
};
```

5.2.3 モデルの形式

ここでは、系のモデルの表現形式として S-システム (式 (4.2)) を導入した。これにより系を実数パラメータの組で表現することができる。前説で述べた実数の表現形式を用いて、その集合をどう表すかという問題になるが、これは遺伝的アルゴリズムにおける個体表現と密接にかかわる。つまり、個体表現の形式により交叉や突然変異といった遺伝的操作の形式を決めなければならないが、これらの操作はそれぞれ概念的に、局所 (近傍) 探索及び大域探索に相当するからである。

数理モデルとしての形式は式 (4.2) で定めるとおりである。これによりモデルを定義するためには α_i、β_i、g_{ij}、h_{ij} の合計 $2n(n+1)$ 個のパラメータを定める必要がある。ここではこれらのパラメータを、式 (4.2) の形式に沿って並べたものを一つのパラメータセッ

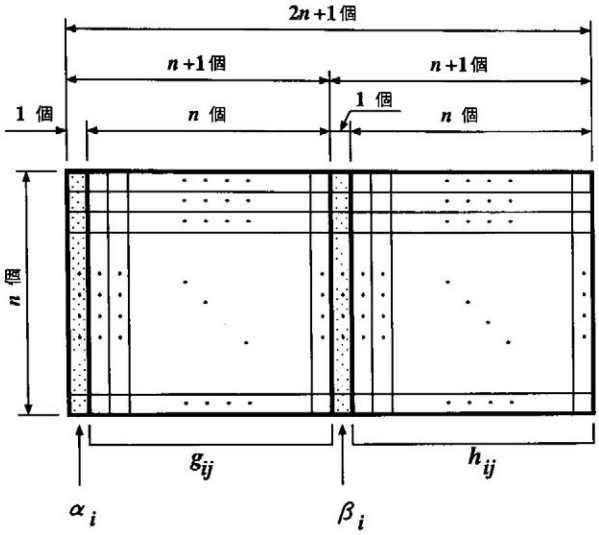

図 5.1: S-システムパラメータを行列上に並べた図。系に含まれる要素数が n のとき、全てのパラメータが一組になって $n \times 2(n+1)$ の行列を作る。各行は式 (4.2) において、一つの要素 X_i の挙動を記述するためのパラメータの組 $(\alpha_i, \beta_i, g_{ij}, h_{ij},\ \ 1 \leq j \leq n)$ に相当する。

トとした。系の要素数を n とするとき、この並びは $n \times 2(n+1)$ の行列になる。この行列は部分行列として二つの n 次の正方行列 g_{ij}、h_{ij} と二つの n 次のベクトル α_i と β_i を持つ (図 5.1)。

この行列に含まれる各要素 (S-システムパラメータ) は実数値である。最適化対象はモデルであるため、一つの個体が一つのモデルを表すとするのが適当である。この場合に考えられる交叉の操作は、子の個体を作る際に、それに含まれる実数値を親からどう取り出すか、つまり交叉点を発生させる単位はどうするか、ということである。一般的な GA による数値最適化では、一つの数値をビット列としてあつかい、その中に一つまたは複数の交叉点をとる。これはユークリッド距離とハミング距離の違いが問題になる場合には、大域探索性を高める働きがあり有効であるが、符号なし整数を用いた場合には数値をビット列として扱わないため、適用できない。ここでは、符号なし整数は C++言語に用意されている型 (Unsigned int 型) をそのまま用いるため、その内部表現に手を加える操作は想定しない。

C++言語で記述したプログラム内では、各パラメータを表現する実数は、前説で述べた **bsreal** 型の変数である。プログラムでは、これを保持するためのクラス **genome** を作成した。このクラスの詳細は、後の章で述べる。

5.3 微分方程式の数値解法

ここでは微分方程式の形式がすでに特定されていることから、アーヴィンとサバジョーの方法のみを用いた。前章で述べたアルゴリズムを、そのままC++言語でコーディングした。

微分方程式を解く際の刻み幅は、ユーザーが指定するか、そうでなければ入力される時系列データから自動的に決定する。これは、サンプリング時間をサンプリングポイント数で除した平均サンプリング間隔を、200で除した値である。この数値は経験的に決定した。

対数空間で微分方程式を解くため、各変数の初期値が0であると、その対数が計算できず、微分方程式を解くことができない。したがって与えられた時系列データにおいて、時刻が0の、時系列データの最初のデータは0であってはならない。0が与えられた場合には、0の代わりに、計算精度から見て結果に影響与えず、対数の計算だけを可能にするため、$1.0e^{-15}$という値を用いる。

また、微分方程式を解く際の刻み幅や時間の範囲に制限を設けなかったため、精度を確保しやすいようにテイラー展開は10次の項まで行うことで固定した。これを少なくすると、それに比例して微分方程式を解くのに要する時間は減少する。しかしこのコードでは、連立微分方程式の元数をnとしたとき、乗算の回数が$O(n^3)$であり、これによる影響が非常に大きいため、精度を悪くすることによる劇的な計算時間の短縮は望めない。

5.4 遺伝的アルゴリズム

アルゴリズムの概要は前章で述べたとおりである。以下には、具体的な計算方法及びC++言語での実装について述べる。

5.4.1 個体表現

各個体は、S-システムパラメータを一組だけ、および自分自身の評価値の合計$2n(n+1)+1$個の実数を持っている。評価される前は、その評価値は0である。概念的には図5.1のように一つの行列の形であるが、内部ではコードの可読性の面から二つのベクトルと二つの正方行列をそれぞれ分けて保持している。

C++言語でプログラムするにあたり、一つの個体を表すクラス`genome`を作成した。ベクトルを一次元配列、行列を二次元配列で保持するため、クラスのメンバーは前述の`bsreal`型へのポインタへのポインタが二つ(`**a`と`**b`、それぞれS-システムパラメータのα_iと

β_i)、**bsreal**型へのポインタへのポインタへのポインタが二つ(***g**と***h**、それぞれS-システムパラメータのg_{ij}とh_{ij})、それと評価値を保持する実数型の変数が一つである。一次元ベクトルは、C++言語では一般的にポインタを使うことが多いが、ここでは型が既に用意されている実数型ではなく、コンストラクタ呼び出しを要するクラスである。すでに大きさの決まっている配列を静的に確保する場合は、ポインタで差し支えないが、動的に確保する際には**new**演算子を使ってインスタンスを生成する。この演算子はインスタンスへのポインタを返すため、これを利用するためには、ポインタへのポインタを用いねばならない(下記コード参照)。

また、三種類の遺伝的操作のうち交叉と突然変異はメンバー関数を用いて行う。これは、呼び出し側の関数のコードをより簡潔にするためである。またここではS-システムパラメータは全て**public**であるが、これを**private**にしてメンバー関数以外から操作できないように仕様を変更しても、遺伝的操作がそのまま行えるようになっている。淘汰は、S-システムパラメータ値には関与せず、その操作は集団全体に対して行うため、メンバー関数にするのは適当ではない。なお、下記コード中の**makechild()**というメンバー関数は、別の**birth()**というメンバー関数内でのみ用いられる関数である。

```
class genome {                  // 個体を表す染色体の型
public:
  bsreal **a;                   // S-system の係数
  bsreal **b;                   //       〃
  bsreal ***g;                  //       〃
  bsreal ***h;                  //       〃
  real   point;                 // 個体の評価値

  void birth(genome *p, genome *c);  // 増殖関数  引数:交叉の相手と子供
  void mutation(bsreal, int);        // 突然変異関数
  friend void makechild(genome *par1, genome *par2, genome *child);
  friend int gnmdiff(genome *, genome *);

  genome() {// コンストラクタ: メモリを確保し、0で初期化する
    int i, j, k;
    real tmp;

    age = 0; point = 0.0;
    a = (bsreal  **)calloc(CHEMICALS, sizeof(bsreal  *));
    b = (bsreal  **)calloc(CHEMICALS, sizeof(bsreal  *));
    g = (bsreal ***)calloc(CHEMICALS, sizeof(bsreal **));
    h = (bsreal ***)calloc(CHEMICALS, sizeof(bsreal **));
    if ((a==NULL)||(b==NULL)||(g==NULL)||(a==NULL))
      puts("Allocation Error.");
    for (i = 0; i < CHEMICALS; i++) {
      a[i] = new bsreal;
      b[i] = new bsreal;
      g[i] = (bsreal **)calloc(CHEMICALS, sizeof(bsreal *));
      h[i] = (bsreal **)calloc(CHEMICALS, sizeof(bsreal *));
      for (j = 0; j < CHEMICALS; j++) {
        g[i][j] = new bsreal;  h[i][j] = new bsreal;
}}}};
```

5.4.2 評価

評価は、個体の持つS-システムパラメータを式(4.2)に実際にあてはめて微分方程式を解き、それにより得られる時系列データと、最適化条件として与えられている時系列データとの、各サンプリングポイントでの二乗誤差の合計の逆数として計算し、各個体の評価値とすることで行う(式(4.4))。全ての個体の評価値を計算した時点で、もっとも評価値の高い個体をエリート個体、その評価値をエリート評価値と呼ぶ。

微分方程式を解く際の刻み幅は、指定がなければ、与えられる時系列データの平均サンプリング間隔の$\frac{1}{200}$なので、各サンプリングポイントにはその点、あるいはその近傍の点での計算値がある。したがって、補間を用いて、サンプリングポイントにおける近似値を計算することはしていない。また要素によってはサンプリングポイントにデータが与えられないものもあり得るが、その要素のその点での二乗誤差は計算しない。こうすると、サンプリングポイントでのデータが少ない要素では、二乗誤差の合計が小さくなり、評価値がデータの一致、不一致を適切に反映しなくなることがあるが、ここではそういったデータは想定しない。

また式(4.4)にしたがって評価値を計算する際、計算機の演算精度の問題などで与えられる時系列データ$x_{i,t}$が0、または二乗誤差和eが0になると、評価値が計算できない。これを避けるため、除算の計算をするときには、その分母に、評価に影響を与えない程度の微小な値EPS$= 1.0e^{-15}$という値を加え、ゼロ除算による演算エラーを避ける。この値は、計算機の実数の表現精度から決定した。

全ての個体の評価が終わった時点で、最適化を終了すべきか否かを判断する。最適化開始から世代交代毎にエリート評価値を調べ、世代交代を行ってもエリート個体の評価値が変化しなければ、カウントを1増やす。このカウントが、別に指定される最大世代交代回数の1/2に達したら、最適化を終了する。

5.4.3 淘汰

淘汰には、ランキング戦略を改変した方法を用いている。各個体の選択確率は式(4.6)に示すc_iを用い、η^+は1.1に固定した。この確率に従い、個体集団から1つの個体を選び出す。これをP回繰り返し、P個の個体からなる集団を作る。この際、一つの個体が重複して選択されることを許す。こうして生成した集団を親集団と呼び、これに含まれる個体から、交叉により新しい個体を作る。

5.4.4 交叉

親集団に含まれる全ての個体に対し、等確率で二つの個体を選び出す。これから新しい個体を一つ作り、子個体とする。

子個体は一組のS-システムパラメータを持つが、その値は親個体の持つ、対応するパラメータ (たとえば子個体の g_{ij} に対する親個体の g_{ij}) の値とする。どちらの親個体の値とするかは、ランダムに等確率で選ぶ。子個体に含まれる各パラメータに対し、それぞれどちらの親から選ぶかを決定する。

5.4.5 突然変異

交叉により作られた子個体の持つS-システムパラメータに対して、確率 (突然変異率)m でその値を変更する。変更後の値は乱数で決定するが、ここでは二通りの方法を用いた。一つは一様乱数を使う方法であり、S-システムの各パラメータに対して、それぞれに指定されている探索範囲全体にわたって等確率で発生する乱数を用いて、置き換える値を決定する。もう一つは正規分布乱数を使う方法であり、これは変更前のパラメータ値を平均とする正規分布乱数を用いて、置き換える値を決定する。乱数の分散が大きくなれば、これは一様乱数に近づいていき、大域探索となる。分散を小さくしていくと、変更前パラメータ付近の局所探索となる。またどちらの方法でも、突然変異率 m を大きくすると大域探索的に、小さくすると局所探索的になる。

一様乱数を使っている場合に、ある世代数 G_m を決めておき、世代交代を G_m 回行ってもエリート個体の評価値 (以下エリート評価値) が変化しなかった場合には、局所解に捕らわれていると判断し、大域探索を行うために、突然変異率を k 倍する。これによりエリート評価値が変化した場合は、突然変異率はその初期値に戻す。

正規分布乱数を使っている場合には、まず G_m 世代間エリート評価値が変化しなかった場合には、高精度な局所探索を行うために、正規分布の分散を小さくし、エリート個体周辺に探索を集中する。さらに G_m 世代間変化がない場合には、分散を大きくして大域探索に切り替える。分散の拡大、縮小の大きさは探索範囲から経験的に既定しておく。突然変異率は一様乱数の場合と同様に、エリート評価値が G_m 世代間変化がない場合に k 倍される。

5.4.6 骨格構造化

この最適化問題は逆問題であり、同じ挙動を示す多数の数理モデルから、現実に存在する可能性の高い構造のものを選びだすために、骨格構造化を行う。これは、できるだけ簡素な構造のモデルを得るため、既定の閾値以下の値を持つS-システムパラメータについては、その値を0にする操作である。これを既定の数G_sの世代交代毎およびアルゴリズムの終了時に行う。

第6章　検証

6.1　実数の表現形式と、突然変異の方式による違い

遺伝的アルゴリズム (GA) を実際に適用するに当たって、個体表現は非常に重要である。数値最適化に GA を適用する場合、一般的にはビット列を用い、実数値を IEEE 規格などに定められる浮動小数点形式で表現する。この場合、各ビットが遺伝子と見なされ、任意のビット間が交叉点となり得るようにする。しかしこの方法では、二つの実数値を比較したとき、遺伝型における差と表現型における差が著しく異なることがある。つまり遺伝型としては1ビットしか違わないのに実数値としては1桁違う、というようなことである。そこで、探索空間をある間隔の格子で等分し、座標軸と格子の交点に実数値の小さな方から正の整数で番号をつけ、その番号で実数を表す。プログラミングに用いた C 言語ではこの番号は符号なし整数 (unsigned int) で表現しているため、これを符号なし整数表現と呼ぶ。

また、この最適化では局所探索の速さが最適化の速さを決定する。ここでは効率よく局所探索を行うために、突然変異操作に正規分布乱数を導入した。乱数の発生範囲はすなわち探索範囲であるが、一様乱数を用いた場合には、探索範囲全体にわたって乱数が発生し、突然変異は大域探索の働きを行う。しかしここでは、突然変異も局所探索的な探索とするため、親個体の持つ S-システムパラメータの値を平均とする、正規分布乱数を用いる。正規分布の分散を小さく取れば、もとのパラメータ値に近い値が発生しやすくなるため、その近辺の局所探索となる。実際にはエリート個体の周辺を局所探索することになる。

最適化の進行中、世代交代後に全ての個体の評価値を計算した時点で、エリート個体の評価値を世代交代前と比較し、評価値が向上していれば正規分布乱数の分散 d は既定の初期値 d_i に設定する。しかし変化がなく、かつ変化がないまま行われた世代交代の回数が既定の回数 G_m に達したときに、乱数の分散を既定の値 $d_s(d_s < d_i)$ に変更する。

二つの実数表現形式のどちらが、数値最適化に適しているか、また分散を制御された正規分布乱数による突然変異でどの程度の効果があるのかをを検証するため、簡潔な系に対して最適化を行い比較した。方法1として浮動小数点による実数表現、一様乱数を用いた

突然変異による最適化法(いわゆるシンプルGAと呼ばれるアルゴリズム)、方法2として符号なし整数を用いた実数表現で一様乱数を用いた突然変異による最適化法、方法3として符号なし整数と正規分布乱数を導入した方法を、それぞれ図6.1にしめす時系列データに対して適用した。このデータは要素数2、サンプリング点数は50点であり、表6.1に示すS-システムモデルから計算して作成した。最適化条件は表6.2に示す。

評価関数(式6.2)は、各サンプリングポイントにおける相対二乗誤差の総和(式6.1)の逆数とした。逆数を計算する際、0による除算を避けるために、分母に 1.0×10^{-15} を加えた。以下の最適化では評価関数の値はおおよそ 0.5 から 20.0 の間になるため、これに影響がなく、計算機の演算精度から 0 として認識されないできるだけ小さな値として、この値を決定した。また表6.2に示す最大世代交代回数 G_{max} の 1/2 の回数の世代交代を行っても、エリート個体の評価値が変化しなかった場合、または世代交代の回数が G_{max} に達した場合に最適化を終了し、その時点でのエリート個体を解とした。分母に 1.0×10^{-15} を加えること、および最適化を終了する条件は、以下の全ての場合において共通である。

$$e = \sum_{i=1}^{n} \sum_{t=1}^{T} \left(\frac{x_{i,t} - X_{i,t}}{x_{i,t} + \epsilon} \right)^2 \tag{6.1}$$

$$f = \frac{1}{e + \epsilon} \quad (\epsilon = 10^{-15}) \tag{6.2}$$

ここで $x_{i,t}$ は観測値として与えられる時系列データ、$X_{i,t}$ は個体が表現するモデルから計算される時系列データ、i は要素の添字、t はサンプリングポイントの添字である。e が相対二乗誤差和で、f が個体の評価値となる。ϵ は 0 による除算を避けるための微小な値である。

浮動小数点方式では一つの実数を、1桁の符号部、3桁の指数部、4桁の仮数部の合計8桁(1バイト)の2進数で表現した。つまり最適解(絶対値で1.0および4.0、5.0付近)では、実数の表現精度は0.125となり、これよりも小さな違いは表現できない。符号なし整数方式では探索範囲を 2^{32} に等分するが、10進数で表現したときに小数点以下二桁以下は四捨五入して無視することにした。つまり探索精度は0.1である。

浮動小数点方式と符号なし整数方式のそれぞれについて、20回の最適化を行った結果を表6.3に示す。最適化時間に注目すると、符号なし整数方式が優れていると言える。単に最適化法として考えた場合には、大まかに見ると、方法1は長時間かけて高精度な最適化を、方法2は短時間で粗い最適化を、方法3は短時間でその中間の精度の最適化を行っている。いずれの場合も、サンプリングポイント1点あたりの、与えられたデータと最適化されたモデルから計算される時系列データの相対誤差は非常に小さく、十分な最適化が行われていると言える。

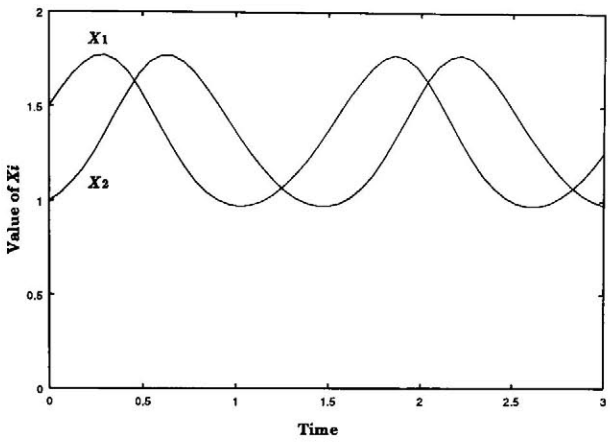

図 6.1: 表 6.1 に示す S-システムパラメータから、微分方程式を解くことによって得られる時系列データのプロット。各要素 X_1、X_2 について、サンプリングポイントは 50 点。

最適化に要した CPU 時間は、浮動小数点方式と符号なし整数方式で分かれていて、符号なし整数を用いると、浮動小数点方式に比べて約 46%最適化時間を短縮できた。得られる個体の評価値をいずれも十分に高いと考えると、最適化時間が短い方が優れた最適化法であると言える。探索点総数は一様乱数の場合と正規分布乱数の場合とで、約 20%ほど異なる。この点で正規分布乱数の方が優れていると言うことができ、つまり符号なし整数と正規分布乱数を用いる最適化が優れた最適化法であると言える。方法 1 と比べて、方法 3 では総探索点数が 2 割しか違わないのに最適化時間が 4 割以上短くなっているのは、方法 3 では発生した個体が表現する S-システムモデルから、時系列データが計算できなかった、つまり微分方程式が数値的に解けなかったことが多かったことを意味する (微分方程式として解けない場合には、その分計算時間が省かれるため)。

S-システムパラメータのうち g_{ij}、 h_{ij} の値は要素間の直接または間接の相互作用の合計の大きさを表すが、これを正、負、ゼロの三種類に分けて考えた場合、それぞれ増加過程 (g_{ij})、減少過程 (h_{ij}) に対して促進 (正)、抑制 (負)、無関係 (ゼロ) という物理的意味を持つ。つまり値がゼロか非ゼロか、および符号が系のおおよその構造を示すと考えられるため、この観点で、最適化によりオリジナルの系の構造が発見できたかどうかを見てみると、表 6.3 に示すように、方法 3 でのみオリジナルの構造 (表 6.1) が見つかった。

表 6.1: 浮動小数点方式と符号なし整数方式の比較に用いた系を表現する S-システムパラメータ。これから微分方程式を解いて時系列データを生成し (図 6.1)、それをもとにこのパラメータセットが最適化により見つかるかどうか、どの程度の速さで見つかるかを検証する。

i	α_i	g_{ij}		β_i	h_{ij}	
1	3.0	0.0	-2.5	3.0	0.125	1.0
2	3.0	2.5	0.0	3.0	0.0	0.125

表 6.2: 実数の表現に浮動小数点方式を用い、突然変異には一様乱数を用いる方法 (方法 1)、実数を符号なし整数で表現し、突然変異には一様乱数を用いる方法 (方法 2)、実数を符号なし整数で表現し、突然変異には分散を制御する正規分布乱数を用いる方法 (方法 3) による、最適化の比較のための最適化条件。

	方法 1	方法 2	方法 3
要素数 (n)	2	2	2
サンプリングポイント数 (T)	50	50	50
探索範囲 (α,β)	[0,20.0]	[0,20.0]	[0, 20.0]
探索範囲 (g_{ij},h_{ij})	[-4.0,4.0]	[-4.0,4.0]	[-4.0,4.0]
探索精度	0.01	0.01	0.01
個体数 (P)	1000	1000	1000
最大世代交代回数 (G_{max})	500	1000	1000
突然変異率初期値 (m_0)	0.004	0.1	0.1
突然変異率更新 (G_m)	20	20	20
正規分布乱数分散初期値 (d_0)	-		$4.0(\alpha, \beta)$
	-		$1.6(g, h)$
分散縮小値 (d_1)	-		$0.5(\alpha, \beta)$
		-	$0.2\ (g, h)$
骨格構造化頻度		5 世代ごと	
骨格構造化閾値 (S)	0.1	0.1	0.1

表 6.3: 実数の表現に浮動小数点方式を用い、突然変異には一様乱数を用いる方法 (方法 1)、実数を符号なし整数で表現し、突然変異には一様乱数を用いる方法 (方法 2)、実数を符号なし整数で表現し、突然変異には分散を制御する正規分布乱数を用いる方法 (方法 3) による、最適化の比較。それぞれ 20 回の試行を行った平均を示す。計測は Tempest 2 ((Concurrent Systems Inc., Japan) (Processor:Alpha 21164A, 600MHz, SPECfp95: 21.3, SPECint95: 18.6)) で行った (以下全て同じ)。

	方法 1	方法 2	方法 3
探索点総数	134000	136000	112000
CPU 時間 (sec.)	92.2	53.7	54.2
平均評価値	1.365	0.7971	1.074
一点あたりの平均相対誤差 ($\times 10^{-2}$%)	3.53	4.62	3.98
オリジナルの構造が見つかった回数	0	0	1

6.2 集団の大きさによる効果

GA による探索では、集団の個体数がすなわち同時に探索される点の個数であり、これを P とし世代交代の回数を G とすると、探索点の総個数は PG となる。個体数が多ければ並列探索的性格が強くなり、大域探索となる。個体数を少なくしていけば局所探索的になっていく。そこで、S-システムで表現された系に対して最適な個体数があるかどうかを検証した。ここで、最終的な探索点数を、個体数が異なっても同じにするため、最大世代交代回数と個体数の積を 1×10^6 の一定値になるようにした。最適化条件を表 6.4 に、第 6.1 章に示す系の時系列データ (図 6.1) を用いて 20 回の最適化を行った結果を表 6.5 に示す。

直感的に、個体数は多いほど同時探索点の個数が多くなり、大域探索的な性質が強くなるため、よりよい最適解を得やすいであろうことが考えられるが、もっとも個体数が多い場合に、もっとも高い評価値が得られている。しかし個体数によってばらつきがあり、一概に個体数が多いほど高い評価値の個体が得られるとは言えない。最適化に要する CPU 時間も同様である。最適化時間を基準に考えると、平均 CPU 時間は個体数 10000 の場合は他の個体数 10 から 1000 の場合の約 3 倍であり、個体数は 1000 以下がよいと考えられる。

表 6.4: 個体数の効果を見るための最適化条件。発生しうる探索点の最大数を個体数 (P) と最大世代交代回数 (G_{max}) の積とし、この値が一定となるようにした。個体数は $P = 10, 100, 1000, 10000$ の 4 通りとした。G_{max} はそれぞれ、$10000, 1000, 100, 10$ とした。

要素数 (n)	2
サンプリングポイント数 (T)	50
探索範囲 (α,β)	[0,20.0]
探索範囲 (g_{ij},h_{ij})	[-4.0,4.0]
探索精度	0.01
個体数 (P) と最大世代交代回数 (G_{max}) の積	1000000
突然変異率初期値 (m_0)	0.1
突然変異率更新 (G_m)	20
正規分布乱数分散初期値 (d_0)	$4.0(\alpha, \beta)$
	$1.6(g, h)$
分散縮小値 (d_1)	$0.5(\alpha, \beta)$
	$0.2\ (g, h)$
骨格構造化頻度	5 世代ごと
骨格構造化閾値 (S)	0.1

表 6.5: 図 6.1 を満たす S-システムパラメータセットを最適化する時の個体数による最適化時間の変化。各個体数での 20 回の最適化の平均を示す。個体数 1000 の場合に示しているのは表 6.3 の方法 3 のものと同じである。

個体数	10	100	1000	10000
平均世代交代回数	84306	9437	686.2	89.70
平均総探索点数	843060	943700	686200	897000
平均 CPU 時間 (sec.)	49.6	69.2	54.2	158.3
平均評価値	1.332	2.075	1.074	2.558
一点あたりの平均相対誤差 ($\times 10^{-2}$%)	3.57	2.86	3.98	2.58
表 6.1 と同じ構造が得られた回数	0	0	1	0

6.3 評価関数による違い

評価関数の形は、GAによる最適化の可否に非常に大きな影響を与える。ここでは与えられた時系列データを再現するモデルを最適化により得ることを目的としているため、与えられたデータとモデルの挙動との差をもとに評価関数を作る必要がある。ここでは三種類の評価関数を考え、比較した。

表6.6に示す相対二乗誤差による結果は、表6.3に示す符号なし整数の場合と同じものである。

6.3.1 重み付き相対二乗誤差

与えられる時系列データを補完して曲線を描いたときに、微分係数がゼロになるような時刻では、定常状態やピークなどの状態であり、過渡的な状態よりもこういった時刻でより正確にデータを再現するようなモデルを得ることが望ましい、との考えから、微分係数の最大値 d_{max} と各時刻での微分係数 d_t の差を d_{max} で除した値 $w_{i,t}$ と、各時刻での与えられる時系列データとモデルが示す挙動との相対二乗誤差の積を合計し、それを重み付き相対二乗誤差 e_w とした。

時刻順に見て t 番目のサンプリングポイントにおいて、その時刻を $TIME_t$、与えられるデータの値あるいは個体から計算される値を $x_{i,t}$ とすると、そのサンプリングポイントに対する重み係数 $w_{i,t}$ は以下のように計算する。

$$\begin{aligned} d_{i,t} &= \frac{x_{i,t+1} - x_{i,t}}{T_{t+1} - T_t} \quad (0 \leq t \leq T-1) \\ d_{i,T} &= d_{i,T-1} \\ d_{max} &= max(d_{i,t}) \quad (0 \leq t \leq T, 0 \leq i \leq n) \\ w_{i,t} &= 1.0 - \frac{d_t}{d_{max}} \end{aligned} \tag{6.3}$$

ここで添字のある T_t はサンプリング時刻を、x_t はそのサンプリング時刻における観測データあるいは個体から計算された値を表す。これを用いて、重み付き相対二乗誤差 e_w と、評価関数 f は以下のように計算する。

$$\begin{aligned} e_w &= \sum_{i=1}^{n} \sum_{t=0}^{T-1} w_{i,t} \left(\frac{x_{i,t} - X_{i,t}}{x_{i,t} + \epsilon} \right)^2 \\ f &= \frac{1}{e_w + \epsilon} \end{aligned} \tag{6.4}$$

分数を計算する際には、ゼロ除算を避けるため、分母に $\epsilon = 10e^{-15}$ を加える。

図6.1に示す時系列データに基づいて、最適化を行った結果を表6.6に示す。評価関数に式(6.4)を用いたこと以外の最適化条件は、全て表6.2に示されている通りである。

6.3.2 微分係数の相対二乗誤差

定常状態そのものよりも、いつ増加、減少するかというタイミングが重要なのではないかという考えから、時刻 T_t における時系列データの微分係数 $d_{e,t}$ とモデルが再現する挙動の微分係数 $d_{c,t}$ の相対二乗誤差に基づいて評価関数を設定した。

微分係数は、まず与えられる時系列データの初期値を用いて連立微分方程式を解き、その結果から二次式補間を用いて計算した。

データの連続するサンプリングポイント $t-1$、t、$t+1$ の三点において、個体から計算された時系列データ $X_{i,t-1}$、$X_{i,t}$、$X_{i,t+1}$ があるとするとき、この三点を補間する二次方程式

$$aT_t^2 + bT_t + c = X_{i,t} \tag{6.5}$$

の係数 a、b、c は、クラメルの公式を用いて

$$a = \frac{\begin{vmatrix} X_{i,t-1} & T_{i-1} & 1 \\ X_{i,t} & T_{i,t} & 1 \\ X_{i,t+1} & T_{i+1} & 1 \end{vmatrix}}{\begin{vmatrix} T_{i,t-1}^2 & T_{i-1} & 1 \\ T_{i,t}^2 & T_{i,t} & 1 \\ T_{i,t+1}^2 & T_{i+1} & 1 \end{vmatrix}}$$

$$b = \frac{\begin{vmatrix} T_{i,t-1}^2 & X_{i-1} & 1 \\ T_{i,t}^2 & X_{i,t} & 1 \\ T_{i,t+1}^2 & X_{i+1} & 1 \end{vmatrix}}{\begin{vmatrix} T_{i,t-1}^2 & T_{i-1} & 1 \\ T_{i,t}^2 & T_{i,t} & 1 \\ T_{i,t+1}^2 & T_{i+1} & 1 \end{vmatrix}}$$

$$c = \frac{\begin{vmatrix} T_{i,t-1}^2 & T_{i-1} & X_{i.t-1} \\ T_{i,t}^2 & T_{i,t} & X_{i,t} \\ T_{i,t+1}^2 & T_{i+1} & X_{i,t+1} \end{vmatrix}}{\begin{vmatrix} T_{i,t-1}^2 & T_{i-1} & 1 \\ T_{i,t}^2 & T_{i,t} & 1 \\ T_{i,t+1}^2 & T_{i+1} & 1 \end{vmatrix}} \tag{6.6}$$

と計算される。二次方程式(式(6.5))を時間で微分すると、微分係数$d_{e,t}$が得られる。

$$d_{e,t} = aT_{i,t} + b \tag{6.7}$$

与えられた時系列データと、個体から計算されるデータの両方に対して式6.7から、各サンプリングポイントにおける微分係数を計算し、その二乗誤差の合計の逆数を個体の評価値とする。与えられた時系列データの微分係数を$d_{e,i,t}$、個体から計算された時系列データの微分係数を$d_{c,i,t}$とすると、個体の評価値fは

$$\begin{aligned} e_d &= \sum_{i=1}^{n}\sum_{t=0}^{T-1}\left(\frac{d_{e,i,t} - d_{c,i,t}}{d_{e,i,t} + \epsilon}\right)^2 \\ f &= \frac{1}{e_d + \epsilon} \quad (\epsilon = 10^{-15}) \end{aligned} \tag{6.8}$$

として計算する。

図6.1に示す時系列データに基づいて最適化を行った結果を、重み付き相対二乗誤差を用いた場合の結果とともに表6.6に示す。評価関数に式(6.4)を用いたこと以外の最適化条件は、全て表6.2に示されている通りである。

結果を見ると、重み付き相対二乗誤差の場合も、微分係数の二乗誤差の場合も、時系列データの相対二乗誤差を用いた場合に比べ最適化時間はあまり変わらい。しかし得られる個体から計算される時系列データは微分係数を用いた場合に、与えられる時系列データと大きく異なっている。微分係数の二乗誤差を用いて、相対誤差の小さな個体を得るのは難しい。

6.4 骨格構造化の閾値

できるだけ簡素なモデルを得るための骨格構造化であるが、その閾値は経験的に決定するしかない。ここでは、骨格構造化の閾値をさまざまに変化させることで、最適化の速

表 6.6: 重み付き相対二乗誤差に基づいた評価関数 (式 6.4)、および微分係数の相対二乗誤差を用いた評価関数 (式 6.8) を用いた最適化の結果。相対二乗誤差を用いた場合の結果は、表 6.3 の方法 3 のものと同じである。

評価関数	相対二乗誤差	重み付き	微分係数
平均世代交代回数	686.2	701.6	708.4
平均総探索点数	686200	701600	708400
平均 CPU 時間 (sec.)	54.2	53.7	57.2
平均評価値	1.074	1.354	0.01175
一点あたりの平均相対誤差 ($\times 10^{-2}$%)	3.98	1.42	12.7
表 6.1 と同じ構造が得られた回数	1	0	0

度や精度、得られるモデルの構造などがどうなるかを検証した。図 6.1 に示す時系列データに基づいて、表 6.7 の条件で最適化を行った結果を表 6.8 に示す。閾値を 0.125 以上にした場合には、表 6.1 に示すオリジナルのモデルは得られないが、ここではオリジナルと同じ構造を持つものを探索する可能性を検証するため、閾値を高く設定した場合についても検証した。

いずれの場合も、最適化の速度、精度ともにほとんど変化がない。閾値を大きくするということはすなわち、パラメータの取りうる値の範囲が狭くなるということであり、探索空間が小さくなることになる。しかし探索の速度、精度ともにあまり変わらないということは、エリート個体周辺の局所探索に探索点が集中していて、探索範囲の広さそのものは影響が少ないことを表している。また閾値が 0.1 の場合にはオリジナルの構造を持つモデルが見つかったが、そうでない場合には、オリジナルの構造は見つかっていない。

6.5 データセット数による効果

表 6.1 に示す S-システムパラメータから、微分方程式を解く際に初期値を変えることにより 4 種類のデータセットを得た (図 6.2)。より多くのデータを与えることは、探索において拘束条件となり探索空間を狭くする効果があると考え、与えるデータセットの数の違いによる効果をみた。結果を表 6.10 に示す。評価関数には、時系列データそのものの相対二乗誤差和を用いた。データセットはそれぞれ、サンプリングポイント数は 50 点で同じであり、各状態変数の値もおおよそ同じ範囲 (0.0〜2.0) に入るようにした。

全ての拘束条件を満たす (全ての与えられる時系列データに完全に一致する) 個体は、一

表 6.7: 骨格構造化の閾値による最適化の様子の変化を見るための最適化条件。骨格構造化の閾値だけを変えて、4つの条件で最適化を行った。

要素数 (n)	2
サンプリングポイント数 (T)	50
探索範囲 (α,β)	[0,20.0]
探索範囲 (g_{ij},h_{ij})	[-4.0,4.0]
探索精度	0.01
個体数 (P)	1000
最大世代交代回数 (G_{max})	1000
突然変異率初期値 (m_0)	0.1
突然変異率更新 (G_m)	20
正規分布乱数分散初期値 (d_0)	4.0(α, β)
	1.6(g, h)
分散縮小値 (d_1)	0.5(α, β)
	0.2 (g, h)
骨格構造化頻度	5 世代ごと
骨格構造化閾値 (S)	0.0, 0.1, 0.5, 1.0

表 6.8: 骨格構造化の閾値による最適化の様子の変化。閾値 0.1 の場合の数値は、表 6.3 の方法 3 のものと同じである。

骨格構造化の閾値	0.0	0.1	0.5	1.0
平均世代交代回数	727.8	686.2	751.0	690.5
平均総探索点数	727800	686200	751000	690500
平均 CPU 時間 (sec.)	57.2	54.2	59.4	54.1
平均評価値	1.339	1.074	1.278	1.190
一点あたりの平均相対誤差 ($\times 10^{-2}$%)	3.57	3.98	3.65	3.78
表 6.1 と同じ構造が得られた回数	0	1	0	0

表 6.9: データセット数を変えて行った最適化の、最適化条件。データセット数が 1,2,3,4 のそれぞれの場合について最適化を行った。

要素数 (n)	2
サンプリングポイント数 (T)	50
データセット数	1, 2, 3, 4
探索範囲 (α,β)	[0,20.0]
探索範囲 (g_{ij},h_{ij})	[-4.0,4.0]
探索精度	0.01
個体数 (P)	1000
最大世代交代回数 (G_{max})	1000
突然変異率初期値 (m_0)	0.1
突然変異率更新 (G_m)	20
正規分布乱数分散初期値 (d_0)	4.0(α, β)
	1.6(g, h)
分散縮小値 (d_1)	0.5(α, β)
	0.2 (g, h)
骨格構造化頻度	5 世代ごと
骨格構造化閾値 (S)	0.1

つの条件 (時系列データ) だけを満たす個体よりも少ないと考えられるため、与えるデータセット数を増やすと、評価値の高い個体は見つかりにくくなると考えられるが、かならずしもそうとは限らないという結果になった (表 6.10)。得られる個体の評価値の平均には、影響は小さいと考えられる。総探索点数にも影響は少ない。しかし、得られた個体の構造をみると、オリジナルのモデルと同じ構造を得る確率が非常に高くなっており、これはデータセット数を増やすほどよいと考えられる。これは生体内反応系の構造最適化を考える上で、非常に重要な特徴である。データセット数が 3 および 4 の場合、いずれの場合でもオリジナルのモデル (表 6.1 に示す S-システムパラメータ) と全く同じモデルが 20 回の最適化のうちの 3 回で得られた。

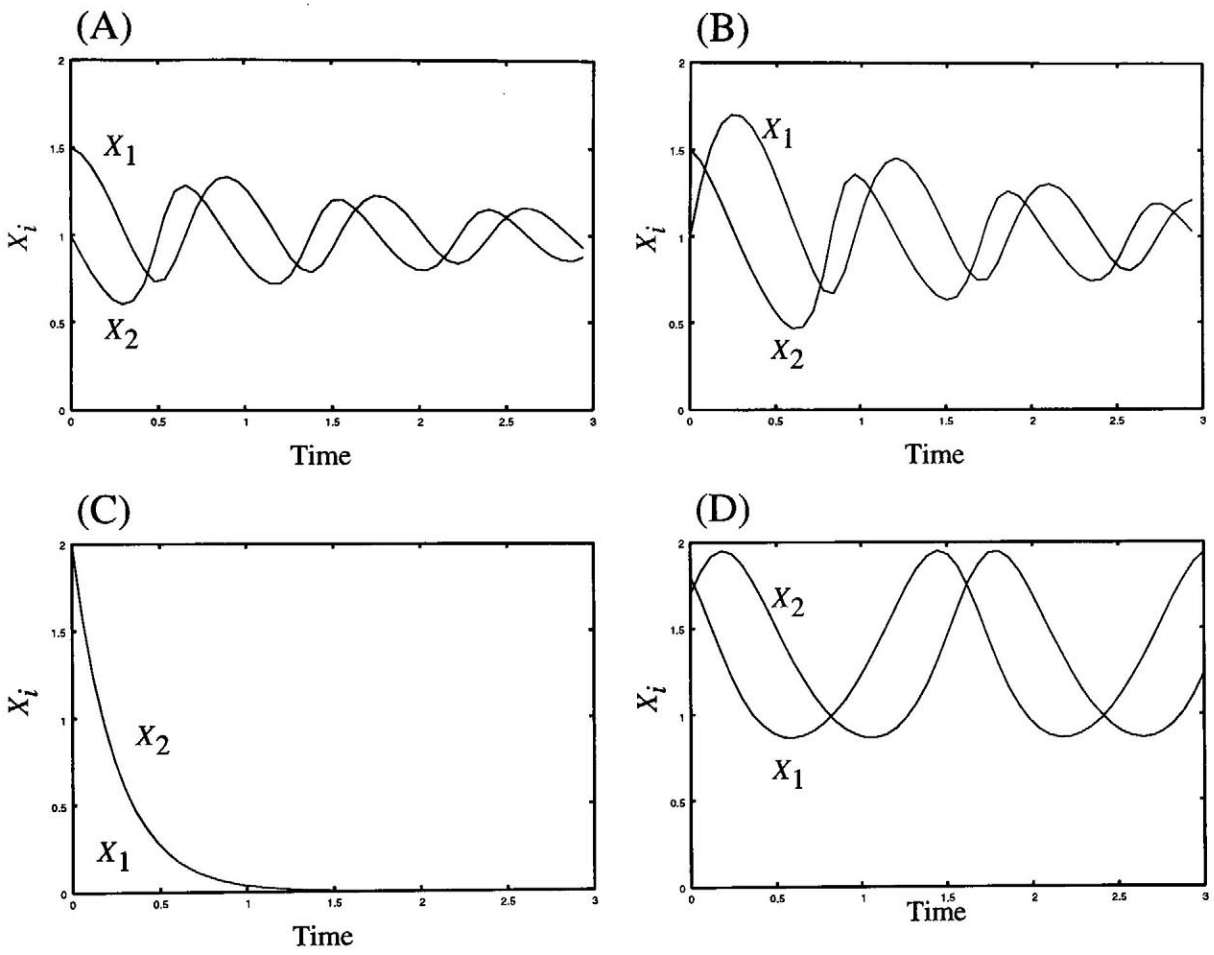

図 6.2: 表 6.1 から初期値を変えて計算される 4つつのデータセット。それぞれ要素数は 2、サンプリングポイント数は 50 である。初期値はそれぞれ、(A)$X_1 = 1.5, X_2 = 1.0$; (B)$X_1 = 1.0, X_2 = 1.5$; (C)$X_1 = 0.0, X_2 = 2.0$; (D)$X_1 = 1.8, X_2 = 1.7$; である。(C) における X_1 は、数値計算上のエラーを避けるために初期値と $1.0e^{-14}$ としており、$t = 0$ から $t = 3.0$ までの間に、$3.60e^{-10}$ まで単調増加している。

表 6.10: 与えるデータ数による効果。データセット数1の場合は、表6.3の方法3のものと同じである。データセット数1の場合は図6.2の (A) に示す時系列データを、データセット数2の場合は図6.2の (A) および (B) に示す時系列データを、データセット数3の場合は図6.2の (A)、(B) および (C) に示す時系列データを最適化アルゴリズムに与えた。

データセット数	1	2	3	4
平均世代交代回数	686.2	700.4	497.5	731.7
平均総探索点数	686200	700400	497500	731700
平均 CPU 時間 (sec.)	54.2	99.6	101.4	203.1
平均評価値	1.074	0.4341	0.08096	0.02566
一点あたりの平均相対誤差 ($\times 10^{-2}$%)	3.98	3.13	4.83	6.44
表6.1と同じ構造のモデルが得られた回数	1	1	3	3

6.6 複数の集団による最適化

最適化をさらに高速に行うための並列処理を想定して、複数の最適化を同時に行い、それにより得られる複数の最適解を統合して最終的な最適化を行う手法を考た。つまり、個体数の少ない小集団での最適化を複数回行い、これにより得られる複数の最適解を初期集団に含む小集団を生成し、これによる最適化で得られる最適解を、最終的な最適解とするアルゴリズムである。複数の小集団による最適化は、GA では島モデル [19][31] と呼ばれる方法である。島モデルでは一般に小集団間の個体の交換を行うこと (移住) で集団中の個体の多様性を保つが、ここでは、各小集団による探索は局所探索性が強く、それだけ短時間で最適化を行うことができることから、移住により集団内の個体の多様性を保つことは、局所探索の進行の妨げになると考え、移住は行わない。

最適化に用いる時系列データは表6.11に示すパラメータで定義されるS-システムモデルを用いて作成した。それをプロットした図を図6.3(A)に示す。また最適化条件を表6.9に示す。複数の小集団では、各小集団の個体数は100で、集団の総数を10とした。このうち9の集団で独立に最適化を行い、得られた9個の最適解と、ランダムに生成した91個の個体で100個の個体からなる集団を作り、これを用いて最適化を行い、最終的な最適解を得る。小集団による最適化は、合計10回行われることになる。各最適化で、最大世代交代回数は10000とし、最大総探索点数は、個体数 $P\times$ 最大世代交代回数 $G_{max}\times$ 集団数 $= 1000000(10^7)$ である。これは一つの集団による最適化の場合 ($P \times G_{max} = 10000 \times 1000 = 10^7$) と同じである。

これにより得られた結果を表6.13に示す。表6.13中の方法2は複数の集団による最適

表 6.11: 一つの集団による最適化と複数の集団による最適化を比較するために用いた時系列データ (図 6.3 に示す) を生成するのに用いた、S-システムモデルを定義するパラメータ。

i	α_i	g_{ij}		β_i	h_{ij}	
1	3.0	0.0	-2.5	3.0	-1.0	0.0
2	3.0	2.5	0.0	3.0	0.0	2.0

表 6.12: 一つの集団による最適化と複数の小集団による最適化を比較するための最適化条件。個体数と最大世代交代回数にある数値はそれぞれ、一つの集団による最適化、および複数の小集団による最適化の順になっている。骨格構造化は、世代交代が行われる度に行う (各個体の評価値を計算する直前に行う)。

要素数 (n)	2
サンプリングポイント数 (T)	50
データセット数	1, 2, 3, 4
探索範囲 (α,β)	[0, 5.0]
探索範囲 (g_{ij},h_{ij})	[-3.0,3.0]
探索精度	0.01
個体数 (P)	10000 および 100
最大世代交代回数 (G_{max})	1000 および 10000
突然変異率初期値 (m_0)	0.05
突然変異率更新 (G_m)	20
正規分布乱数分散初期値 (d_0)	$4.0(\alpha, \beta)$
	$1.6(g, h)$
分散縮小値 (d_1)	$0.5(\alpha, \beta)$
	$0.2\ (g, h)$
骨格構造化頻度	1 世代ごと
骨格構造化閾値 (S)	0.5

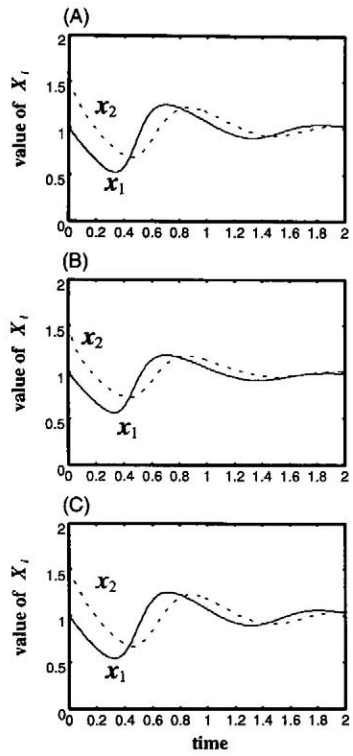

図 6.3: (A) 表 6.11 に示すパラメータで定義される S-システムモデルから、微分方程式を解くことにより得られる時系列データのプロット、(B) 表 6.13 に示す方法 1 により得られた個体から計算される時系列データ、(C) 表 6.13 に示す方法 2 により得られた個体から計算される時系列データ。

表 6.13: 一つの集団による最適化と複数の小集団による最適化を比較。方法 1 は一つの集団による最適化、方法 2 は 10 個の小集団による最適化である。方法 2 の世代交代回数は、小集団による各最適化の平均値であり、総探索点数は平均世代交代回数 (5519×) 各集団の個体数 (100×) 集団の数である (10)。

方法	方法 1	方法 2
平均世代交代回数	1000	5519(1 小集団あたり)
総探索点数	10^7	5.5×10^6
CPU 時間 (sec.)	58803	31677
評価値	7.50	51.6
一点あたりの平均相対誤差 ($\times 10^{-2}$%)	3.65	1.39
表 6.1 と同じ構造が得られた回数	0	1

表 6.14: 一つの集団による最適化と複数の小集団による最適化を比較。それぞれ 20 回の試行による結果の平均を示す。

方法	方法 1	方法 2
評価値	2.86	17.55
CPU 時間 (sec.)	58392	36459

化であるが、各小集団による最適化は、表 6.5 に示す個体数 100 の場合と、時系列データが異なること以外は同じ条件である。表 6.13 に示す、複数の小集団による最適化に要した CPU 時間は、小集団による 10 回の最適化それぞれに要した CPU 時間の合計である。つまり複数の小集団を用いることで、約 46%程度短縮することができたことになる。小集団による各最適化に要した、平均の CPU 時間は約 3168 秒になり、これは表 6.5 に示す個体数 100 の場合の 69.2 と比べると非常に大きく異なっているが、これはコンパイラが異なることが原因の一つである。サンプリングポイント 1 点あたりの、与えられた時系列データと最適化されたモデルから計算される時系列データの相対誤差も、複数の小集団による方が小さくできた。また複数の小集団を用いた最適化では、オリジナルの構造を発見することができた。

また、20 回の試行による結果を表 6.14 に示す。

6.7 遺伝子ネットワーク

以上の結果をもとに、典型的な形式の遺伝子ネットワークに対して最適化を行った。図6.4に示す構造の遺伝子ネットワークを表6.15のS-システムパラメータで記述し、図6.5に示す時系列データを算出した。これは、いわゆる遺伝子ネットワーク(遺伝子間の制御構造を記述するネットワーク)とは異なり、遺伝子のみではなく、その遺伝子から作られるタンパク質、そのタンパク質が生成、分解を制御する別のタンパク質が含まれている。遺伝子が2種類、タンパク質が3種類あり、系を構成する要素は合計5種類であり、したがって状態変数の個数も5である。ここでは各状態変数を各物質の量として X_1 から X_5 で表している。酵素 X_2 は遺伝子のmRNA(X_1)から作られ、X_2 は制御酵素 X_3 の生成と分解をともに抑制する。また別の制御酵素 X_5 はその遺伝子のmRNA(X_4)から作られる。X_3 は遺伝子 X_1 および X_4 の発現をともに促進し、X_5 はどちらの発現も抑制する。

ここで、遺伝子の発現量はその遺伝子の情報を写し取ったmRNAの量で測定されるという実験的な条件を考慮に入れ、遺伝子の発現量を示すmRNAの量を状態変数とした。S-システムモデルを用いるということは、個々のタンパク質について、どのような化学反応で生成されるかは考慮せず、その生成、分解を制御するのはどの要素であるかということだけを解析の対象として捉えるということなる。またここでは遺伝子ネットワークの解析という考えから、各mRNAの発現量を増加させようとする作用のみを解析対象とした。つまりS-システムパラメータのうち、$i=1$ および $i=4$ に対する α_i と g_{ij} のみを最適化の対象とした。この場合、最適化されるS-システムパラメータの総数は12個であり、前節までの要素数が2の系の最適化の例と同じである。図6.5に示す時系列データは、表6.15のS-システムパラメータに対して、各状態変数の初期値は等しいまま、一つは表6.15のパラメータ値をそのまま用いて、一つは表6.15のパラメータ値のうち α_1 を0.0としたモデル、一つは表6.15のパラメータ値のうち α_4 を0.0としたモデルで微分方程式を解いて得られたものである。これは、遺伝子ネットワークの構造を調べるために行われる実験を想定したもので、$\alpha_1=0.0$ とするのは、遺伝子の発現を強制的に抑制することで、mRNAが作られず、X_1 が増加することがないことを表す。同様に $\alpha_4=0.0$ はmRNAの X_4 が作られないことを表す。したがって図6.5では、対応する状態変数の値が時間にともなって単調減少している。現在はmRNAの発現量を直接得ることは難しいが、近年の実験技術の急激な進歩により、近い将来可能になると考えられている。

以上の考えのもと、表6.16に示す条件下で最適化を行った結果を表6.17に、これは複数の試行の平均ではなく、最適化は1回だけ行った。GAでの世代交代を266回行った段階で、表6.15に示すS-システムモデルが得られた。つまり、最適化は極小点のうちの最小点を探索することができ、また逆問題をも解くことができたといえる。またモデルの構造に着目すると、第78世代で表6.17に示すのS-システムパラメータを得ることができ

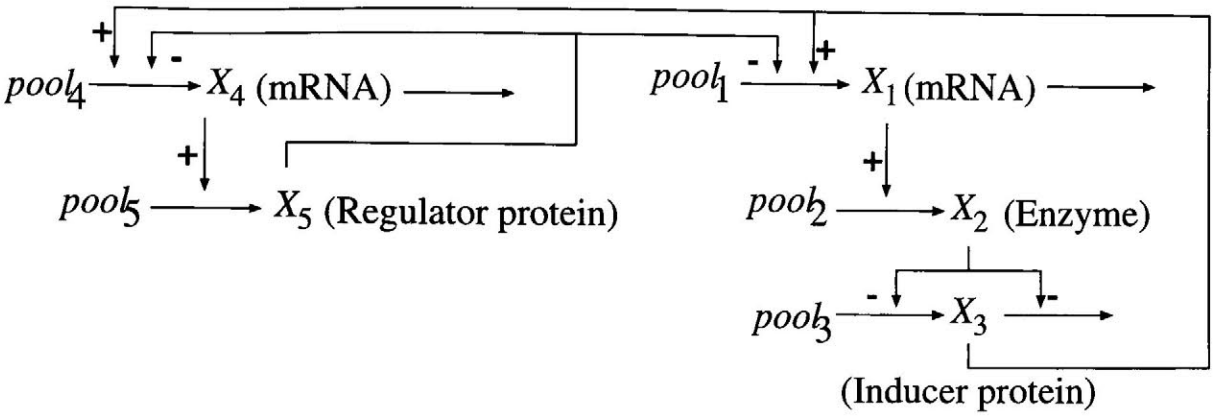

図 6.4: 模式的な遺伝子ネットワーク [18]。

表 6.15: 図 6.4 に示す遺伝子ネットワークを記述する S-システムパラメータ。

i	α_i	g_{i1}	g_{i2}	g_{i3}	g_{i4}	g_{i5}	β_i	h_{i1}	h_{i2}	h_{i3}	h_{i4}	h_{i5}
1	15.0	0.0	0.0	1.0	0.0	-0.1	10.0	2.0	0.0	0.0	0.0	0.0
2	10.0	2.0	0.0	0.0	0.0	0.0	10.0	0.0	2.0	0.0	0.0	0.0
3	10.0	0.0	-0.1	0.0	0.0	0.0	10.0	0.0	-0.1	2.0	0.0	0.0
4	8.0	0.0	0.0	2.0	0.0	-1.0	10.0	0.0	0.0	0.0	2.0	0.0
5	10.0	0.0	0.0	0.0	2.0	0.0	10.0	0.0	0.0	0.0	0.0	2.0

た。構造の発見は非常に早い段階で行われたと考えられる。

次に、遺伝子ネットワークの最適化を、小集団による複数の最適化を用いて行った。一つの個体集団による最適化とこれを複数に分けた最適化を比較するため、双方で総探索点数の上限が等しくなるように個体数と最大世代交代回数を設定した。一つの集団による最適化では個体数を 5000、最大世代交代回数を 400 とし、その積は 2000000($=2\times10^6$) である。複数の集団による最適化では、個体数 100 の 10 個の小集団を生成し、そのうちの 9 個の集団で独立に (同時に) 最適化し、その終了後、それぞれの集団で得られた 9 個の最適解を残った 1 個の集団に入れ、最終的な最適化を行い、これにより得られた解を最終的な最適解とする。合計 10 回の最適化が行われるが、その全てで最大世代交代回数は 2000 に固定しておく。各小集団による最適化での総探索点数の最大値は $100\times2000=200000(=2\times10^5)$ であり、小集団による最適化は 10 回行われるため、総合的な総探索点数の最大値はこの 10 倍、2000000($=2\times10^6$) となる。この最適化法による結果を表 6.19 に示す。それぞれ

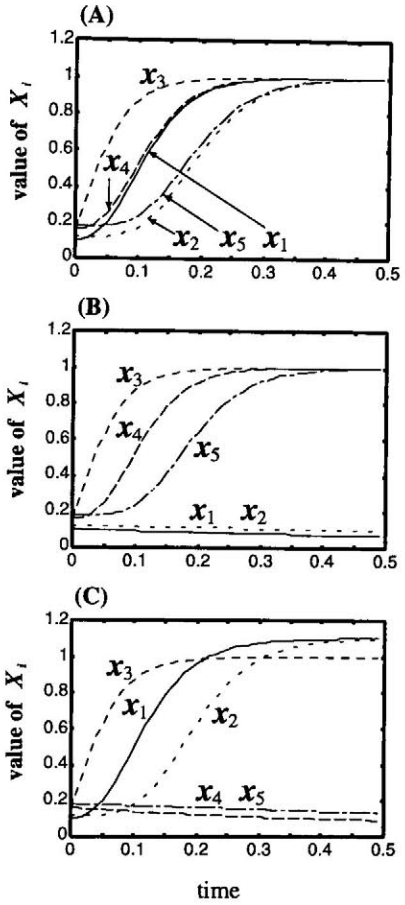

図 6.5: 表6.15 に示すS-システムパラメータから計算される時系列データ。(A) 表6.15 のパラメータから計算される時系列データ。(B) 表6.15 のパラメータのうち、α_1 を 0.0 に設定して計算される時系列データ。(C) 表6.15 のパラメータのうち、α_4 を 0.0 に設定して計算される時系列データ。いずれの場合も、各状態変数の初期値は、$X_1 = 0.10$; $X_2 = 0.12$; $X_3 = 0.14$; $X_4 = 0.16$; $X_5 = 0.18$ である。

表 6.16: 遺伝子ネットワークのための最適化条件。

要素数	5
サンプリングポイント数	50
サンプリング時間	[0, 0.5]
データセット数	3
α_i 及び β_i に対する探索範囲	[0, 20.0]
g_{ij} 及び h_{ij} に対する探索範囲	[-3.0, 3.0]
一世代あたりの個体数	5000
突然変異率を変化させるまでの世代交代回数	2
骨格構造化閾値	0.05
骨格構造化回数	1 回の世代交代につき 1 回
最大世代交代回数	400
使用した乱数	正規分布乱数

表 6.17: 遺伝子ネットワークの最適化。1 回の最適化について、途中の第 78 世代での値と最適化が終了した第 266 世代での値を示す。第 78 世代で得られた S-システムパラメータの値は表 6.18 に示す。第 266 世代で得られた個体の S-システムパラメータは完全に表 6.15 に示すものと一致しているため、相対二乗誤差は 0 である。

世代交代数	78	266
データセット数	3	3
総探索点数	390000	1330000
CPU 時間 (sec.)	11544	39368
1 点あたりの相対誤差	3.54×10^{-4}	0

表 6.18: 表 6.17 に示す 78 回の世代交代で得られたモデルの S-システムパラメータ。各パラメータについて、その符号およびゼロか非ゼロかは、表 6.15 に示すオリジナルのモデルと一致している。

i	α_i	g_{i1}	g_{i2}	g_{i3}	g_{i4}	g_{i5}	β_i	h_{i1}	h_{i2}	h_{i3}	h_{i4}	h_{i5}
1	16.28	0.0	0.0	1.07	0.0	-0.08	10.0	2.0	0.0	0.0	0.0	0.0
2	10.0	2.0	0.0	0.0	0.0	0.0	10.0	0.0	2.0	0.0	0.0	0.0
3	10.0	0.0	-0.1	0.0	0.0	0.0	10.0	0.0	-0.1	2.0	0.0	0.0
4	8.55	0.0	0.0	2.09	0.0	-1.01	10.0	0.0	0.0	0.0	2.0	0.0
5	10.0	0.0	0.0	0.0	2.0	0.0	10.0	0.0	0.0	0.0	0.0	2.0

表 6.19: 一つの集団による最適化と、10 個の小集団による最適化の比較。方法 1 は表 6.16 に示す条件での、一つの集団による最適化の結果である。方法 2 は 10 個の小集団による最適化で、個体数が 100 であること、最大世代交代回数が 2000 であること以外の最適化条件は表 6.16 に示す通りである。方法 2 での CPU 時間は、10 個の最適化に要した CPU 時間の合計である。それぞれ 20 回の試行の平均である。

	方法 1	方法 2
平均 CPU 時間 (sec.)	1096	327.6
平均評価値	1.081	1.028
一点あたりの平均相対誤差 ($\times 10^{-2}$%)	6.05	6.36
平均総探索点数	336222	80006

20 回の試行の平均であり、個体数と最大世代交代回数以外の最適化条件は表 6.16 に示す通りである。

小集団による最適化では、10 個の小集団による最適化に要した CPU 時間の合計が、一つの集団での最適化に比べると約 1/3 になっており、総探索点数では約 1/4 になっている。9 個の集団による最適化を並列で行わなくても、約 3 倍の高速化ができている。

第7章　結論

ここでは、2個の要素が相互作用を行うネットワークと、典型的な構造の遺伝子ネットワークに対して、S-システムで記述された数理モデルから時系列データを計算し、その時系列データをもとに構造推定を行った。前者では与えられる時系列データとの相対二乗誤差の和が同程度のモデルが多数得られることがあり、この問題が逆問題であることが示された。これに対して、遺伝的アルゴリズムを改良したアルゴリズムにより、もとの数理モデルと同じ構造を持つモデルを得ることができた。これにより、逆問題を解決するアルゴリズムを開発した、ということができる。遺伝子ネットワークに対しても、同様にもとの数理モデルと同じモデルを得ることができた。

一般的に用いられる、単純な遺伝的アルゴリズム (Simple GA、SGA) に対する、ここで行った改良についてその効果を考えてみる。

まず実数の表現形式であるが、最適化に要する総探索点数は一世代あたりの個体数と世代交代回数の積で求められ、これは浮動小数点方式と符号なし整数方式とでは、符号なし整数の方が少ない探索点で最適化を行えることが示された。また得られる個体から計算される時系列データと、与えられるデータとの相対二乗誤差の平均は、おおよそ1 : 1.3で、他の例での評価値のばらつきを考えると、これは差はあまりないと言える。これらの表現方法による差は小さいが、しかしこれは、系が大きくなり (要素数の大きな系、つまり最適化するS-システムパラメータの個数が多い最適化)、かつ探索がさらに局所探索的になると差は大きくなると思われる。つまり極小点の近傍に個体が集中してきた場合、浮動小数点方式では第4.1章に述べるハミング距離とユークリッド距離の差が大きくなり、交叉によって極小点への収束することが期待できなくなる。つまり交叉がユークリッド空間での局所探索にならず、大域探索的になることになる。したがって局所探索は突然変異のみによることになり、極小点への収束は遅くなる。これは要素数が大きな系を対象とした場合、特に顕著になってくると思われる。要素数が多くなるにつれ、探索空間は爆発的に大きくなる。しかしGAにおける個体数をそれにしたがって爆発的に大きくするわけにはいかないため、探索空間の大きさに対する探索点の個数の割合は、小さくなっていくことになる。これはすなわち、系が大きくなるにつれ、次第に局所探索的にならざるを得ないということである。遺伝子ネットワークなどの生体内反応系は巨大であるため、現実の実験

データから最適化を行おうとするときには、これは致命的な欠点になる可能性がある。そのため、生体内反応系に対する最適化には、符号なし整数方式がより適していると考えられる。

符号なし整数を実数の表現法として用い、さらに正規分布乱数を導入した場合、一様乱数を用いた場合と比べ、得られる個体から計算される時系列データと、与えられるデータとの相対二乗誤差の平均は $1:0.861$ で、これもあまり差はないと言える。また総探索点数の比も $1:1.01$ であり、差は小さい。この場合の特筆すべき差は、オリジナルの構造が探索されたかどうかである。方法3では20回の試行により、オリジナルのモデルの構造を見つけることができた。これはこのアルゴリズムが逆問題に対する有力であることを示す。

次にこの場合の、探索空間の大きさと、その中で実際に探索した領域の大きさの比を考えてみる。$alpha_i$ と β_i の探索範囲はどちらも $[0,20]$ で、探索精度を 0.01 としている。したがって一つの $alpha_i$ または β_i に対して、$20/0.01=2000$ 点の探索点があり、これが一つの $alpha_i$ または β_i に対する探索空間の大きさである。そして $alpha_i$ と β_i の総数は4個なので、全ての $alpha_i$ と β_i に対する探索空間の大きさは $2000^4=2^4\times10^{12}$ 点である。同様に g_{ij} および h_{ij} の探索範囲 $[-4,4]$ で、各パラメータに対する探索点は800点、g_{ij} および h_{ij} の総数は8個なので、g_{ij} および h_{ij} に対する探索空間の大きさは $800^8=2^{24}\times10^{16}$ となる。したがって、最適化対象となる探索空間全体の大きさはこれらの積 $2^4\times10^{12}\times2^{24}\times10^{16}=2^{28}\times10^{28}$ となる。これに対して探索した点の総数は、GAで生成した個体の総数であるが、これは集団の個体数と世代交代の回数の積である。表6.3に示すように、この値はおおよそ $1.1\sim1.4\times10^5$ であり、この表の方法3の場合では 1.12×10^5 である。これらの比、つまり全探索空間の大きさに対する、探索した空間の大きさの比は、$=2^{28}\times10^{28}:1.12\times10^5\approx6.3\times10^{30}:1$ である。広大な探索空間の中の、ほんの一部の領域のみを探索することで、オリジナルの構造を発見することができたと言える。

評価関数による違いは、時系列データの相対二乗誤差と、それ以外の重み付き相対二乗誤差と微分係数の相対二乗誤差とで、非常に大きな差があり、重み付き相対二乗誤差と微分係数の相対二乗誤差はこの場合には効果がないという結果であった。最適化の目的は、時系列データを再現するモデルの探索であり、最適化により得られたモデルについては、時系列データの再現性を問題とするため、これを直接に表現しない評価関数では、うまく最適化できないということである。

次に、骨格構造化の効果を調べるため、その閾値による影響を調べた。閾値の大きさは、得られる個体の評価値には影響が小さい。つまりこれは、局所解の近辺に存在しているであろうエリート個体の、ごく近傍だけを探索するため、骨格構造化によって探索範囲の境界が発生しても、その境界が実際に探索されている領域に関わりにくいということを

示している。

与えるデータ数を増やした場合には、それによりオリジナルのモデルの構造を10%以上の確率で得ることができた。逆問題に対するアプローチとしては、効果的である。ある1組の時系列データに対して、それと一致する挙動を示すモデルは複数存在する。しかし一致すべき時系列データの組が複数になると、その全てに一致する挙動を示すモデルの数は減少する。時系列データの組数が変化したときに、その全てに一致するモデルの個数がどうなるかは問題によって変化し、見積もりは困難であるが、データの組数の増大にともない、全てを満たすモデルの個数が単調減少することは予想される。データ組数が増えた結果、それら全てを満たすモデルがただ一つになれば、これは逆問題ではない。ここでの逆問題は、挙動が一致するモデルが複数存在し、二乗誤差のみを用いた最適化では複数の最適解のうち、どれがもっとも望ましいものであるかを判定できないことである。しかし評価関数が二乗誤差のままでも、データ組数を増やすことで最適解の個数を減らし、逆問題的な性質を減らすことができることを、ここでの結果は示している。

集団の個数を複数とした場合、最適化時間に大きな向上が見られた。複数の集団を用いることで総探索点数およびCPU時間はおおよそ1/2になった。また評価値もより高い解が得られた。前述したようにここでは並列処理を想定して集団を複数としたが、複数、ここでは10個の小集団による最適化のうち、9個の小集団はそれぞれ全く独立に最適化した。これは計算機が複数ある場合に、それぞれ同時に最適化することができるということであり、そうした場合には、ここでは9回連続して行った最適化を、1回分の時間で行うことができる。そしてその後に1回の最適化を行うため、最適化の実時間は2回分の時間で済むことになる。ここで示した結果では、最適化1回あたりの平均CPU時間は約3168秒であり、2回分では6336秒となる。これは一つの集団による最適化の約1/9であり、約89%の時間を短縮できることになる。より高速な最適化を行うためには、もっとも簡潔で高効率な方法であると言える。さらに複数の最適化により得られた複数の最適解は、一般に異なる構造を持つ数理モデルである。ここでの最適化問題は逆問題であるため、もとより唯一の解を特定するには情報が欠如していることから、最終的に解を一つだけ報告することにすると、他のネットワーク構造を無視することになり、場合によっては有力な情報が失われることになる。ここでの方法における最終的な最適化を行う前に、複数得られている最適解候補をユーザーに提示することで、有益な情報をユーザーが得られやすくなることが考えられる。

最後に遺伝子ネットワークに対して行った最適化では、複数のデータセットを用いることで、ネットワーク構造を特定することができた。最適化対象としたS-システムパラメータは、モデルの一部であるが、この結果は現実の系への適用の可能性を示すものである。最適化の目的がネットワーク構造の探索であることを考えると、表6.17に示す第78世代

で構造の探索はすでに終了した言える。第78世代から最適化が終了する第266世代までは局所探索によるパラメータ値の決定が行われていると考えられる。つまり、終了条件を系の特徴などに応じて適切に設定することにより、この場合の第266世代まで最適化を行わなくても、早期に探索を終了し、最適化時間を短縮することができると思われる。

現実の実験によるデータに、ここで開発した最適化アルゴリズムを適用して未知のネットワーク構造の推定を行おうとする際には、データに大きな測定誤差が含まれていることが多いため、ここで行っているような精度での最適化はあまり意味がないことも考えられる。

第8章　総合考察

開発したアルゴリズムが、逆問題の解法として有効であることをこれまで示してきたが、現実の生体内反応系に適用するには、まだ明らかでない点が多い。その一つが大規模な系に対する最適化である。ここでは最適化パラメータ数が12の最適化しか行っていないが、実在の系では数万の桁になる。そこでまず、2値論理ネットワークと、ここで開発した手法を組み合わせて最適化を行った例を示す[9]。

この例では、要素数30の、遺伝子ネットワークを想定したネットワークに対して構造最適化を行った。おおまかな手順は以下のようになる。(0) 遺伝子ネットワークの、発現量測定結果 (時系列データ) を用意する。(1)2値論理ネットワークでモデリングし、測定結果からは論理的には遺伝子間の制御の順序関係が特定できない遺伝子 (ループ構造などになっている場合、これらの遺伝子を同値類と呼ぶ) をグルーピングする。(2) 同値類に対して、サンプリングポイントが複数ある時系列データを用意し、開発した最適化法を用いてS-システムでモデリングする。

最適化するための情報は、各遺伝子の時系列データ、およびその総数が30個であるということである。まず時系列データから2値論理ネットワークにより同値類となる遺伝子群をグルーピングする (図8.2(B))。各遺伝子群間の相互作用の有無は、2値論理ネットワークにより確定される。しかし一つの遺伝子群内に含まれる遺伝子間の相互作用は分からないため、これらに対してここで開発したアルゴリズムを用いて、最適化を行う。こういった方法を用いると、ここで開発した手法に対して、要素数を少なく抑えることができ、またそのため最適化に要する時間も短くて済むため、すでに実用的なアルゴリズムとして適用することができる。適用例[9][38]では、まず要素数30の系をS-システムで記述し、これから、α_i を $i=1$ から $i=30$ まで順次0に設定して時系列データを計算することで、遺伝子を破壊して発現させる実験 (遺伝子欠損株による発現の有無の観察) を想定した30セットの時系列データを用意した (図8.2左部)。そして、要素数が30であることと、この30セットの時系列データのみを用いて、まず2値論理ネットワークで最適化を行った (図8.2(A)、(B))。ここで得られた同値類に対してここで開発したアルゴリズムを適用し、図8.2(C) に示すネットワーク構造を得た。これはもとの時系列データを生成するのに用いたS-システムモデルと同じものである。つまり、2値論理ネットワークと組み

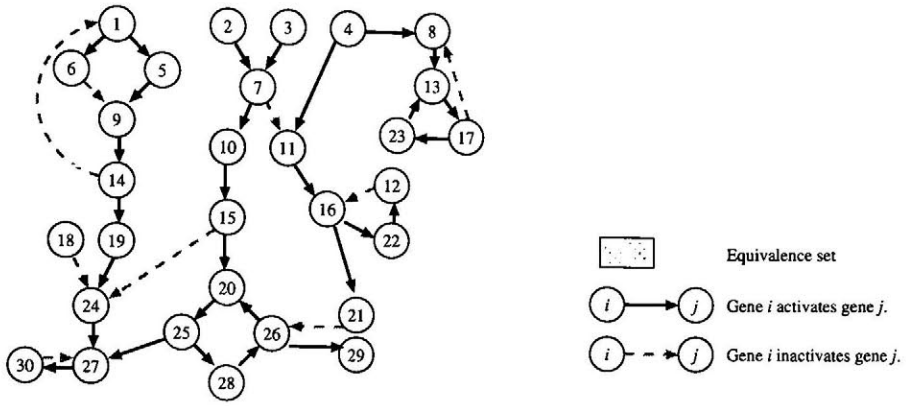

図 8.1: 要素数 30 の遺伝子ネットワークを想定したモデル。

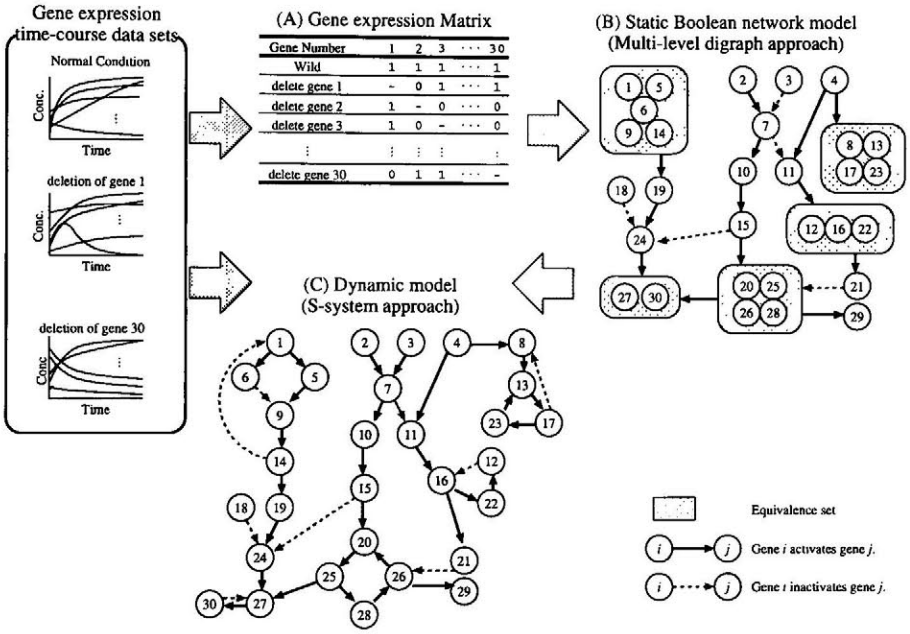

図 8.2: 2 値論理ネットワークと組み合わせた手法で、遺伝子ネットワーク構造を特定するための手順。まず遺伝子ネットワークに含まれる各遺伝子に対して強制発現、または発現抑制をした実験を行い、その発現量を調べる。そして (1)2 値論理ネットワークでモデリングを行い、遺伝子を同値類としてグルーピングする。(2) 同値類に属する遺伝子の時系列データを用いて、ここで開発したアルゴリズムを用いてネットワーク構造を最適化する。(3) さらに不明な構造に対して、少数の実験を行うことでネットワーク構造を特定することができる。

合わせた手法を用いることで、現段階で既に、構造解明を行うことができるアルゴリズムであることが示されている。

しかし現実のネットワークに適用するには、問題点がまだ残されている。一つは実験技術の問題で、現在は遺伝子の発現量の時系列データを得るのは非常に難しいということである。つまり個々で開発したアルゴリズムを適用するのに必要な情報は、実在の遺伝子ネットワークから得られないということである。遺伝子ネットワークの構造解明のための実験で、現在もっとも注目され、用いられているのはDNAマイクロアレイ(ジーンチップ)である。これは、調べたい遺伝子(cDNA)の塩基配列が既に特定されているときに、ある一時点での遺伝子の発現の有無を知る方法である。発現量を知ることはできない。2値論理ネットワークによるモデリングは、この実験手法を想定して開発されたものである。現在は発現の有無だけしか調べることはできないが、発現量を知ることができるようなDNAマイクロアレイも開発されつつある。これが実用化されれば、発現量の時系列データを得ることができるものと期待されている。

実験技術による問題はもう一つある。測定誤差とサンプリングポイントである。生体内反応系を対象とした定量では、通常数十%、場合によっては100%を上回る測定誤差が、測定値には含まれている。ここで開発した最適化手法はおおよそ数%の誤差で最適化しており、高精度であるとも言えるが逆に言うと、そういった細かい差を手がかりに最適化を行っているとも言える。また開発したアルゴリズムの検証に用いた時系列データは、各要素に対してサンプリングポイントが50点あったが、現実の測定では10点以下であることも少なくない。サンプリングポイントが少ない場合は、相対誤差を用いた評価関数が、どの程度モデルの時系列データの再現性を反映しているか、信頼性が低くなることになる。しかし、サンプリングポイントが少なく、しかも大きな誤差を含んでいるデータに対して、データを補間して最適化アルゴリズムに与える時系列データの点数を増やすことで、最適化できることを確かめている。その例では、要素数5の遺伝子ネットワークを想定した、オリジナルとなるS-システムモデルから時系列データを計算し、データを間引きしてサンプリングポイントを各要素につき10点に減らし、一様乱数で20%以下のノイズをデータに加えたものを、実験により得られたものと想定して用意した。これに対して3次のスプライン補間で、データ点数を51点に増やして最適化したところ、符号なし整数と正規分布乱数を用いたアルゴリズムで、おおよそオリジナルに近い構造のモデルを得ることができている。

そして最大の問題は、遺伝子ネットワークは巨大な系であることである。ここで解析した例は、開発したアルゴリズム単独では要素数2、2値論理ネットワークと組み合わせた方法では要素数30が最大である。これに対して生物での遺伝子の総数は数千から数万個にのぼる。またDNAマイクロアレイによる解析は、装置のしくみが単純であり集積化し

やすいため、遺伝子数が非常に多い場合でも、一度に発現の有無を調べることができる。実験技術の開発が進めば、近い将来に同様の規模で、時系列データが得られるようになると考えられる。そういったデータに対して、ここで開発したアルゴリズムを何の改良もなくそのまま適用しても、最適化、さらには逆問題の解決は困難であると考えられる。

もっとも簡潔で効果のある改良は、並列化である。ここでも様々な最適化条件に対して複数回の最適化を行ったが、これをまとめて1回の最適化とすることである。開発したアルゴリズムによる1回の最適化は、局所探索に外ならない。これを最適化の速度を保ったまま大域探索とするには、同時に複数の個体集団を生成し、それぞれ独立で最適化を行うことが考えられる。これにより得られた複数の最適解は、ぞれぞれ真のネットワーク構造の候補であり、真のネットワーク構造を実験により確かめるための助言となりうる。さらに、得られたネットワーク構造の候補をもとに個体集団を生成して最適化を進める、最適化の進行中に、個体集団間でエリート個体を交換するなどの方法も考えられる。これはGAにおいて島モデルと呼ばれるアルゴリズムであり、現在その有効性を検証中である。

S-システムパラメータの総数は、要素数をnとするとき$2n(n+1)$個となり、要素数(系の規模)の増大にともなって、$O(n^2)$で増えていく。これは探索空間の次元が$O(n^2)$で増えていくということでもあり、探索空間を格子状に区切ったとしても、探索空間の大きさ(格子の交点の個数)は$O(n^2)$乗で増えていくということである。これをそのまま受け入れると、現実の遺伝子ネットワークに対する最適化は不可能である。しかし、実際には系の大きさはおおよそ$O(n)$で増えていくと考えられる。これは、系に含まれる各要素は、ごく少数の他の要素とだけ、直接に相互作用を行うと考えられるからである。つまり遺伝子ネットワークでは、ある遺伝子は他の数個の遺伝子の発現にのみ関わっている、ということである。多数の遺伝子の発現を直接制御する遺伝子やタンパク質も、もちろん存在するが、その個数はやはり、ごく少数であると考えられる。この直接に相互作用する要素の個数をある一定数A以下であるとすると、要素iの変化速度を記述するS-システムの指数係数g_{ij}およびh_{ij}の個数は、A以下となる。そうすると系を構成する要素が1個増えるごとに、系全体を記述するS-システムの指数係数のうち、非ゼロのものがAずつ増えていくことになる。これにより骨格構造化を改良することができる。つまりある閾値以下の指数係数をゼロにすること、および指数係数のうち絶対値の大きなものからA以外はゼロにすること、である。しかしこの方法は、構造が未知である系に対して適用するには不用心であり、真の構造では小さな指数係数で表現される相互作用が、骨格構造化の閾値を大きく設定してしまったために見つからなくなってしまうのと同様の問題を引き起こすため、適用するには、ある程度ネットワーク構造について予測がついている場合などに限られる。

現在は数理モデルはS-システムで表現しているが、これはあらゆる相互作用ネットワー

クモデルを表現できる反面、パラメータ数が非常に多いという欠点を持つ。そこで、巨大な系に対しては異なった形式の数理モデルを導入することも考えられる。しかし構造が未知である系を表現するためには、全結線モデルであることが必要であり、S-システムよりも簡潔な表現はむずかしい。簡潔なモデルはすなわち表現の自由度が低いためである。したがってこういったアプローチは、ネットワーク構造が予測されている場合などに有効である。モデルを記述するパラメータ数が少なくなれば、開発したアルゴリズムによる最適化はS-システムの場合よりも容易であると考えられる。また勾配法などの解析的手法が有効であることもあり得る。ここには示していないが、解析的手法とGAを組み合わせたアルゴリズムでは、最適化するS-システムパラメータの個数が10個を越えるような場合には、最適化が困難であった。しかし解析的手法は局所探索としては、GAに比べてはるかに高速、高精度な最適化法であり、これを適用することができれば、非常に有効なアルゴリズムとなり得る。解析的手法は目的関数の形状により、速度や精度が大きく影響されるので、GAで発生した探索点から目的関数の形状を推測し、適用する解析的手法の種類や適用の可否を判断する手法を開発し、ここで開発した手法、これを並列化した手法と組み合わせることで、大規模な系に対して高速な最適化手法を構築できるものと思われる。

また前章で触れたように、逆問題に対して最終的に得られる解を、計算機アルゴリズムで唯一に特定してしまうことは、ユーザーの判断を誤らせる原因となることがあり得るため、複数の解をユーザーに提示することが有用である。つまり、実験における測定誤差が数十%、あるいは100%を越えることがしばしばあることを考えると、誤差数%になるまで最適化を行っても、意味がないことがあり得る。測定誤差が全くランダムに含まれている場合には、補間を用いることで最適化精度を上げることができることは確かめているが、実験環境によっては、様々な傾向を持った、全くのランダムではない誤差が観測値に含まれている場合がある。そう言った場合には、系の特徴から適切な骨格構造化の閾値を設定し、データセット数を複数与え、最適化の終了条件を、相対誤差がある設定値に到達するまでなどと設定することで、最適化を早期に終了し、多数の複数回の最適化を行うことが考えられる。この場合に、得られた多数の最適解の中に、同じ構造が複数個現れていれば、その構造は真の構造である可能性が高いと考えられる。そう言った構造が複数あれば、それらは真のネットワーク構造への候補であるとすることができる。そして、それらの間の構造を違いから、どれが真の構造であるかを特定するための最小限の実験で、真の構造を得ることができる。つまりこのアルゴリズムによる1回の最適化だけで完全に最適化、構造推定を終えてしまうのではなく、ユーザーによる時系列データの観測、それに基づくネットワークの真の構造への複数の候補の提示、候補から真の構造を選び出すための実験の計画と実行といった、ユーザーと最適化プログラムのやりとりによる構造の解明が、現在のアルゴリズムが、もっともよく貢献できる使い方であると思われる。また現

在、すでに解明されているネットワーク構造をデータベース化する研究、作業も行われている。こういったデータベースを利用することができれば、最適化対象となる系の部分的な局所構造に対して、あらかじめ構造を仮定したり、ありえない構造を探索範囲から外すことなどで、解の候補を絞り込むことができる。これは人工知能の分野で研究されているエキスパートシステムと呼ばれるものと似たしくみである。エキスパートシステムは医療診断システムなどへの応用が研究されているが、本研究で開発したアルゴリズムを推測エンジンとして、ネットワーク構造のデータベースを知識データベースとして用いることで、大規模ネットワーク構造の診断システムといったものを構築できると考えられる。

謝辞

本研究を行うにあたって、長期にわたり御指導、御助言をいただいた九州大学大学院農学研究院生物機能科学部門の岡本正宏教授に、心から深く感謝いたします。また広い視点から様々な御助言、御意見を頂いた九州工業大学情報工学部生物化学システム工学科の柏木浩教授に、深く感謝いたします。セミナー等で有益な御助言をいただいてきた、情報工学部生物化学システム工学科の佐藤文俊助手に、深く感謝いたします。

第8章の要素数30のネットワークに対する最適化、サンプリング・ポイント数が少ない場合や測定誤差が大きなデータに対する最適化の検証は、三井情報開発株式会社総合研究所情報環境研究センターの牧幸浩氏によるものです。深く感謝します。共同で研究を行ってきた九州工業大学大学院情報工学研究科情報科学専攻博士前期課程の古賀信人氏、研究環境としての計算機管理を担っていた九州工業大学大学院情報工学研究科情報科学専攻博士後期課程の田中康司氏に深く感謝いたします。また研究にあたり、公私にわたって支えであった九州工業大学情報工学部生物化学システム工学科の柏木、岡本研究室のメンバーに、深く感謝いたします。

参考文献

[1] Alberts, B., Bray, D., Lewis, J., Raff, M., Roberts, K., Watson, J.D., Molecular Biology of the Cell 3rd ed. Garlnd Publishing, 1994.

[2] Baker, J.E., Adaptive seletion methods for genetic algorightms, Proceedings of the International Conference on Genetic Algorithms Lawrene Erlbaum Associates. Hillsdale, 101-111, 1985.

[3] Davis, L., Handbook of genetic algorithms. Van Nostrand Reinhold, 1991.

[4] De Jong, K.A., An Analysis of the Behavior of a Class of Genetic Adaptive Systems. Doctoral sidertation, University of Michigan, 1995.

[5] Goldberg, D.E., Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, 1989

[6] Holland, J.H., Reitman, J.S., Cognitive systems based on adaptive algorithms, in Waterman, D.A., Hayes-Roth, F.(eds), Patter-Directed Interence Systems. Academic Press, 1978.

[7] Irvine, D.H, Savageau, M.A, Efficient solution of nonlinear ordinary differential equations expressed in S-system canonical form. SIAM J. of Numerical Analysis, **27**, 704-735(1990).

[8] Maki, Y., Watanabe, S., Eguchi, Y., Tominaga, D., Okamoto, M., AIGNET: A System That Infers Large Scale Genetic Networks, Genome Informatics 1999, 253-254, Universal Academy Press, 1999.

[9] Maki, Y., Tominaga, D., Okamoto, M., Watanabe, S., Eguchi, Y., Development of a System for the Inference of Large Scale Genetic Networks. it Proceedings of Pacific Symposium of Biocomputing 2001, in press.

[10] Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller A., Teller, E., J. of Chemical Physics, **12**, 1087-1092, 1953.

[11] Michalewicz, Z., Genetic Algorithms + Data Structures = Evolution Programs, Second, extened edition. Springer-Verlag, 1994.

[12] Powell, M.J.D., An effective method for finding the minimum of a function of several variables without calculating derivatives. Conputer J., **7**, 155-162, 1964.

[13] Okamoto, M., Morita, Y., Tominaga, D., Tanaka, K., Kinoshita, N., Ueno, J-I., Miura, Y., Toword a Virtual-Labo-System for Metabolic Engineerig: Development of Biochemical Engineering System Analyzing Tool-Kit(BEST-KIT). Pacific Symposium on Biocomputing '97 World Scientific, 304-315, 1997.

[14] Okamoto, M., Nonata, T., Ochiai, S., Tominaga, D., Nonlinear numerical optimization with use of a hybrid Genetic Algorithm incorporating the Modified Powell method. Applied Mathematic and Computation, **91**, 63-72, 1998.

[15] Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P., Numerical Recepies in C. Cambidge University Press, 1987.

[16] Rosenbrock, H.H., An automatic method for finding the greatest or least value of a function. Computer J., 175-184, 1960.

[17] Savageau, M.A., Biochemical system analysis: a study of function and design in molecular biology. Addison-Wesley, 1976.

[18] Savageau, M.A., Rules for the evolution of gene circuitry. Proceeding of Pacific Symposium on Biocomputing '98, 54-65, World Scientific, 1998.

[19] Tanase, R., Distributed Genetic Algoriths Function Optimization. Ph. D. Thesis, Department of Electrical Engineering and Computer Science, University of Michigan, 1989.

[20] Tominaga, D., Okamoto, M., Discovery of Reaction Network Describing complex Nonlinear Dynamics: Essential Model for Temporal Input-Output Matching. Proceeding of The Ninth Symposium on Chemical Engineering —Kyushu-Taejon/Chungnam, 105-106, 1995.

[21] Tominaga, D., Ueno, J., Miura, Y., Okamoto, M., Discovery of a Skeletal Network Describing Complex Nonlinear Dynamics: Optimized Essential Model for Temporal Input-Output Matching. Tutorials of the 4th International Conference on Soft Computing, Fuzzy Logic Systems Institute, Japan, 65-79, 1996.

[22] Tominaga, D., Okamoto, M., Design of Canonical Model Describing Complex Nonlinear Dynamics, 7th International Conference on Computer Applications in Biotechnology -CAB7-, 85-90, Elsevier Science, 1998.

[23] Tominaga, D., Okamoto, M., Maki, Y., Watanabe, S., Eguchi, Y., Nonlinear Numerical Optimization Technique Based on Genetic Algorithm for Inverse Problem: Towards the Inference of Genetic Networks. Computer Science and Biology (Proceedings of the German Conference on Bioinformatics GCB '99), 127-140, GBF-Braunschweig, German, 1999.

[24] Tominaga, D., Koga, N., Okamoto, M., Efficient Numerical Optimization Algorithm Based on Genetic Algorithm for Inverse Problem. Proceedings of the Genetic and Evolutionary Computation Conference: GECCO-2000, 251-258, Morgan-Kaufmann, 2000.

[25] Voit, E.O., Canonical nonlinear modeling: S-system approach to understanding complexity. Van Nostrand Reinhold, 1991.

[26] 上滝、長田、白川、長谷川、深尾、自動制御理論. 電気学会, 1975.

[27] 奥村, C 言語による最新アルゴリズム事典. 技術評論社, 1991.

[28] 金久, ゲノム情報への招待, 共立出版, 1996.

[29] 金久 (編), シリーズ・ニューバイオフィジックス 11 ヒューマンゲノム計画. 共立出版, 1997.

[30] 嘉納, コンピュータ制御機械システムシリーズ 3 システムの最適理論と最適化. コロナ社, 1987.

[31] 酒和, 田中, 遺伝的アルゴリズム. 朝倉書店, 1995.

[32] 清水 (編), バイオプロセスシステム工学. アイピーシー, 1994.

[33] 洲之内, サイエンスライブラリ理工系の数学 15 数値計算. サイエンス社, 1978.

[34] 戸川, シリーズ新しい応用の数学17 共役勾配法. 教育出版株式会社, 1977

[35] 森田, 生体内非線形反応系解析のためのシミュレータの設計・開発. 九州工業大学修士論文, 1995.

[36] 松野、遺伝的プログラミングを用いた非線形連立微分方程式の自動推定. 九州工業大学卒業論文, 2000.

[37] ロザノフ、ソボリ (坂本、磯野訳), 確率論・モンデカルロ法. 総合科学出版, 1970.

[38] 岡本, S-system による遺伝子の相互作用推定. ゲノム情報生物学 (高木, 冨田編), 165-188, 中山書房, 2000.

[39] 富永, 岡本, 逆問題 (Inverse Problem) 解決のための遺伝的アルゴリズムを用いた多次元非線形数値最適化手法の開発. 化学工学論文集, **25**, 2, 220-225, 1999.

付 録A　プログラムの概要

A.1　プログラムの使用法

ここでは、プログラムに対する入力の用意の仕方、出力の内容、プログラム実行中の動作について述べる。

プログラムに対する入力は、最適化の情報源となる系の要素数と各要素の時系列データ、および最適化手法を制御するパラメータ群の二つに大別される。前者には、あらかじめ構造が予測されている場合に、それを乱数で作成するGAの初期集団に混ぜるために、初期推定として与えることも含まれる。GAの個体数や突然変異率など、骨格構造化の閾値などは後者である。

プログラムの出力は、必要となるのは最適化されたS-システムパラメータのみである。それに付属して、プログラムの実行時間やGAの世代交代の回数なども出力する。また最適化中には、現在の世代交代の回数、エリート個体の評価値なども出力する。

入力はすべて、UNIX上のファイルを用いて行う。出力はプログラムの実行環境の標準出力を用いる。このためプログラムの実行には、初歩的なUNIXオペレーティングシステムの知識を要する。

A.1.1　入力

入力する事項は全てファイルに記述しておき、プログラムの起動時にファイル名をコマンドライン引数として与えることで入力する。ファイルはテキストファイルであり、1文字で1バイトの数字とアルファベットのみを用いる。これにより、利用者は一般的な計算機操作の知識のみで入力ができることになる。またほかのプログラム(JavaやGUIを用いた別の最適化プログラムなど)との連携や、このプログラムのほかの計算機またはほかのオペレーティングシステムへの移植を容易にする。

入力ファイルはviやemacsなどのエディタを用いるなどして、適宜作成する。ファイル中の各事項は行単位で記述される。行の先頭が#の行は、無視される。そのため備考などを記入するために使うことができる。そこを含め入力ファイル中には、多バイト長の

文字(日本語などのいわゆる2バイト、または全角の文字)を用いることはできない。またアルファベットの大文字と小文字は同一視される。行の先頭および行末には余分な空白文字が入っていてもよい。空白文字とは、スペース文字およびタブ文字である。改行文字はデータの区切りとして扱われる。

基本的な最適条件

ファイル先頭の3行は、系に含まれる要素の個数、時系列データのサンプリングポイントの数データセットの数である。これらは順不同である。それぞれはキーワードと、数字の組で記述されるが、それらの間には一つまたは複数の連続する空白文字を挿入する。要素数、サンプリングポイント数、データセット数それぞれのキーワードは **CHEMICALS**、**POINTS**、**NUMBER_OF_SETS** である。具体例は以下のようになる。

```
CHEMICALS          2
POINTS            51
NUMBER_OF_SETS     3
```

これは要素数が2、サンプリングポイントは51点、データセット数は3の場合である。

最適化法の制御パラメータ

ファイルの先頭から4行目以降には、最適化法を制御するためのパラメータを上述と同様に記述する。記述されるパラメータは(括弧内はキーワード、記述する値または文字列、および既定値)は以下のようになる。

最終サンプリングポイントでの時刻(**TIME_RANGE**、実数、既定値なし)、微分方程式を解く際の独立変数(時間)の刻み幅(**TIME_RESOLUTION**、実数、既定値なし)、GAの個体数(**INDIVIDUALS**、整数、既定値なし)、突然変異率の初期値(**MUTATION_RATIO**、実数、0.01)、突然変異率の最大値(**MAX_MUTATION_RATIO**、実数、0.99)、突然変異率を上げるために要する、エリート個体の評価値が変化しないまま経過した世代交代の回数(**MUTATION_RAISE**、整数、10)、乱数の種類(**RANDOM_TYPE**、NORMALかOTHER、NORMAL)、骨格構造化を行う世代交代の回数(**CUTOFF**、整数、10)、骨格構造化の閾値(**CUTOFF_VALUE**、実数、0.2)、最大世代交代回数(**COUNT_LIMIT**、整数、500)、実数の精度(**SEARCH_RESOLUTION**、実数、0.01)、S-システムパラメータ α_i と β_i の探索範囲(**AB_SEARCH_RANGE**、実数、10.0)、S-システムパラメータ g_{ij} と h_{ij} の探索範囲(**GH_SEARCH_RANGE**、実数、4.0)、一つの要素についての0でない g_{ij} または h_{ij} の個数(**CONNECTIONS**、整数、系に含まれる要素数と同じ)、

最適化終了と見なすための、エリート個体から計算される時系列データと与えられる時系列データの、サンプリングポイント1点あたりの相対二乗誤差(ALLOWED_ERROR、実数、0.1)、GAの選択方式(SELECTION、1か2、1(ランキング戦略))、ランキング戦略を行う場合の淘汰圧勾配(RANKING_DIVERSITY、実数、1.01)。具体的には、以下のように記述する。

```
TIME_RANGE        3.0
TIME_RESOLUTION     0.05
INDIVIDUALS        10
MUTATION_RATIO       0.01
MAX_MUTATION_RATIO    0.50
MUTATION_RAISE        2
RANDOM_TYPE         0
CUTOFF            1
CUTOFF_VALUE        0.5
COUNT_LIMIT       50000
SEARCH_RESOLUTION    0.1
AB_SEARCH_RANGE       5.0
GH_SEARCH_RANGE       2.0
CONNECTIONS         2
LOCAL_OPTIMIZE        0
ALLOWED_ERROR         0
SELECTION           1
RANKING_DIVERSITY    2.0
```

GAでの選択方式では、1を指定するとランキング戦略に、2を指定するとルーレット戦略になる。また探索範囲は正の実数値で指定するが、α_i と β_i は0以上指定値以下、g_{ij} と h_{ij} は指定値を a とすると、$[-a,a]$ が探索範囲となる。

以上はすべて1行につき一組づつ記述していくが、以下のものについては、キーワードのみの行に続いて行列の形で数値を指定する。系の要素数を n とするとき行列は $n\times n+1$ 行列で、行列中の要素の位置は、対応する位置のS-システムのパラメータを表す。つまり第1列と第 $n+2$ 列の第 i 行要素はそれぞれ α_i と β_i、第1行第2列の要素を左上とする $n\times n$ 行列は g_{ij} を、第1行第 $n+3$ 列の要素を左上とする $n\times n$ 行列は h_{ij} を表す。

$$\begin{array}{cccccccc} 10.0 & 1.0 & \cdots & 0.5 & 12.0 & 1.3 & \cdots & 1.1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 10.0 & 1.0 & \cdots & 1.8 & 12.0 & 0.8 & \cdots & 1.2 \end{array} \longrightarrow \begin{pmatrix} \alpha_1 & g_{11} & \cdots & g_{1n} & \beta_1 & h_{11} & \cdots & h_{1n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha_n & g_{n1} & \cdots & g_{nn} & \beta_n & h_{n1} & \cdots & h_{nn} \end{pmatrix}$$

最適化対象のS-システムパラメータ (OPTIMIZED_PARAMETER、要素は1か0、すべて1)、S-システムパラメータの初期推定値 (INITIAL_GUESS、実数、既定値なし)。具体的には、以下のように記述する (要素数5の場合の例)。

```
INITIAL_GUESS
10.0  0.0  0.0  0.0  0.0  0.0   10.0  2.0  0.0  0.0  0.0  0.0
10.0  0.0  0.0  0.0  0.0  0.0   10.0  0.0  2.0  0.0  0.0  0.0
10.0  0.0  0.0  0.0  0.0  0.0   10.0  0.0  0.0  2.0  0.0  0.0
10.0  0.0  0.0  0.0  0.0  0.0   10.0  0.0  0.0  0.0  2.0  0.0
10.0  0.0  0.0  0.0  0.0  0.0   10.0  0.0  0.0  0.0  0.0  2.0
OPTIMIZED_PARAMETER
1 1 1 1 1 1  0 0 0 0 0 0
1 1 1 1 1 1  0 0 0 0 0 0
1 1 1 1 1 1  0 0 0 0 0 0
1 1 1 1 1 1  0 0 0 0 0 0
1 1 1 1 1 1  0 0 0 0 0 0
```

この例では、初期推定として α_i と β_i はすべて10.0、g_{ij} はすべて0.0、h_{ij} は $h_{11} = h_{22} = h_{33} = h_{44} = h_{55} = 2.0$ 以外の h_{ij} はすべて0.0となっている。また最適化されるパラメータは α_i と g_{ij} だけで、β_i と h_{ij} は初期推定の値がそのまますべての個体にコピーされ、その値で固定される。したがってキーワードOPTIMIZED_PARAMETERで最適化すべきパラメータを指定した場合、最適化されないパラメータには、適切な値をINITIAL_GUESSで指定しておかねばならない。

時系列データ

ファイルの末尾に、時系列データを記述する。時系列データの先頭には、キーワードDATAのみの行を挿入する。時系列データは、1行につきサンプリングポイント1点分のデータを記述する。また行頭にはサンプリングポイントの時刻を記述する。したがって要素数 n の場合、1行につき実数値が $n+1$ 個、空白文字で区切られて並ぶ。あるサンプリングポイントにおいてデータがある要素とない要素がある場合があるが、この場合データがない要素は負の実数を記述しておく。時系列データが負のサンプリングポイントでは、相対二乗誤差の計算を省くことにしている。プログラム内ではサンプリングポイントの先頭時刻は0.0に固定しているが、これも記述しなければならない。またデータセットが複数ある場合には、時系列データのみを連続して記述する。この時系列データよりも後に記

述されている内容は、すべて無視される。具体的には以下のようになる (要素数 2、データセット数 3 の場合、サンプリングポイントの最終時刻が 3.0 の場合)。

```
DATA
          0          1.5            1
       0.06    1.587645592    1.038707355
       0.12    1.666077566    1.095278963
...(中略)...
          3    1.259287186     0.9784717043
          0              1          1.5
       0.06     0.9783778662       1.40954008
...(中略)...
          3    1.149116204    1.714155469
          0           1e-14             2
       0.06  9.231021363e-15    1.573255722
...(中略)...
       2.94  2.857285362e-10  1.562164947e-05
          3  3.632312509e-10  1.228842471e-05
```

データセット数が 3 であるので、各時系列データの先頭 (サンプリングポイントの時刻が 0 である行) が 3 行がある。ここでの各要素の値を、個体の評価値を計算するために微分方程式を解く際の、初期値とする。データセットが 3 の場合には、各個体について 3 つの初期値の組を用いて、微分方程式を 3 回解く。その結果と、ここに記述する時系列データとの相対二乗誤差から、個体の評価値を計算する。

ここでの最後の行が、入力ファイルの末尾である。

A.1.2 コマンドラインオプションと出力

基本的な動作

プログラムは GUI(Graphical User Interface) を備えない。プログラムの起動は UNIX 上の端末から、すでに起動されているシェルプログラムがユーザーからの入力を待っている状態で、プログラム名を入力することで起動する。このとき、コマンド名に続けて入力ファイル名を指定する。もっとも簡潔な起動方法を示す。以下、例に示す%は、コマンドがその前までに受け付けた処理を終了、またはバックグラウンドで実行していて、ユー

ザーからの入力を受け付けられる状態であることを示す、コマンドプロンプトである。

```
% ssearch data.bhv
```

ここで**ssearch**がプログラム名、**data.bhv**が入力ファイル名である。これらの名前はまったく任意であり、ユーザーが自由に変更できる。ここでは、この例に示す名前に固定しておく。

プログラムを上記のように起動すると、最適化終了まで端末画面上には何も出力されず、最適化終了と同時に、エリート個体の評価値とそのS-システムパラメータを表示し、プログラムを終了してコマンドプロンプトが表示される。上記の例で要素数が2である場合、以下のようになる。

```
% ssearch data.bhv
0.201818
   3.2*     0*    1.8*    3.3*     1*     1*
   2.4*  -1.8*  -1.7*    0.9*     0*     0*
%
```

最後の%はコマンドプロンプトである。0.201818がエリート個体の評価値、それに続いてエリート個体のS-システムパラメータが表示されている。どの数値がどのパラメータの値であるかは、第A.1.1章で示すとおりである。ここでは系の要素数が2であるためS-システムパラメータは2行で表示され、$g_{11} = h_{21} = h_{22} = 0.0$、$h_{11} = h_{12} = 1.0$である。また$g_{21} = -1.8$と$g_{22} = -1.7$は負であり、$\alpha_1$と$\beta_1$はそれぞれ3.2と3.3で似た値である。$\alpha_2 = 2.4$、$\beta_2 = 0.9$である。この例が示す構造を図A.1に示す。

出力されたS-システムパラメータの数値の末尾の*は、そのパラメータが最適化対象であったことを示す。入力ファイルでキーワード**OPTIMIZED_PARAMETER**で最適化対象にしない、と指定されたパラメータには*はつかず、初期推定の値がそのまま表示される。

プログラムを起動してから、最適化結果が出力されるまでの間に、端末上で**C-c**を入力する(計算機のキーボードのコントロールキー(Control あるいは Ctrl)を押したまま**c**を入力する)と、プログラムは以下の文字列を端末の画面上に出力し、ユーザーからの入力を待つ。

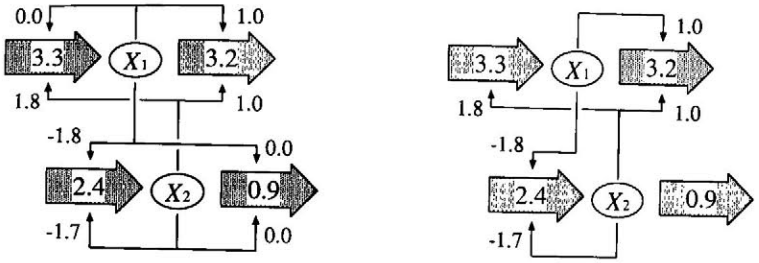

図 A.1: もっとも簡潔な起動方法とその出力の例に示す、最適化されたS-システムパラメータが示す系の構造。パラメータ g_{ij} と h_{ij} が表す直接の相互作用とその大きさを左図に示す。太い矢印は、要素の左側がその要素の生成過程、右側が分解過程を表し、その中に書かれている数値は、系を微分方程式で記述したときの係数であり、S-システムパラメータの α_i と β_i である。パラメータ値が0.0であるところは直接の相互作用がないと考えられるため、それを省くと右のようになる。要素 X_2 は一定の速度で分解され、X_1 と X_2 はそれぞれが X_1 の分解を促進する。X_2 は X_1 の生成を促進し、X_1 と X_2 はそれぞれ X_2 の生成を抑制する。

```
以下の中から、やりたいことを選んでください。
1 : 一番いい個体の遺伝コードを表示する
2 : 任意の個体の遺伝コードを表示する
3 : 現状報告
4 : 一番いい個体の表現型（濃度変化の数値列）をファイルに落す
5 : まちがって押したので、続きをそのまま再開する
6 : 続けてもしょうがないので、終了する
>
```

以下ではこの表示と入力を待っている状態を、選択画面と呼ぶ。下端の行に表示されている>に続けて、1から6までの数字を入力する。6を入力した場合は、エリート個体の評価値とそのS-システムパラメータを表示して終了する。画面上では以下のようになる。

```
得られた最良解 :
0.272883
   0.9*  -1.6*   1.2*   0.8*   0.5*   0.9*
   4.8*     0*  -0.7*   3.9*  -1.8*   1.5*
```

末尾の3行は、上述の出力例と同様である。

1を選んだ場合は、エリート個体の評価値をS-システムパラメータを表示し、また選択

画面を表示してユーザーの入力を待つ。具体的には以下のようになる。

```
得られた最良解 :
0.251949
   1.6*  -0.7*   1.5*   1.2*   1.6*   0.8*
   3.9*     0*  -1.7*   3.4*  -0.6*  -0.6*

  以下の中から、やりたいことを選んでください。
  1 : 一番いい個体の遺伝コードを表示する
  2 : 任意の個体の遺伝コードを表示する
  3 : 現状報告
  4 : 一番いい個体の表現型（濃度変化の数値列）をファイルに落す
  5 : まちがって押したので、続きをそのまま再開する
  6 : 続けてもしょうがないので、終了する
>
```

2を選んだ場合は、評価値順に並べた時の個体の順位を入力する。すると、その個体の評価値とパラメータが表示され、1を選んだときと同様に、選択画面に戻る。

```
表示したい個体の順位は？ > 10
0.000000
   1.6*  -0.7*   1.5*   1.2*   1.6*   0.8*
   1.2*     0*  -1.7*   3.4*  -1.3*    -2*

  以下の中から、やりたいことを選んでください。
  1 : 一番いい個体の遺伝コードを表示する
  2 : 任意の個体の遺伝コードを表示する
  3 : 現状報告
  4 : 一番いい個体の表現型（濃度変化の数値列）をファイルに落す
  5 : まちがって押したので、続きをそのまま再開する
  6 : 続けてもしょうがないので、終了する
>
```

3を選んだ場合は、その時点でのGAの集団の各個体の評価値について、最大、平均、分散、微分方程式が解けない(評価値が0.0の個体)の数を表示し、それに続いて要素数、サンプリングポイント数、GAの個体数、そのうち微分方程式が解ける個体の個数、その

時点での世代交代の回数、その時点での突然変異率、骨格構造化の閾値、骨格構造化の頻度、世代交代の最大回数を表示する。具体的には以下のようになる。

```
1 : 統計的データ
    最優秀の個体の評価値    : 0.251802
    評価値の平均          : 0.045897
    評価値の分散          : 0.005645
    (Except 0 valued indivuduals.
2 : パラメータ
    反応種の数          : 2
    タイムステップ数       : 51
    個体数             : 100
    (そのうち微分方程式が解けるもの : 73)
    現在までに重ねた世代数 : 7
    突然変異率          : 0.010000
    カットオフの値        : 0.500000
    カットオフは何回おきか : 1
    世代交代の限界        : 20

以上で現状報告を終ります。
遺伝コードを見たい個体があったら、その番号を入力して下さい。なければリターンキーだけを押して下さい。>
```

これに続いて、個体の番号を入力すると、その数字の順位の個体の持つS-システムパラメータが表示される。これは、選択画面で2を入力した場合とまったく同じである。

選択画面で4を選んだ場合には、エリート個体の持つS-システムパラメータから微分方程式を解き、それにより得られた時系列データを新しく生成したファイルに保存する。4を入力すると、そのファイル名の入力を促され、ファイル名を入力すると、ふたたび選択画面に戻る。

```
出力するファイル名を入力して下さい > data
カレントディレクトリのファイル data に、微分方程式を解いた結果を書き込みました。

  以下の中から、やりたいことを選んでください。
  1 : 一番いい個体の遺伝コードを表示する
  2 : 任意の個体の遺伝コードを表示する
  3 : 現状報告
  4 : 一番いい個体の表現型（濃度変化の数値列）をファイルに落す
  5 : まちがって押したので、続きをそのまま再開する
  6 : 続けてもしょうがないので、終了する
>
```

この例で指定したファイル **data** には、一行に 1 サンプリングポイントの数値データが書き込まれる。一行には要素数+1 個の数値データがあり、第一カラムはサンプリングポイントの時刻である。このような形式により、様々なグラフ作成ソフトウェアを用いてのプロットが非常にやりやすくなっている。要素数 2 の場合は以下のようになる。

```
0.000000 1.500000 1.000000
0.060000 1.483757 1.065829
0.120000 1.469020 1.115051
0.180000 1.442548 1.180978
0.240000 1.430375 1.203008
0.300000 1.418730 1.220059
0.360000 1.407528 1.233304
(以下略)
```

左端、0.000000、0.060000、0.120000 と並んでいるのが時刻、その右の 1.500000、1.483757、1.469020 と並んでいるのが要素 1、残りが要素 2 となる。

選択画面で 5 を入力すると、以下のように表示し、何もせずにそのまま探索を再開する。

```
探索を再開しました。
```

```
  str = (short *)calloc(ALL_BITS, sizeof(short));
  exp = str;
  man = str + EXP_BITS;
  for (i = 0; i < EXP_BITS; i++)     exp[i] = (int)(rand()%2);
  for (i = 0; i < MANTISSA_BITS; i++) man[i] = (int)(rand()%2);
  }
  bsreal(real xx) {
        *this = xx;
  }
};
#else
// 探索範囲を ULONG_MAX 個の小区間に分割して、表現しようとする実数値がその
// 何番目の小区間に入るかで、その実数を表現する
extern real bs_upper_range;
extern real bs_lower_range;
class bsreal {
    unsigned long int num;
public:
    real bs2real(void);        // 実数型への型変換
    bsreal real2bs(real);      // 実数型からの型変換
    operator real();          // 実数型への型変換
    int operator =(bsreal);     // 代入演算子
    int operator +=(bsreal);    // 代入演算子
    int operator -=(bsreal);    // 代入演算子
    int operator =(real);      //     〃
    int operator +=(real);     //     〃
    int operator -=(real);     //     〃
    // 以下の二つは遺伝的操作で直接 bit string を操作できるようにするため
    friend void makechild(genome *par1, genome *par2, genome *child);
    friend void point_mutation(bsreal *, int);
    void print(FILE *);

  // bsreal型のコンストラクタ
    bsreal() {
```

```
  str = (short *)calloc(ALL_BITS, sizeof(short));
  exp = str;
  man = str + EXP_BITS;
  for (i = 0; i < EXP_BITS; i++)     exp[i] = (int)(rand()%2);
  for (i = 0; i < MANTISSA_BITS; i++) man[i] = (int)(rand()%2);
  }
  bsreal(real xx) {
        *this = xx;
  }
};
#else
// 探索範囲を ULONG_MAX 個の小区間に分割して、表現しようとする実数値がその
// 何番目の小区間に入るかで、その実数を表現する
extern real bs_upper_range;
extern real bs_lower_range;
class bsreal {
    unsigned long int num;
public:
    real bs2real(void);        // 実数型への型変換
    bsreal real2bs(real);      // 実数型からの型変換
    operator real();          // 実数型への型変換
    int operator =(bsreal);     // 代入演算子
    int operator +=(bsreal);    // 代入演算子
    int operator -=(bsreal);    // 代入演算子
    int operator =(real);      //     〃
    int operator +=(real);     //     〃
    int operator -=(real);     //     〃
    // 以下の二つは遺伝的操作で直接 bit string を操作できるようにするため
    friend void makechild(genome *par1, genome *par2, genome *child);
    friend void point_mutation(bsreal *, int);
    void print(FILE *);

  // bsreal型のコンストラクタ
    bsreal() {
```

```
        num = (unsigned long int)0;     // 中身を 0 で初期化
    }
    bsreal (real xx)                    // 引数があるときはそれを代入
    {
        if (xx > bs_upper_range) xx = bs_upper_range;
        if (xx < bs_lower_range) xx = bs_lower_range;
        num = (unsigned long int)ULONG_MAX*
            (xx-bs_lower_range)/(bs_upper_range-bs_lower_range);
    }
};
#endif
```

A.2.2 genomeクラス

GAにおける個体を定義するクラス。S-システムモデルを定義するパラメータと要素数の他、エリート個体として世代交代を生き延びた回数をageというメンバ変数に保持する。

```
class genome {          // 個体を表す染色体の型
    int  age;           // 個体の年齢（増殖回数）
public:
    bsreal **a;           // S-system の係数
    bsreal **b;           //      〃
    bsreal ***g;          //      〃
    bsreal ***h;          //      〃
    real   point;         // 個体の評価値

    void copy(genome *p);           // *p = *this
    void birth(genome *p, genome *c);  // 増殖関数  引数:交叉の相手
    void mutation(bsreal, int);     // 突然変異関数
    void setinitial(void);          // 各遺伝子に値をセットする関数
    void print(FILE *, char *);     // 染色体の内容を表示する関数
    friend void makechild(genome *par1, genome *par2, genome *child);
    friend int gnmdiff(genome *, genome *);

    genome() {
```

```
        int i, j, k;
        real tmp;

        age = 0; point = 0.0;
        a = (bsreal  **)calloc(CHEMICALS, sizeof(bsreal  *));
        b = (bsreal  **)calloc(CHEMICALS, sizeof(bsreal  *));
        g = (bsreal ***)calloc(CHEMICALS, sizeof(bsreal **));
        h = (bsreal ***)calloc(CHEMICALS, sizeof(bsreal **));
        if ((a==NULL)||(b==NULL)||(g==NULL)||(a==NULL))
           puts("Allocation Error.");
        for (i = 0; i < CHEMICALS; i++) {
            a[i] = new bsreal;
            b[i] = new bsreal;
            g[i] = (bsreal **)calloc(CHEMICALS, sizeof(bsreal *));
            h[i] = (bsreal **)calloc(CHEMICALS, sizeof(bsreal *));
            for (j = 0; j < CHEMICALS; j++) {
                g[i][j] = new bsreal;  h[i][j] = new bsreal;
            }
        }
        setinitial();
    }

    ~genome() {
        int i, j;
        for (i = 0; i < CHEMICALS; i++) {
            for (j = 0; j < CHEMICALS; j++) {
                delete g[i][j]; delete h[i][j];
            }
            delete a[i]; delete b[i];
            free(g[i]);  free(h[i]);
        }
        free(a); free(b); free(g); free(h);
    }
};
```