# 遺伝的プログラミングによるシステム関数の近似と経済動学モデル分析への応用

時永, 祥三
九州大学大学院経済学研究院 : 教授 : 経営情報システム

https://doi.org/10.15017/1067

出版情報：經濟學研究. 68 (6), pp.31-63, 2002-10-18. 九州大学経済学会
バージョン：
権利関係：

# 遺伝的プログラミングによるシステ関数の近似と経済動学モデル分析への応用

時　永　祥　三

## 1．まえがき

近年、経済社会における複雑な挙動をモデル化する方法論として、複雑系の理論が応用されている。これまでの経済動学、あるいはこれを背景として展開されているマクロ経済分析では、多くの場合米国流の均衡理論あるいは線形性を基本とした構造方程式による分析、あるいは回帰分析が中心であったと言えよう。これらの理論は、経済は均衡することが通常の姿であり、この中間の過渡的な現象として不均衡現象があるとする考えである。この根拠となっているものは、小さな政府論に裏付けられている社会的なシステムの効率性、あるいは合理性を前提としていることは言うまでもない。

これ対して、社会政策の有効性を支持する傾向の強い欧州の研究者は、継続的に不均衡モデルを分析対象としている。これらのどちらが正しいかを証明することは不可能であり、神学論争の域をでないであろう。

以下では、不均衡モデル分析を含めて複雑系理論の提唱する経済モデル分析について、特に遺伝的プログラミング(GP:Genetic Programming)について、これまでの著者の研究と今後の関連性について述べていく。特に、不均衡モデルによる経済動学モデル分析だけではなく、最近では、エージェント理論に代表されるような、小さな行動主体が集合することにより、複雑な現象を発生させるとする、いわゆるエージェントモデルは、ここで取り上げる遺伝的プログラミングそのものであると言える。

論文の性格上、全部の分野について詳細を述べる余裕はないので、以下では、主として、今後の経済動学モデルの新しい構成手法に関連した話題について述べる。

以下では、2．において、GPの原理について述べ、3．では、GPによるシステム方程式近似と制御について述べる。4．では経済動学モデルとカオスについて示し、5．ではInput Pricingのシステム推定へのGP応用を考察する。更に、6．においてワークフロー管理システムの設計とGPについて述べ、7．、8．では、プロダクションルールの生成とGP、および、CNNによるダイナミックスの表現について考察する。

## 2．GPの原理

### 2．1　GPの実行方法

GPはGA(Genetic Algorithm:遺伝的アルゴリズム)の1つの拡張であり、個体はGAのようにビット列ではなく、数学演算のための演算子、変数からなっている[1]－[7]。システム方程

式を表現する方法にはいくつか存在するが、ここでは、比較的効率のよいprefix表現による方法を用いる[14]。

GPのシステムは基本的に、3つの部分からなりたっており、その第1番目は個体の表現である。方程式は木構造で表現できるが、これをprefix表現により置き換えておく。例えば、次のように方程式を表現する。

$(3\times x_1-x_2)\times(x_3-4)\rightarrow$

$\times-\times 3\,x_1\,x_2-x_3\,4$　(1)

時系列を示すシステムの場合、$x_1$、$x_2$などの変数は、$x(t-1)$、$x(t-2)$などのラグ付き変数に置き換えて考える。

次に必要なのが、GPにより表現された個体の解釈である。これには式(1)に示すような式により表現された方程式の右辺の形をもとにして、関数の値を求めることである。このような演算をGPにおいて実施するには、コンパイラなどで用いられるスタックを使う。

第3番目に必要なのが、GPにより個体を性能の良いもの（今の場合には関数近似の能力のたかいもの）に変換していく方法である。この場合、GPを適用する前に、それぞれの個体の能力（適合度とよぶ）を計測する必要がある。例えば、時系列の予測問題では、個体により表現される関数系により予測される時系列の予測値$\hat{x}(t)$と、観測された時系列$x(t)$の2乗誤差をとり、この逆数がこの個体の適合度となる。

個体の集合（プール）の能力をたかめることは、個体に対して交叉処理、突然変異処理を行うことにより可能である。しかし、GAにおける遺伝的操作と異なり、GPにおいては任意の位置で交叉をすると意味をなさない個体を生成する可能性がある。このため、以下で述べるような*StackCount*というカウンタを用いる。

*StackCount*の値は、prefix表現で表現された個体のストリングを左側からサーチしていき演算記号に出会うとその数値を1つ増やし、被演算子に出会うとその数値を1つ減らす操作を実施した結果である。個体のストリングの全体をサーチ終えたあとに、*StackCount*の数値は必ず1になる。従って、GPにおける初期個体を生成するときに、この条件を満足しないものは個体として採用しない。突然変異の処理においては、個体である表現の一部を別のものに置き換える。これらについては、後で述べる。

なお、初期個体を1000個程度生成するが、それぞれの個体が方程式を表現しているかどうかを検査しながら実施する。一般的には、望ましい初期個体を生成するのに多くの試行が必要になるが、個体の長さの最大（関数の複雑さ）を制限していることなどから、本論文で行ったシミュレーションでは、1つの個体に対して平均して25回程度実施すれば、適切な初期個体を生成できる。従って、例えば、1000個体の初期値の生成は800msec程度の時間で終了する。

また、以下の方法では、方程式表現における定数の決定、変数の選択などのために、通常のGAを用いている。GPを用いて関数の形を決定しても、カオス力学系においては初期値も推定する必要がある。また、関数の構造は同じであっても、用いる変数を変更することにより、更に適合度のたかい個体を求めることができる。このような目的のために、GAにおけるストリング（個体表現）で初期値や定数を表現したものを用いることにより、GPを用いて関数形を改善する方法を更に効率的にできる。

具体的には、パラメータをGAにおける個体ストリングに数値として埋め込んで、通常のGAにおける遺伝的操作（交叉処理および突然変異）

により最適化する。

なお、数値は10進数のままストリングとして表現する。ここに示すGAを、GPによる最適化の操作ステップが一巡し終了した段階で適用する。ただし、GPにおける個体のすべてに、ここに述べるGAを適用するのは適切ではないので、適合度の高い個体にだけ用いる。

GPにおいて、実施する交叉処理の原理を、図１に例を用いて示している。図１では個体Ａの交叉位置を乱数により選択し、この位置における*StackCount*を計算しておく。次に、個体Ｂについて同じ*StackCount*をもつ位置を検出する。この例では場所は２つある。この２つの場所から、図１に示すような場所を選択する（選択の確率は等確率であると仮定する）。これらの位置を境界として、それぞれの個体の前半と後半を、相互に交換した個体が図１に示すように生成される。

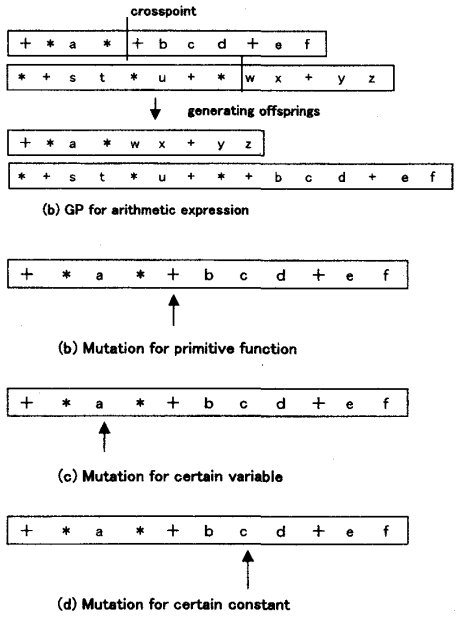



**図１　交叉処理の概要**

## ２．２　アルゴリズム

以上のような方法をアルゴリズムとしてまとめると、次のようになる[１]－[10]。

（ステップ１）

乱数を用いて被演算子、演算記号の並びからなる初期個体のプールを構成する。生成された個体が関数を表現するものであるかどうかの妥当性を、すでに述べた*StackCount*を検査することにより行う。条件を満足しない個体はプールに含めない。必要数の個体が得られるまで繰り返す。

（ステップ２）

個体に表現された関数をもとに、それぞれの個体により得られる予測値を求める。これをもとにして、個体における適合度を求める。個体のプールを適合度の順にソートしておく。

（ステップ３）

遺伝的操作により新しい個体を生成する。適合度に応じて２つの個体を選択する。次に示す適合度から変換された確率に応じて個体$i$が選択される。

$$p_i=(S_i-S_{min})/\sum_{i=1}^{N}(S_i-S_{min}) \qquad (2)$$

ここで、$S_{min}$は適合度の最低値、$N$はプールの大きさである。個体Ａ、Ｂに対する交叉処理は、次のように行う。個体Ａの任意の場所を乱数をもとにして選択する。ストリングの左から数えて、この位置までも*StackCount*の値を$k$としておき、個体Ｂの中で、その位置までの*StackCount*の値が$k$となる場所（一般に複数）を特定しておく。この個体Ｂの複数の場所から、等しい確率で１つを選択する。この位置を基準として、個体Ａ、Ｂの交叉処理を行い、新しい

個体を生成する。生成された新しい個体により、プールに存在する適合度の低い個体を置き換える。

（ステップ4）

初期値と変数の組合せを最適化するために、GAを適用する。GAの個体はすでに述べたように、初期値、定数を数値で表現したものである。GAの操作として、通常よく用いられている交叉処理、突然変異処理を用いる。

（ステップ5）

ステップ2からステップ4までの交叉処理をすべての個体に適用し、新しい個体のプールを作成したあとに、次に示す突然変異を実施確率$p_M$の確率に従って実施する。突然変異にはG－突然変異とL－突然変異の2つの種類があり、適用する確率は同じ（$p_M/2$）とする。

G－突然変異：グローバルな突然変異を意味し、基本的には図1に示すような2つの個体に対する交叉処理である。ただし、今の場合には、個体Aは選択され突然変異を適用する個体であるが、個体Bは、初期個体の発生と同じ手順を用いて一時的に発生させた作業用の個体である。これを個体Sとしておく。最初に、個体Aの適当な位置を乱数で選択しておき、ここまでの*StackCount*を計算する。個体Sにおいて、同じ*StackCount*をもつ場所（一般に複数個）を検出し、その場所から同じ確率で、1つの場所を決定する。個体Aの後半を個体Sの後半と交換する（個体Sに対しては、特に何も操作や保存はしない）。

L－突然変異：任意に個体を選択して、この個体の被演算子、演算記号の部分を、任意に選択した被演算子、演算記号により置き換える。

（ステップ6）

ステップ2からステップ5までの操作を規定回数繰り返す。途中で、関数による近似誤差があらかじめ決めた値より小さくなったらアルゴリズムを終了する。そうでないときには、規定回数まで繰り返す。

以下では、既知のカオス力学系で生成された時系列データに対して、GPによりカオス力学系を推定するため、次に示すHenon mapをも用いる[8]。

$$x(t)=1+y(t-1)-ax(t-1)^2 \quad (3)$$

$$y(t)=bx(t-1) \quad (4)$$

また、やや複雑な時系列として、次に示すUshiki mapにより生成された時系列を用いる。

$$x(t)=[3.7-x(t-1)-0.1]x(t-1) \quad (5)$$

$$y(t)=[3.7-0.15x(t-1)-y(t-1)]y(t-1) \quad (6)$$

表1にはGPにより求めた方程式による予測誤差（次式で定義）を示している。

$$n-rmse=[\sum(x(t)-\hat{x}(t))^2/Ns]^{1/2}\sigma \quad (7)$$

**表1　GPにより求めた方程式による予測誤差**

| name | 2乗平方誤差 | 求解までのGPステップ数 |
|---|---|---|
| Logistic | 1.7e-7 | 9 |
| Henon | 7.2e-8 | 15 |
| Ushiki | 2.6e-7 | 10 |

## 3　GPによるシステム方程式近似と制御

## 4　カオス制御を例として

カオス制御とはカオスを生成するシステムのパラメータを変化させるのではなく、カオス的な挙動をするシステムを、外部から入力を加え制御することにより、カオス的な挙動をするシステムをカオス的ではない状態に移行させる方

法である[12]－[15]。

カオス制御が必要とされる分野やその意義については次のように整理される。不均衡経済モデルに見られるように、カオス的な状況のもとでは経済モデルに変動要因が含まれ、システムはいつまでたっても安定しない。しかし、課題として、ある経済政策をとることにより、不規則に変動する経済モデルから、安定した挙動へと移行させる方法を見いだす必要があったとする。例えば、混乱する市場や経済環境を、政府の政策で鎮静化するなどの例が考えられる[10]。

カオス制御により不動点に移行させるほかに、周期的な振動に対応するアトラクタを目標とする場合もある。もちろん、アトラクタにこのような周期振動が含まれている必要がある。周期振動へのシステムの移行は、経済動学では季節的な変動を許した上での経済の安定化などとして解釈できるであろう。

### 4．1　OGY法によるカオス制御

カオス制御の方法に関しては制御工学を中心として研究がすすめられ、現在ではその方法は次の２つの方法に集約されている。

第１番目はフィードバック制御による方法であり、古くから用いられているシステム安定化手法である。しかし、この方法は工学的には振動の発生を抑制したりシステムが発散するのを防止する意義があるが、システム構成が変化するため、経済モデルなどの挙動を見ながら制御する場合には適していない。

第２番目はOGY法とよばれる方法であり、OGYは手法の開発者の頭文字(Ott, Grebogi, Yorke)に由来している[12]。OGY法は、カオスアトラクタに埋め込まれた多くの不安定軌道が存在する場合、局所安定化が達成できる近傍に近づいたときに、適切な制御入力を加えることにより安定化された周期軌道へと導く方法である。この方法では、システムの構造を変形させることなく、必要とされる短時間にシステムへの入力を加えるだけで安定化を達成できるメリットがある。ただし、一方では、システムに対する即応性がないことや、制御入力をすべき時間が初期値に依存するなどの問題ももっている。

カオスは周期的な変動と不規則な変動の両方を含んでいるので、アトラクタは１つの曲線ではなく、複数の曲線の集合として記述される。点$x_f$に収束する方向に進む線分がある一方で、これから離れていく線分、あるいは$x_f$に近づきながら、結局は$x_f$には至らない線分から成り立っている。このような、$x_f$としては不動点がある。不動点に収束するアトラクタを安定多様体とよび、不動点には至らない、あるいは離れていく線分を不安定多様体とよんでいる[12]。

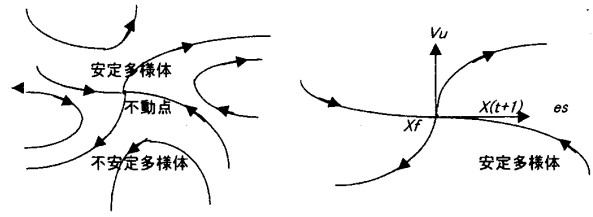



**図２　不安定多様体、安定多様体の接ベクトル**

カオス制御とは、$x_f$の近傍に近づいたときに、外力を加えることにより不動点$x_f$へと収束させる方法であると言える。いま、入力を含む$n$次元非線形離散時間システムを考える。

$$x(t+1)=f(x(t),u(t)) \qquad (8)$$

ただし、$x(t)$は時刻$t$でのシステムの状態、$u(t)$は制御入力であるとする。$x_f$を$u(t)=0$であるときのシステムの不動点であるとする。すなわち、$x_f=f(x_f, 0)$。不動点での線形化システムは、次のようになる。

$$x(t+1)-x_f=A[x(t)-x_f]+bu(t) \quad (9)$$

$$A=D_x f(x_f, 0),\ b=D_u f(x_f, 0) \quad (10)$$

いま、2次元のシステムを考え、不動点$x_f$がこのカオスアトラクタに埋め込まれており、双曲形不安定不動点であるとする。すなわち、$A$の固有値$\lambda_1$、$\lambda_2$はともに実数であり

$$|\lambda_u|>1>|\lambda_s| \quad (11)$$

これらの固有値に対する長さ1の固有ベクトルを$e_u$、$e_s$とする。これらのベクトルは不動点$x_f$の局所不安定および安定の方向を表わしている。すなわち、図2に示すように、安定多様体、不安定多様体のそれぞれに接する接ベクトルとなる。

これらのベクトルを1列目、2列目とする行列$P$を考え、この逆行列$P^{-1}$の行ベクトルを$v_u$、$v_s$とする。これらを反変ベクトル(contravariant vector)とよぶ。

$$P=[e_u, e_s],\ P^{-1}=[v_u, v_s]^T \quad (12)$$

ベクトル$x(k+1)-x_f$はベクトル$e_s$と並行になるので、直交の条件に代入すると

$$v_u[x(t+1)-x_f]=0 \quad (13)$$

となる。従って、$x(t)$が$x_f$の近傍にあり、式(13)が成り立つならば、$x(t+1)$は$x_f$の局所安定多様体に収束する。この条件を書き直すと、次のような入力を印加することにより安定化できることがわかる。

$$u(t)=-\lambda v_u \delta x(t)/(v_u b) \quad (14)$$

$u(t)$はシステムの状態が不動点の近くまで接近するまでゼロにしておき、不動点の近傍にきたときに式(14)の入力をシステムに加える。

文献[12]ではエノン写像 (Henon map) により生成された時系列のカオス制御の例題が示されている。

## 4．2　GPによるシステム同定とカオス制御の結合

GPによりカオス力学系を直接近似する方法を用いれば簡単な制御方法が適用できる[9]。いま、システムの挙動を

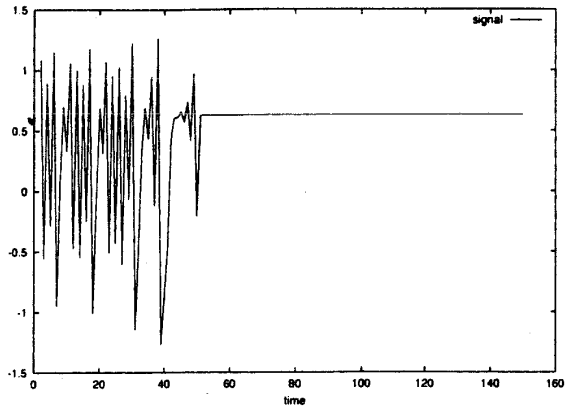
$$x(t+1)=f(x(t))+u(t) \quad (15)$$

と仮定しておく。システムの入力$u(t)$がゼロのときの$f(x(t))$をGPにより近似し、これを$\hat{f}(x(t))$とする。方程式が推定されているので数値計算によりシステムの不動点が求まる。このとき、現在の状態$x(t)$から制御を始めて、次の状態が不動点$x_f$に移行すればよいので$x_f=\hat{x}(t+1)=\hat{f}(x(t))+u(t)$となるように制御$u(t)$を加える。

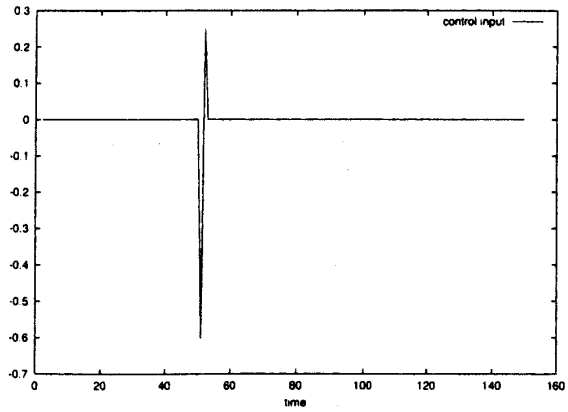
このようにして求められる次の時刻$t+1$における状態について、元のシステムでは$x(t+1)=f(x(t))$となるが、これを近似されたシステムにより求められる望ましい状態、$\hat{x}(t+1)=\hat{f}(x(t))+u(t)$により置き換える。

図3にはUshiki mapにより生成された時系列に対して、ここで述べている制御方法を適用し

た場合の収束状況を示している[9]。




(a) Controled signal




(b) Control input

**図3　カオス力学系推定と制御（上:安定化された時系列、下:入力）**

## 5　経済動学モデルとカオス

### 5．1　経済動学の定式化

カオス理論では、一見不規則と思われる現象が方程式により記述されることを基本としており、予測や推定に好都合となっている。経済モデル分析においても、不均衡の動学モデルの定式化に利用されている。

カオスはもともと予期しない変動に属するものであり、工学分野においても除去すべき対象として見なされ、経済動学においても、不安定な周期変動から安定的な挙動へと移行するための入力やシステムの変更について議論がなされている。

経済時系列に代表される経済データは、これまで確率過程のモデルにより解析が行なわれているが、カオス理論は決定論的な方程式により発生されると仮定することが異なる。この確認の方法としてはいくつか存在するが、基本的にはアトラクタの形状をもとにした方法であると言える。アトラクタは時刻$t$と$t+1$における変数（関数）値を2次元の横軸と縦軸にとって描いたもの（軌跡）であり、このアトラクタの形状がはっきりと見えるのがカオスの特徴である。

例えば、良く知られているカオス発生のモデルとして、次を考える。

$$y(t+1)=a\,y(t)[1-y(t)] \qquad (16)$$

ここで、$a$はカオスになる領域を制御するパラメータであり、約4.0の値とする。このモデルの経済学的な解釈としては、変数$y(t)$を広告宣伝費用とした場合、今期$t$における売上$R(t)$は宣伝費用に比例するが、反面、これを増やしすぎると減少に転じるので、$R(t)=a\,y(t)[1-y(t)]$となる。この利益に比例して来期の宣伝をするので、式(1)で来期の宣伝費用を決めている。これをロジスティック・マップ(Logistic map)とよんでいる。

カオスのもう1つの大きな特徴は、初期値への依存性である。式(1)に示した方程式において、変数$y(t)$の最初の値（初期値）がわずかに異なる2つのケース（例えば、1.0000と1.0005）を考えた場合に、時間の経過とともに、これら2つのケースでの変数$y(t)$の値（挙動）は大きくずれてしまうことが理論的に証明

できる。カオスの発生は決定論的な方程式によりなされるが、その微妙な変化が大きな差異をもたらすという意味で、従来の確率過程に基づく不規則変動とは異なる側面を与えている。

1990年ごろから経済学においても積極的にカオス理論を適用する試みがなされており、従来とは異なる新しい結果や解釈を生んでいる。

(1)経済時系列のモデル化

時系列をカオスによりモデル化することが可能なら、確率過程の理論ではなく確定的な式で時系列の予測が可能となることが期待される。経済分野に限定すれば、その範囲は為替レート変動、株価、ＴＢなど債券価格などから各国のGDPの成長率まで及んでいる。これらの研究の多くが、アトラクタの描画、相関次元の計算、リアプノフ指数の計算により、時系列がカオスであるかどうかを判断するものである[３][13]。

(2)不均衡経済モデル

現在まで、マクロ経済学を中心として経済の諸要素のバランスを前提とした、いわゆる均衡モデルが展開されてきている。これは、例えば、予算制約のもとで便益を最大化するような条件と、その場合の解をもとめる方法であり、経済の均衡を解明する問題である。しかし、現実には証券市場や資本市場における激しい変動、あるいは、そのような変動を積極的に利用する投資など、不均衡経済モデルを前提とした政策や行動が見られる。従って、経済が変動することを前提とした不均衡モデルを積極的に活用し、決定論的なモデルの記述手段としてカオス理論が用いられている[46]。

(3)動学モデル

経済学における動学モデルとは、１つの世代だけのモデルだけではなく、次の世代までを考慮したモデルである。一般にこれを拡張すると、複数の世代にまたがった解析が可能となるので、通常は２世代にわたる解析が行われる。老年世代と青年世代の２つの世代を同時にモデルに組み込む方法（人は青年と老年の２期間だけ生存して消費する）により、カオス性が生まれることがGrandmontとGeanakoplosにより独立に理論的に証明され、これを用いた分析方法が提案されている（世代重複モデル）[46]。

(4)フォン・ノイマン・モデル[46]

フォン・ノイマン・モデル(von Neumann model)とは、次の方程式で記述される離散システムを指しており、パラメータ$a$の選択の方法により$u(t)$と$v(t)$とを描いた相図がカオス的となるケースが生じる。$y(t)$、$u(t)$、$w(t)$、$v(t)$は、それぞれ、産出、単位労働費用、賃金、雇用比率である。$N(t)$、$a_L(t)$は人口と労働投入係数。

$$y(t+1)=y(t)/[a+u(t+1)] \tag{17}$$

$$u(t+1)=a_L(t+1)w(t+1) \tag{18}$$

$$v(t)=L(t)/N(t)=a_L y(t)/N(t) \tag{19}$$

(5)その他

工学分野では、最初の実験的に観測できるカオス現象として、電子回路により模擬されるファン・デル・ポールの非線形方程式が良く知られている。経済学者のグッドウイン(Goodwin)は、このファン・デル・ポール方程式にヒントを得て、ビジネスサイクルをモデル化する方法を提案している[46]。

## ５．２　不均衡経済モデルとカオス

世代重複モデルにおけるカオス生成のメカニズムについて述べる。以下では、世代にわたる

需要と供給のダイナミックスを仮定し、これにより、ビジネスサイクルにカオス性の振動が発生することを示す。短期的には、需要と供給のバランスから価格や労賃は一定であると仮定できるが、時間の経過とともに不均衡状態に移行する。資本と労働からなら２セクターモデルにおいて、世代重複のダイナミックスを用いた在庫分析からカオスが発生する。Li－Yorkの定理により、３周期の振動が発生するとカオスが発生することが示されるので、

検証では、この点に注意する。この定理に基づいてBohmは、老年と若年の２世代、企業、政府からなる経済において消費レベルがカオス性を示すことを証明している[47]。このことから、古典的なWarlasian model均衡のもとでも、パラメータの選択方法によっては、容易にカオス発生が見られることが示される。

時刻$t$における実質賃金、政府歳入を仮定すると、政策変数である支出と税率$(g, \tau)$を決めるための関係式が導出される。

$$y^D = m(t) + g + c(1-\tau)\, y^D \qquad (20)$$

ここで$y(t)$、$L(t)$はそれぞれ生産と資本市場における最適な交換を仮定した場合の産出と雇用(production and employment)であり、$y^D$は効率的な消費(income consistent aggregate effective consumption demand)である。

貨幣ストックのダイナミックスを求めるために、商品市場における効率性(rationing scheme on the goods market)を仮定する。最初に若年世代の効率的行動がなされ、ついで老年世代における行動があり、最終的に政府により調整がなされると仮定する。これにより、次の関係式が得られる。

$$M(t+1) = p(t)[min(y(t), g + m(t) - \tau\, y(t)] \qquad (21)$$

価格と労働が需要と供給の関係で変化すると仮定すると、次のようなダイナミックスを得る。

$$p(t+1) = p(t)[1 + P(\alpha(t), m(t); g, \tau)] \qquad (22)$$

$$w(t+1) = w(t)[1 + W(\alpha(t), m(t); g, \tau)] \qquad (23)$$

ここで$P(.)$と$W(.)$は成長率(functional growth rates)であり、微分可能であると仮定しておく。

これらの関係式から、変数$w(t)$、$p(t)$について周期変動などのカオス性（stationarity, cyclicity, coexisting cycle with complicated basins of attractors and chaotic limit behavior）が見い出せる。

### ５．３　非線形マクロモデルの導出への応用

日本経済のマクロ経済に関する時系列データを集約し、これらの間の非線形モデルをGPにより導出することを目的とする。なお、これらのモデルは機械的に導出されており、今後、見通しのよい簡潔な形式を見い出す検討が必要であることに注意しておく。

今回シミュレーションに用いた時系列と、その略称について、表２に整理している。これらの変数についての線形の回帰分析は、既存の方法で実行することが可能であるので、検証のため回帰分析を実施して比較している。その結果、GPによる方法より一般的に予測精度が悪いことが言える。しかし、論文の本来の目的ではないので、ここでは詳細は省略する[48]。

非線形の関係式を求めるためのGPの条件設定を次のように行なう。

**表2　用いたマクロ変数**

| variables | meaning | variables | meaning |
|---|---|---|---|
| C | consumption | IM | import |
| CPI | consumers price index | W | wage |
| EX | export | R | short term interest rate |
| RATE | exchange rate(vs. US dollar) | T | tax |
| GDP | gross domestic product | U | unemployment |
| I | investment | | |

**表3　推定され方程式**

| | |
|---|---|
| C | $\sqrt{\lvert -4.9875GDP(4.9117-9.6383R+\sqrt{W}+\min(T,IM)\rvert}$ |
| CPI | $C+\sqrt{\lvert C^3R/((\lvert C\rvert-43.19136)(\lvert C\rvert+540.91535)(\lvert C\rvert+63.73490))\rvert}$ |
| EXP | $C+0.0087146TR$ |
| RATE | $\min(C+\sqrt{\lvert C+\sqrt{EXP}-9.15759\rvert},\max(\sqrt{W}$ $-\max(\min(\max(\min(3.24049/R,4.01652),R),$ $\min(CPI,IM)),4.98099),C-3.8238))$ |
| GDP | $(2.44370\min(4.68249+RATE,\sqrt{W})+4.11749\min(RATE-1.00320,\sqrt{\sqrt{W}})$ $+R+\sqrt{\log(CPI)}+30.78961)(CPI+2\log(CPI)+11.78701)$ |
| I | $0.77437(\min(EXP,\max(RATE,\max(0.31920,\log(CPI)))))+C+T)$ |
| IM | $8.76744+2.42996(T-EXP-24.74425CPI+C)/R^2+C$ |
| W | $(T+I)\sqrt{2.03929+U+\min(-3.49390/\max(0.17972RATE,-0.05008),1.83681)}$ |
| R | $C+4.33400\max(-4.51599,0.10840/U)/(4.11499+U-TU/(IMU))$ $-\lvert 0.69506\max(\mathrm{RATE},2.50519)/(-0.47290+\mathrm{T})\rvert$ |
| T | $I\min(0.21197,R)+4.27339+0.28290C$ |
| U | $C-4.70870/(CPI-(T/0.63981\mathrm{mac}(CR,T)))$ $-1.02180/(CPI-(T/0.40377\max(CR,T)))-1.20508R/CPI$ |

**表4　マクロ変数系列の予測誤差**

| variables | n-rmse | variables | n-rmse |
|---|---|---|---|
| C | 0.089 | IM | 0.054 |
| CPI | 0.015 | W | 0.108 |
| EXP | 0.057 | R | 0.179 |
| RATE | 0.099 | T | 0.045 |
| GDP | 0.076 | U | 0.090 |
| I | 0.119 | | |

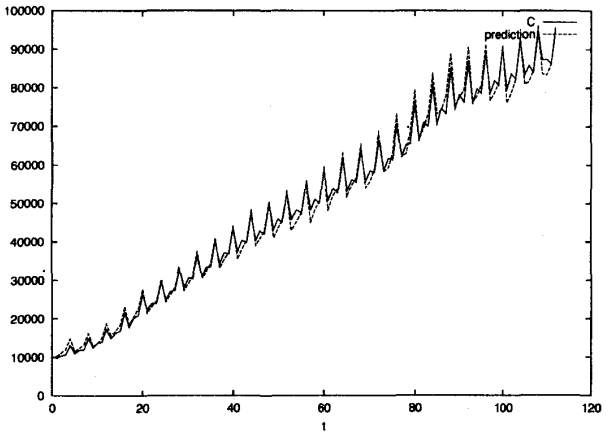



**図4　マクロ変数の予測（C）**

Population size＝500

Maximum size of array＝40

Maximum generation of operation＝500

Data sample＝50

Primitive functions: +, −, ×, /, √, max, min, ||, log

表３には、得られた方程式を示している。これらの予測誤差について、表４に示している。また、図４には予測の例を示している。

## ６　Input Pricingのシステム推定へのGP応用

### ６．１　サービスファシリティにおけるInput Pricing

サービスファシリティにおける顧客の自己目的最適化の原則による参加/退去原則のもとで発生する価格ずけ（Input Pricing）のカオス現象について、観測されたカオス時系列からGPによりダイナミックスを推定、制御する方法を示す。各種のサービス施設利用やネットワーク接続サービスでは、顧客は混雑の状況を見ながらサービスを受けるか拒否するか（参入と退去:join/balk）を決定する。

従来の待ち行列理論は、その前提としてファシリティ側に決定権がある、いわばsupply drivenの環境であるが、顧客がサービスを選択する、いわゆるdemand drivenの仮定のもとでは、更に多くのケースでカオスが観測されることが示されている。

以下では、文献[48]において示されているサービスファシリティにおけるカオス現象の基本モデルについて説明する。客は単位時間当たりΛの到着率でサーバに到着する。しかし、これらのすべてがサービスを受けるものではないので、潜在到着率（potential arrival rate）とよばれる。サーバに到着する客の中で、サービスを受ける者の割合を表すもので、単位時間当たりのサービス人数（arrival rate）をλで表す。

観測期間の間に、客がサービスを受けることによるコスト（費用）を$G_\mu(\lambda)$で表す($\mu$はサーバの容量)。客がサービスを受けることによるサービス価値（service value）の分布関数を$F(x)$、$x \geq 0$とする。

客のモデル化で最も重要なことは、客はサーバにおける待ち（待ち室における客数）を正確には知らないことであり、過去のデータから推定に基づいて行動を決定することである。与えられた到着率のもとで、サーバに加わるためのコストは$p=f+G_\mu(\lambda)$となる（$f$はアドミション料）。客は$p$を正確には知ることができないが、過去のデータから$p$を推定（これを$\pi$とする）し、自己の得ることのできるサービス価値と比較してサーバへの参加/退去を決める。

いま、期間$t$において実際に発生した$p$を$p(t)$としておき、客が次の期間における$p$の値として推定するものを$\pi(t+1)$としておく。指数平滑型の予測（推定）を用いた場合には、次のような関係になる。

$$\pi(t+1)=(1-\omega)\pi(t)+\omega p(t) \qquad (24)$$

ここで、$0 \leq \omega \leq 1$である。変換関数$\overline{F}(x)$の具体的な形としては、例えば、次のようなものが用いられる。

$$\lambda=\Lambda\overline{F}(\pi)=\begin{cases}\Lambda\ (0 \leq \pi \leq d);\\ \Lambda\ (a-\pi)/(a-d)\ (d \leq \pi \leq a);\\ 0\ (\pi \geq a)\end{cases} \qquad (25)$$

サーバの容量の下限$\mu_1$、上限$\mu_2$の範囲にあるとき、変数$\pi$はカオス的な挙動を示す。すなわち、最初の時点でサーバ容量が大きいときには価格$\pi$は1つの安定点に向かう。サーバ容量が次第に小さくなるに従って分岐過程が発生し、価格は2つの値をとるようになる。更に、容量が小さくなると、2周期の振動から4周期振動、更に8周期振動へと移行している。この付近から大きなウィンドウが現れる。更に、サーバ容量を小さくしていくと3周期振動が現れ、最後には、再び安定点へと移行する。

以下では、GPによるカオス力学系の近似方法を拡張して、待ち行列理論で解析される問題にも適用できるように拡張する。GPでシステム方程式を表現する個体の表現において、その前半を関数の分子の表現に、後半を分母の表現に用いる。個体は可変長となるが、その境界を明示することにより、交叉処理における整合性を保持する。これらの概要を図5に示す。

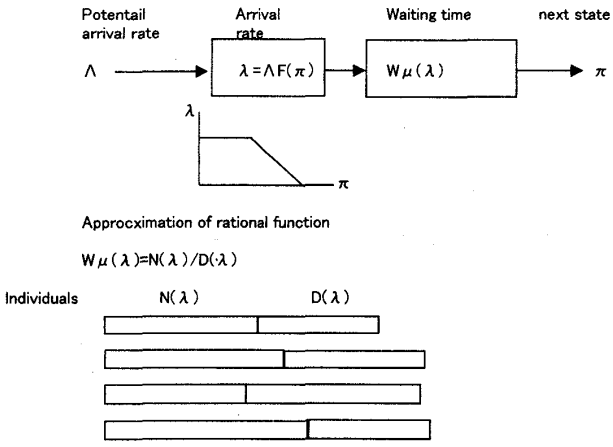



**図5　Input Pricingにおけるモデル設定**

## 6．2　カオス制御

ここで用いる方法は、直接的に推定された方程式を用いる方法であるので、どの時点からでも制御可能であり、しかも、少ないステップ数で制御が完了する利点がある。

ここで用いる制御の原理は次のようなものである。いまシステムの挙動を$x(t+1)=f(x(t))+u(t)$と仮定しておく。システムの入力$u(t)$がゼロのときの$f(x(t))$をGPにより近似し、これを$\hat{f}(x(t))$としておく。このとき、現在の状態$x(t)$から制御を始めて、次の状態が不動点に移行すればよいので$x_f=\hat{x}(t+1)=\hat{f}(x(t))+u(t)$となるように制御$u(t)$を加える（制御法－Ⅰ）。

更に、この制御法を拡張して、入力を加えることにより時系列がある一定の範囲におさまるとなるような入力を探索し、$\sum_{t=T_1+1}^{T_2}$]方法を利用する（制御法－Ⅱ）。

$$R(x_L)=[\pi(t)-x_L]^2/(T_2-T_1)\rightarrow\min \quad (25)$$

$$u(T_1)=x_L-\pi(T_1),\ T_1,T_2:\text{制御開始と終了} \quad (26)$$

## 6．3　応用例

以下では、提案するカオスダイナミックスの推定と、その制御への応用をシミュレーションをもとにして示す。

いま、サーバの特性として$M/M/1$を仮定する。

$$W_\mu(\lambda)=(\mu-\beta\lambda)/[\mu(\mu-\lambda)] \quad (27)$$

この場合の時系列の例を図6に示す。

### システムの推定

もとのシステム関数は有理関数である。従って、近似に用いるシステムの関数は有理関数により表現されると仮定し、これの近似に必要な

原始関数＋、－、＊用いる。

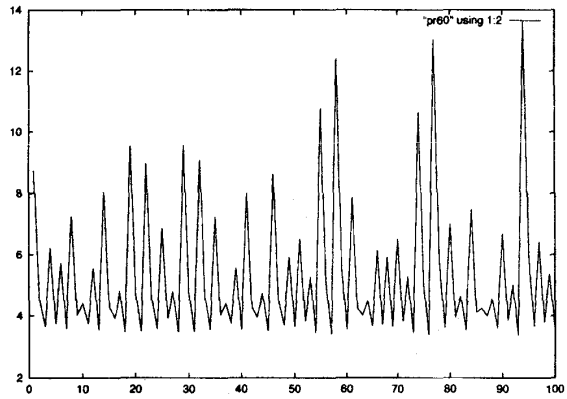



図６　時系列の例（$\mu$＝0.6）

シミュレーションの条件を、次のように設定する。推定された方程式は省略する。また、予測GPの世代数を40としたとき推定の２乗誤差は、0.0073となる。予測誤差とGP世代数の関係の例を図７に示す。

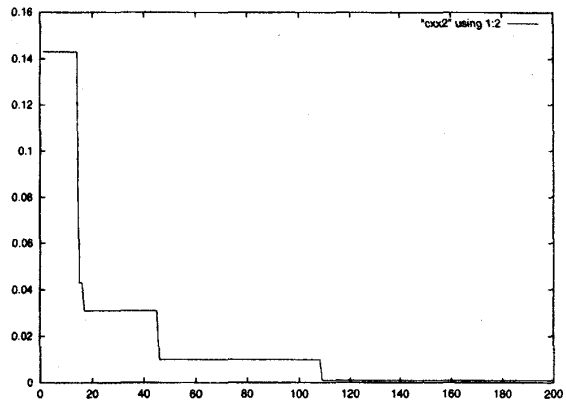


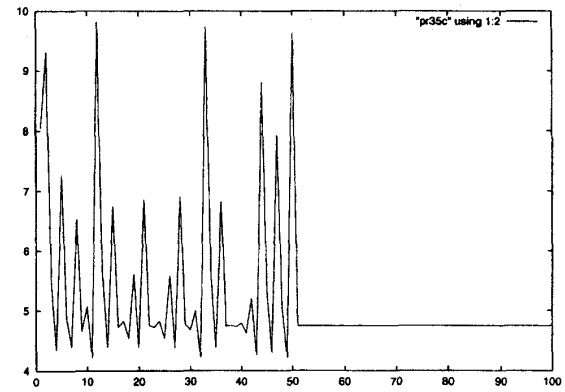

図７　予測誤差とGP世代数の関係例

**制御の例**

カオス振動の状況にあるシステムを仮定する。推定された方程式をもとに制御入力を推定しながら、均衡点へと移行させる。表５にいくつかの制御ケースを示す。$R(x_L)$, $T_s$は制御誤差と制御持続期間。図８には、カオス時系列の制御の例、この場合の入力を示す[49]。

表５　制御結果（ControlⅡ）

| $\mu$ | mean input | $R(x_L)$ | $T_s$ of stabilization |
|---|---|---|---|
| 0.35 | 4.750 | 0.0001 | 50 |
| 0.55 | 4.268 | 0.02 | 9 |
| 0.80 | 3.708 | 0.01 | 23 |



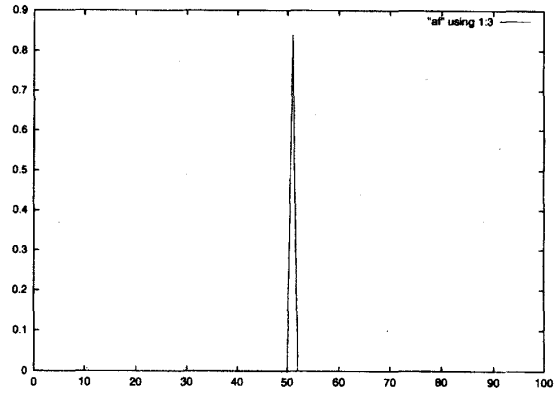




図８　カオス時系列の制御、入力の例

## ７　ワークフロー管理システムの設計とGP

### ７．１　ワークフローの表現とGP

最近、オフィス業務を効率化したり、さまざまな情報管理サブシステムの連係をとる目的で、ワークフロー管理システム(WMS:Workflow Management System)が導入されている[16]－[23]。WMSでは、ノードにおける処理能力を決定する問題のほかにフロー配分の問題を改善する課題があるが、設計段階では経験的に決められることが多い。また、システムの稼働状況を

参考にしてフロー結合や配分を変更してシステムの性能を改善する方法についても検討されているが、試行錯誤的な方法が用いられており、今後、システムが大規模化した場合への適用可能性が問題となる。

フローを再配置する方法論に関しては、例えばコンピュータネットワークにおけるトラヒックのルーティングの問題に関して、パスを遺伝的アルゴリズム（GA:Genatic Algorithm）により入れ換えて最適化する方法が提案されている。しかし、多くの場合、経験的手法で行われ、また、GAによる手法ではネットワークの結合形態を変更することは前提とされていないため最適解を探索する空間が限定される。

本章では、WMS設計を自動的に進める１つの方法として、非線形最適化問題の有効な解法である遺伝的プログラミング（GP:Gnenetic Programming）を用いる方法を提案するとともに、応用例を示す[27]。具体的には、アクティビティを入力、出力をもつ関数と考え、関数の構成順序、結合方法をGPにより最適化するものである。この場合、データベースへのアクセス順序などを制約条件として与え、これを満足し、かつ、コスト、処理時間の面で優れているシステム構成を選択する方法となっている。

本章では、GPによる関数近似において、個体表現で用いた演算子、変数を、ワークフローの構成の個体表現の特定の要素に対応させることにより、GPによるワークフロー管理最適化の手法へと拡張することができることを用いる。具体的には、アクティビティを変数に対応させ、階層構造を含むこれらの結合（フロー配置）を演算子に対応させ、遺伝的操作を用いて、より適合度の高いフロー構成を生成する。また、書類の稟議や決済に対応したデータベースアクセスを含む処理を考慮するようなフロー配置への拡張を含んでいる。

## ７．２　ワークフロー管理システムの概要

WMSの目的はビジネスプロセスの自動化とマネジメントであり、ある目的を達成するための業務の流れの自動化する管理ソフトウェアである。具体的には、ソフトウェアとして製品化される場合には、１）文書管理ソフトウェア、２）グループウェア管理ソフトウェア、３）ワークフロー管理ソフトウェアの３つに分類され、これらを少なくとも１つ含むシステムであると定義できる[19]－[23]。

市販ソフトウェアを操作レベルから分類すると、（1）データ共有型、（2）帳票回覧型、（3）業務実行型、このほかにエージェントとして実現される特定のアプリケーションとして実現されている。

WMSを記述する方法についてソフトウェア製品ごとにかなり異なり、従って、ワークフローを制御する要素の記述方法も異なっている。以下では、比較的分かりやすい製品を例にとって説明する。

（1）ducument（ドキュメント）

WMSの中を流れるフローの実体であり、電子化された形態、あるいは紙の媒体で記述された文書をさす。

（2）field（フィールド）

ドキュメントに含まれる情報であり、数値あるいは記号などのデータをさす。

（3）worker（作業者）

ドキュメント処理を実施する要員をさす。

（4）role（ロール）

作業者の実施する役割であり、作業者の分類

に応じ実施することのできるroleがグループ化されている。

(5)workbasket（作業バスケット）

処理されるべきドキュメントが格納されている論理的な箱であり、適切なworkerによる作業により処理される。

(6)operation（操作）

WMSにおいてなされる最小限の作業単位であり、ここでは、fieldに特定の値を記入することを意味する。

(7)task（タスク）

処理における一連の作業をまとめたものであり、１つの完結するワークフロー処理を記述する。

(8)sequence dependency（順序依存性）

１つのタスクがtaskに従って作業される場合、その順序をさだめるものであり、この順序が守られないと決済される前に支払が実施されるなど問題が起こる。

このような基本的な構成要素からなるWMSにおけるフローの制御を実施する方法として、ここでは、control table（制御表）、sequence constraint（順序拘束性）を用いる。

以下では、WMSを簡潔なモデルで表現し最適構成を求めていくが、用いる要素としてはノードとフローに限定し、次のように言葉を定義しておく。作業者が行う操作を、ネットワークにおけるノードとして表現する。ノードには、作業者が行う操作が役割として記述される（例えば、２つのドキュメントを入力として受けとり、データベースへの書き込みを行うなど）。このノードは１つのサービス窓口としてとらえられ、処理を待つドキュメントなどはバッファに入れられる。これが作業バスケットに相当する。

なお、最初にドキュメントを生成する（例えば、旅費申請をする）ノード処理を仮定する必要がある。以下では、これを入力ノードとよんでおく。

WMSを流れるドキュメントはフローと呼ぶことにする。フローはノードを経由しながら処理され、最終的にWMSから出ていく。これで１つのタスクが完了する。それぞれのノードにおいてフローの処理に要した時間や、WMSを通過する総時間などを推定して、WMSの性能評価に用いる。

## ７．３　制御方法について

以下では、ワークフロー管理を制御表、順序拘束性の２つの制御要素を用いて実施することを仮定する[24]－[26]。まずは、ドキュメントに対してread、writeの２つのいずれかの操作を実施することができると仮定する。更に、これにともなって、ドキュメントの状態を変更することができる操作（status change operarion）が定義される。これらの操作の定義を拡張することも可能であるので、次のような操作に限定する。

read：ドキュメントを読む。

write：ドキュメントに書き込む。

approve：ドキュメントを最終確認する。

このような操作のほかに、initialize（ドキュメントを初期化）increase（fieldの数値を増加）、enter（fieldにデータを記入する）、decrease（fieldの数値を減少）、date（日付けを記入する）、abort（ドキュメントを無効にする）、freeze（ドキュメントを凍結）、suspend（処理を中断する）、unsuspend（中断された処理を再開）などがあるが、上に述べた操作と共通する概念であり拡張は容易であるの

で、省略する。

WMSは、これらの操作を順序拘束性の表に記載された順にフローを処理する必要がある。順序拘束性の詳細は後述するが、例えば、データベースaへの書き込みを実施したあとに、データベースbからデータを読み込み、最後にドキュメントを最終確認するなどの処理の流れとして示される。

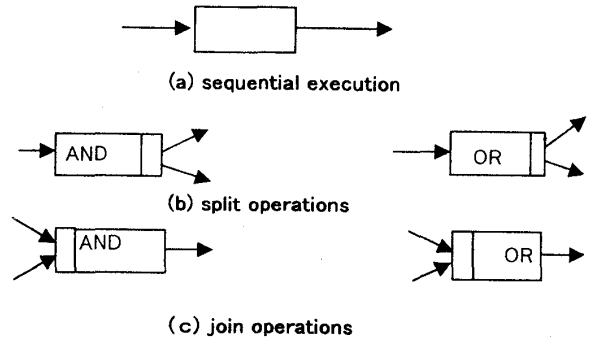



**図9　ノードとフローの関係の表現**

### 7．4　順序拘束性の記述

次の順序拘束性に関しては、routing scheme（ルーテイング方式）として定義できる、ワークフローの流れ、接続関係を記述するもので、WMSではこの順序拘束性を記述して、管理システムの目的を達成することである。ワークフロー管理システムにおけるサブシステムは、ノードの単純な縦続のほかに、分岐が考えられる。ここでは、Workflow Management Coalitionにより示されている基準を参考にして、基本要素を次のように分類する[16][17]。

(a) 単純なノードの縦続（Causality）

ノードが縦続された状態であり、フローは一方向にだけ流れる場合である。

(b) 結合処理（join）

２つ以上のフローが並列的にノードに入り、フローの処理を開始するには同期がとられる。すなわち、AND－joinにおいては２つのフローのいずれもがノードに到着している必要がある。一方、OR－joinでは２つのフローのいずれが到着していれば十分であり処理が開始される。

(c) 分岐（sprit）

ノードからの分岐の１つの形態であり、ノードにおける処理結果は並列的に複数個の部分的なフローに分割され、次のノードへと転送される。この場合、出口からの接続先となっているすべてのノードに同じフローが転送されるケース（AND－sprit）と、これらの出口に接続されるノードの１だけに転送されるケース（OR－sprit）がある。

### 7．5　ワークフローの評価と改善の方法

いくらワークフローが正確に、効率的に実行されても、企業全体として利益の貢献していない、あるいは従来の業務手順と比較して改善されていない場合には、システムの導入は成功していない。導入前との比較、評価関数の計算などの操作が必要となる。

ワークフロー管理システムの性能評価をするために、本章では、次のようなポイントについてコストなどを計測できると仮定し、これらを含む関数により評価を行うこととする。システム評価の基礎となるデータを収集する方法として、実行中のデータ収集、シミュレーションによる推定、理論計算などがある。以下では、これらのいずれかを用いると仮定しておく。

(1)目的の達成

WMSで実現しようとしている業務の目的が達成されることが最低限の要求となる。保険加入申請や旅費の請求が処理されているなどを検査

する。

(2)処理コスト

ノードは1つの操作を実施すると仮定しているので、システム全体に含まれるノードの種類、数からコストを計算することができる。具体的には、熟練した要員ほど作業に慣れていると仮定しノードの処理時間に区別を設ける。一般的な傾向としては、処理時間の短い（処理が早い）ノードの処理コストは高くなるとしておく。

(3)スループット

スループットとは、システム内部の施設などが単位時間内に処理される件数であり、どの程度利用されているかを示す指標である。

WMSが与えられた目的を遂行しているかどうかを検査する必要があるが、ここではフローを追跡する簡便な方法を用いる。最初のノードでの処理開始から最後のノードにいたるすべてのパスを求めて、これらを経由する処理において目的が遂行されるかどうかを検査する。

次に、個体の適応度の計算について述べる。すでに述べたように、1つのワークフロー管理システムの実現に対応する個体の適応度は、管理システムに含まれるロールのコスト、全体の信頼性、処理時間の期待値、スループットで計算される。ここでは、これらの単純な和として計算しておく。

## 7．6　GPの原理と個体によるワークフロー表現

以下では、WMSの構成を木構造で表現し、これをGPにおける個体とみなし最適なWMSの構成を得るためにGPを適用する。関数を木構造で表現することと、WMSを木構造で表現することの対応関係を用いると、WMSで用いるノードとフローについて次のように関連ずけすることができる。木構造による関数表現において、ノードは演算記号であり、WMSではWMSノードすなわち操作の区分である。木構造の枝（ノードをつなぐ線分）は、関数表現においては演算結果の移動であり、WMSにおいてはフローの移動を表す。

更に、関数の木構造表現における葉の部分は、関数に含まれる変数（被演算子）であり、演算の対象となる。WMSにおいてこれに対応するものとしては、WMSのシステムへの入力となるドキュメントなどであり、これらの処理要求を生成する特殊なノード（入力ノード）と考える。

このように、WMSを木構造で表現することと、関数を木構造で表現することに良好な対応関係が見られることが分かる。次に、このような木構造を、GP操作に適した形で表現する方法を考える。

同様に、WMSにおける個体ストリング表現への変換として、次のような形を用いる。例えば2つのフロー、Ｆ1、Ｆ2がノードにおいてand－joinの操作を受け、次に、この結果とフローＦ3が、次のノードでOR－joinの操作を受ける場合には、

OR－join AND－join F1 F2 F3

として、prefix表現できるであろう。すなわち、1つの操作の名前に続けて対象となる2つのフローの名前を並べて書く。ノードで処理された結果は、次のノードへの入力となる中間フローとなる。

WMSを表現する個体を解釈（すなわち、WMSのタスクの実行結果を求める）するには、個体に含まれるノードにおけるフローの組合せを検出し、解釈可能な場合には結果を求めて中間フ

ローとする。これを繰り返したあと、最後に出口となるノードの処理結果だけが残ることになる。この場合、フローに関して、処理コスト、処理時間などの評価値を計算しておく。

なお、ノードにおける操作では同期化が含まれることがあるので、入力となるフローが到着しているかどうかを確認する必要がある。そのため、時間を管理するサブシステムが必要となる。

## 7．7　個体の初期値、妥当性

GPにおいては、個体の初期値を乱数により生成する。この場合、個体はWMSを表現するものでなくてはならないので、乱数をもとに初期の個体を発生させる場合、あるいは後で述べる交叉処理により新しい個体を生成する場合には、その個体が正当なものであるかを検証する必要がある。

prefix表現された関数の形式が妥当であるかどうかを検査する方法として、***StackCount***が用いられる。これは、個体に含まれる演算子の総数から、被演算子の総数を引いた数値である。関数を表現する個体であるならこの数値は1となる。これを、WMSを表現する個体に適用する場合には、具体的には、次のように行う。個体におけるノードに対応する操作の記号を演算子に対応させ、フローの入力に相当するノード木構造の葉の部分を被演算子に対応させる。個体（ストリング）の左から検査を始めて、***StackCount***の最終的な値を求め、これがで1あるかどうかを調べる。もし1でない場合にはこのストリングを個体のプールには含めない。

更に、WMSの場合には順序制約を満足することが要求される。すなわち、最初のドキュメント入力から開始して、定められた順序でデータベースにアクセスして、最終的にapproveを受けることが求められる。WMSのフローの最終状態を監視して、これを満足するフローが全く存在しない場合には、この個体は望ましいを表現していないので、プールに含めてはいけない。

このような個体の表現の正当性を検査するには理論的な方法も考えられるが、以下ではシミュレーションをベースにした、比較的簡単な方法を用いることにする。個体表現から等価なWMSのノードとフローの関係が得られるので、シミュレーションによって入力フローを発生させ、これらが最後のノードでの処理を受けるまでの、それぞれの経過ノードにおける操作の履歴をフローと対応させながら記録しておく。この場合、結合の操作では入力となる2つのフロー（Ｆ1、Ｆ2）の記録は1つにまとめられると同時に、Ｆ1、Ｆ2の2つの記録はＦ3へと引き継がれることに注意する。同様に、分岐においては入力となるフローの記録は出口となるノードの入力フローに引き継がれる。

このように、複数の入力フローがたどる操作の記録をシミュレーションにより求め、これが順序制約を満足しているかどうかを検査する。この検査すべき順序制約を、記号の列で表現することができるであろう。この検査を実施する場合には、目的とする記号の列が、出力として得られたフローが受けた操作の履歴に含まれているかどうかを記号の一致を見ることで検査することができる。しかし、この場合、検査の目的とする記号列が1つであるとは限らず、いくつかの冗長性を含む場合がある。例えば、a、bをデータベースＡ、Ｂへのアクセスとすると、これらの継続アクセスは、Ｐ＝(a, b)であるが、データベースへの2回のアクセスを含むＰ＝

(a, a, b)も順序制約を満足する。

このように、基本となる制約順序をベースとして派生的なフローが受ける処理の履歴があり得る。これを識別する方法として、パターン認識などの方法が考えれる。しかし、ここでは問題の範囲も限定されているので、簡便な方法を用いる。すなわち、シミュレーション実験では、このような発生可能な記号のパターンをあらかじめ生成しておいて、これらが記録されたフローの経過した履歴に含まれているかどうかを検査する方法をとっている。

このようにして１つのWMSを表現する個体のそれぞれについて、シミュレーションを基本としてワークフローの処理能力を推定する。この処理能力の評価としては、フローが最終処理されるまでの平均時間、システムのコスト、スループットなどがあり、これらを総合評価する。１つの個体に対するこの評価値は適合度とよばれ、適合度の高い個体が遺伝的操作を実施する候補として選択される。

## ７．８　書類決裁の例

以下では、シミュレーションによる数値例をもとにしてGP適用によるWMS改善の例題を示す。最初の例では旅費請求などやや簡単な例を対象としている。シミュレーションの条件を次のように設定する。

個体数：1000

個体のデータ長：25

突然変異確率：0.05

ノードの処理時間：平均が３～８の範囲にある指数分布

評価関数：待ち時間の逆数＋コストの逆数＋信頼性

なおコストの計算は簡単化のためノード数（入力ノードと処理ノード）に比例すると仮定するが、この数値が一定値（入力ノードと処理ノードに対してそれぞれ$N_1$、$N_2$を超えるまではゼロでこれを超えてからはその超過数をコストとする。

個体数は、探索する範囲のバリエーションを考慮して、1000程度とする。フローを処理するノードは作業をする作業者の仕事に対応する。次のような３つのケースについてシミュレーションを行なう。

(case　1)read a, read b, read c

(case　2)read a, read/write b, write c

(case　3)read/write a, read/write b, write c

図10には個体の適応度の最大値と、GPの世代数との関係を示している。これより分かるように、約130世代において、適応度の最大値がほぼ最終的な適応度の値と同じとなる構成が得られており、効率的にワークフロー管理システムの最適構成を与える、自動化システムとなっていることが分かる。

表６には、それぞれのケースにおける最適化の最終結果を示している。$f_0$、$f_T$は、それぞれ、populationの中での適合度の初期値と最終値である。

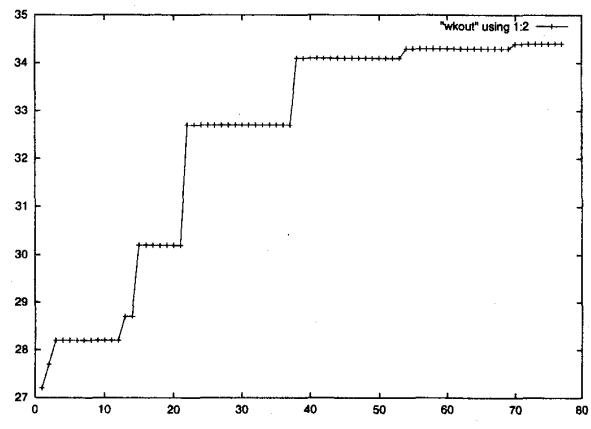



**図10　適応度の最大値とGP世代の関係**

**表6　最適化の結果概要**

| case | $N_1$ | $N_2$ | $f_0$ | $f_T$ | $N_P$ |
|---|---|---|---|---|---|
| case 1 | 5 | 5 | 17.2 | 35.6 | 67 |
| case 2 | 8 | 6 | 17.2 | 34.5 | 121 |
| case 3 | 10 | 8 | 17.1 | 33.5 | 180 |

## 8　プロダクションルールの生成とGP

以下では、与えられたデータから、これを分類したり、そのデータの中に含まれる規則を記述するプロダクションルールを遺伝的プログラミング（GP:Genetic Programmingを用いて生成する方法について述べる。その具体的な応用として、株価予測の規則の生成と、企業が発行する債券の格付けを実施するシミュレーションを行う。

与えられたデータから、それを記述する方法論は、現在ではdata miningなどの手法として知られるようになっている。この方法は、多量に格納された販売データなどから、どのような環境のもとで特定の商品が売れるかなどの傾向を、ルールとして発見するものであり、data warehousingなどと同様に、今日のマーケティングの主要な手法となっている。

これまでの研究においても、数値的な方法ではなく、いわゆる記述的な方法でデータに含まれる規則性を検出する方法が提案されている。すなわち、決定木などと呼ばれる方法で、分類や意思決定の規則（ルール）を記述するものである。その代表的な方法として、1つは、ＩＤ３があり、データがいくつかのグループに分類されると仮定した場合に、分類の方法による整合性の度合いをエントロピーとして計測し、これが低下する方向にグループ化を進める方法である[28]－[31]。最初の段階では、カテゴリカルデータの分類だけに利用されたが、その後、数値的なデータも含む場合まで拡張され、これらカテゴリカルデータ、数値データが混在するケースまで取り扱えるようになっている。

もう1つの方法は、inductive learning（演繹学習法）と呼ばれる方法であり、統計的な手法を基礎として、分類を行うための決定木を生成する方法である。この方法では、分類の主要な要素となる変数から利用する基準をもうけておき、分類が進行するにつれてグループ分類の整合度が向上するように工夫されている[28]。

これらの方法により、数値的な分類ではなく、プロダクションルールの形式で、いわば、人間が言語として理解できるルールを生成することができる[31]－[32]。しかし、これらにもいくつかの問題点がある。その1つは、ルール生成の仮定で、ルールが複雑になることを制御できないことであり、分類の精度が良くても、そのルールをもはや人間が理解できないケースが発生する。

また、もう1つの問題は、ルールの中に算術式などを取り入れることが困難な点である。データはあらかじめ前処理を施されている必要があり、分類を進めながら、同時に、その分類を正確にするための算術式を生成することはできな

以下では、これらの点を克服するために、GPを用いてルールを生成する方法を提案する[33]。具体的には、最終的なルールの候補となる複数のルールを準備しておき、これらを木構造で表現した場合に、性能が優秀である木構造どうしに対して遺伝的操作を実施することにより、ルールを改善する方法である。この過程で、複雑なルールが生成されることを抑制することができる。また、ルールには計算を補助する意味

での算術式を含ませることとし、これらを同時に最適化する。

## ８．１　プロダクションルールの記述

推論過程を表現する方法として、いくつかが考えられるが、プロダクションルール（以下では、簡単のため、単にルールとよぶ）は最もよく用いられる形式である。これは、一般に次のように書かれる。

```
if A then B
```

ここで、Aは条件部とか前件部とよばれ、Bは結論部とか後件部とか呼ばれる。すなわち、Aに書かれた条件が満足された場合に、Bに書かれたことが成立することを主張するものであり、Aはデータの特性の判断部分で、Bはこれにより成立する結論の記述である。Bとしては、例えば、このサンプルが特定のグループに属しているとか、株価が将来は上昇するなどを表現する。結論部には、一度に複数の結論を書くことも可能である。しかし、以下では、原則として結論を１つに限定し、その結論ごとにルールを作成すると仮定する。

例えば、グループ１に属するかどうかを判断するルールと、グループ２に属するかどうかを判断するルールを別に作成する。ルールの条件部の記述については、基本的には論理式を用いるが、この論理式の項目には単なる変数や定数のほかに、算術式も含むと仮定する。すなわち、条件部は、次のような形をしている。

```
C1 O1 C2 O2 C3 O3...
```

ここで、Ciは論理式であり、Oiは論理記号である。論理式と論理記号が繰り返し出現する形式として条件部が記述される。条件部に含まれる論理演算には、論理演算子の優先順位などを考慮しながら、複数の被演算子を含むような一般的な記述が可能であるが、以下の議論では、論理演算子の対象とする被演算子は２つに限定している。すなわち、２項演算に還元された形式で表現されていると仮定する。これを考慮すると、論理式は、次のように書くことができる。

```
A1 H1 A2
```

ここで、Aiは算術式であり、Hiは比較演算子である。算術式には、単純な変数や定数を含んでいる。例えば、次のようなルールを書くことができる。

```
if x1>1 and x2<9 then rise
```

このルールの意味することは、株価を特徴づけるパラメータであるｘ１およびｘ２が、それぞれ１以上、９未満であるときには、株価は将来上昇するであろうという推論を示す。

ルールを構成する方法として、学習による方法を用いる。最初に、すでに結果が分かっているデータを学習用のデータとして準備し、これに適合するようにルールを更新していく。具体的には、論理記号を変更したり、論理式の形をかえること、更には、論理式に含まれる算術演算を変更するなどの方法で実施する。これらを試行錯誤的に実行しても、効率的ではないし、最終的に適切なルールが得られる保証はない。しかし、GPを用いることにより、着実にルールの改善を実施することができる。この学習の過

程で、生成した複数のルールの性能を評価することが必要となる。この性能評価のために、再び学習用のデータを用いる。検査すべきデータにこれらのルールを適用して、得られるべき結論と同じである比率が全体の学習サンプルのどの程度であるかにより判断する。

## 8．2　GPにおける個体表現

GPによりルールを最適化するには、最初にルールを個体として表現する必要がある。GPにより関数を近似する場合には、関数を木構造で表現するため、個体はこれに対応する形式を用いることができる。また、その個体に含まれる要素は、四規演算子などの原始関数と時間遅れを含む変数であり、比較的単純である。

しかし、ルールの場合には、論理演算子の形は限定されるが、被演算子には変数のほかに、変数を含む算術演算も可能である。従って、一般的な表現を許すと、個体表現は極めて複雑となり、その結果として適用するGPも、さまざまなケースを想定した方法となり、複雑化する。以下では、個体を表現するデータ構造を、やや簡単化することにより、このような問題を回避している。

次のような前提を置く。

(1)２項論理演算に限定する

論理演算を記述する場合に、その論理演算の対象とする被演算子は２つに限定する。すなわち、複雑な論理演算は、被演算子２項からなる論理式で記述される。これにより、一般性は失われない。

(2)被演算子は比較演算子を１つ含む

これは、通常の前提条件であるが、すでに論理演算は２項目演算に限定されているので、被演算子は２つの算術式と、１つの比較演算子を含む形式となる。

(3)算術式は４則演算と変数、定数からなるprefix表現

この前提も、それほど厳しい制約ではない。比較演算子により比較される２つの算術式は、通常の方程式や関数を表現する方法と同様であり、GPを適用する場合に便利なように、あらかじめprefix表現へと変換しておく。以上のような前提のもとで構成される個体の形式の概略を図11に示す。なお、この図で示すように、個体表現には階層構造を用いることが便利であり、ルールの最上位を示すルールの全体構造、論理演算の表現を第１階層の個体（これを、レベル１の個体プールとよぶ）としておき、論理演算の対象となる算術式は別の個体として表現し、プール（これを、レベル１の個体プールとよぶ）を構成しておく。

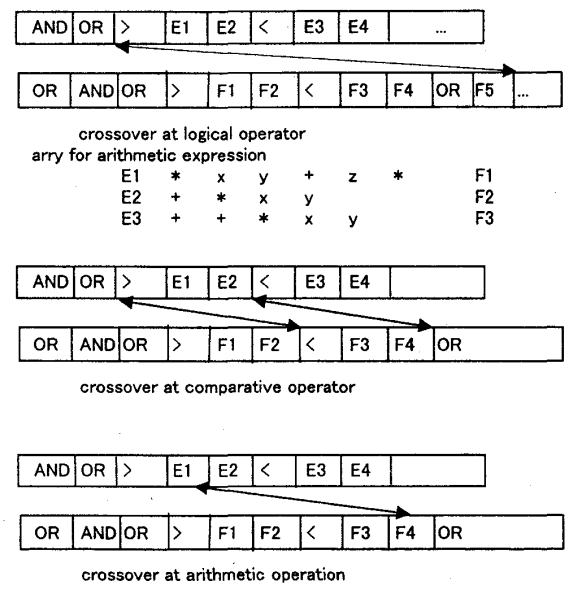



**図11　ルール表現におけるprefix表現**

図11に示すように、レベル１においては、論理記号と論理式の並びが書かれ、これらが識別できるような個体表現となる。ここで、論理式

の全体を示す必要はない。図11に示すように、論理式には、１つの比較演算子と２つの算術式が含まれ、しかも、算術式は２つに限定されている。従って、レベル１における個体表現における論理式の表現では、論理記号の種類だけが区別されておればよい。従って、Ｌ１、Ｌ２などは比較演算子と考えてもよい。

次に、算術式を表現するレベル２の個体については、図に示すように、木構造と等価であるprefix表現を用いる。この表現はプログラム言語をコンパイラで解釈する場合に用いる内部表現と等価である。このprefix表現を解釈して算術式の結果を求めるには、変数に値を代入しながらスタックを用いて計算することにより実施できる。

## ８．３　個体の値の計算

個体は論理演算の結果として求まる論理値である。変数に値を入れて、それぞれの個体の示す論理演算を実行することにより、個体の値を計算することができる。個体は１つのルールに対応しているので、この個体により計算される結果が学習のパターンに一致しているほど、大きな適合度をもつことになる。

個体の値の計算は次のように行われる。

(１)算術式の値の計算

最初に、レベル２の個体である算術式を解釈する。学習サンプルとした与えられたデータには、変数の値が示されているので、これを変数に代入する。定数はGPの過程で更新されるが、算術式の計算のときには、１つの定まった値であるので、そのまま用いる。

(２)論理式の解釈

算術式の値をレベル１の個体に代入したあとで、それぞれの論理式の値を計算する。比較演算子と、これが対象とする算術式があり、算術式の値はすでに分かっているので、論理値が求まる。

(３)レベル１の個体の解釈

それぞれの論理式の値が計算されるので、個体の取りうる最終的な結果が計算される。これは論理値となる。この論理値が、学習サンプルに示された論理値と同じであるかどうかを検査しておき、この個体の適合度の計算に用いる。

## ８．４　交叉処理の方法

これまで述べたように、個体にはレベル１とレベル２の２つの表現があり、このどちらにも遺伝的操作を加えるほうが、最適化の効率や性能が向上すると考えられる。すわわち、適合度の高い２つの個体のもつレベル１の構造とレベル２の構造を交叉処理により、相互に交換することが子供（offspring）の能力を高めることになる。

このような操作を可能にするために、以下の３つのケースに応じて、異なる交叉処理をほどこすことにする。

いま、２つの個体が存在すると仮定し、これらの個体のレベル１の表現をR1、R2としておき、レベル２の表現をS1、S2としておく。交叉処理では、基本的に、一方の個体、例えば、R1、S1のランダムな場所Ａを選択して、ここで個体を切断し、個体R2、S2の後半の部分との交換処理を行う。この場合、個体R2、S2については、交叉点を任意に選択することはできない。交叉処理のあとで意味のある個体になる必要がある。この検査方法はのちほど述べることにし、ここでは、個体R2、S2の交叉点がＢとして定まって

いると仮定する。

交叉処理を行う場合の細かな点は、あとで詳しく述べるが、概略の処理手順は次のようになる。これを図11に示す。

(1)個体R1の交叉位置Aが論理記号の場合

論理記号が交叉の境界となるから、個体R2の交叉位置であるBは、論理企業の直前になるように選択する。ただし、交叉処理により生成される2つの子ども（offspring）が矛盾なく論理演算を表すためには、次のような制約がある。

このような制約を満足するような交叉点Bを選択する。一般には、このようなBが存在しない場合もある。この場合は、交叉は実行しない。また、Bが複数存在する場合には、その中から任意に選択する。

(2)個体R1の交叉位置Aが論理式の場合

次の3つのケースに分けて実行する。ケースの選択は確率を与えておく。

(2－1)算術式の入れ替え

論理式が交叉点になるので、論理式を交換する処理を行う。具体的には、論理式は、すでに述べたように、比較演算子を挟んだ2つの算術式であるので、これらの算術式を相互に入れ換える操作を行う。いま、固体R1の交叉点Aの位置にある論理式をLAとし、その論理式に含まれる2つの算術式を、E1、E2とする。同様に、個体R2の交叉点にある論理式に含まれる算術式をG1、G2とする。このとき、交叉処理として、E1、E2とG1、G2を相互に入れ換える。なお、個体R2の交叉点はランダムに選択するものとする。

(2－2)算術式の部分的な交換

論理式が交叉点である場合に、(2－1)のように論理式に含まれる算術式を全部入れ換えるのではなく、1つだけを入れ換える。(2－1)で説明した例では、算術式E1と算術式G1だけの入れ替えを行う。

(2－3)算術式の上でのGP

この場合は、通常の方程式における近似問題などと同様に、2つの個体R1、R2から任意に選択した2つの算術式（これらはレベル2の個体である）に対して交叉処理を行う。2つの算術式に対する交叉処理を実行するには、任意の交叉点を選択することでは実現できない。そのため、***StackCount***とよばれる演算子と被演算子の数を数えながら、その差に注目して交叉点を決定する。

突然変異には、次の2つの種類がある。

グローバル突然変異

木構造として表現された論理式の部分木S1を、生成された部分木S2で置き換える操作である。この操作は、部分木S2が任意に生成された論理式であることを除けば、すでに述べた(1)の論理記号における交叉処理と同じである。

ローカル突然変異

この操作は、論理式の任意の位置の記号を、同じ種類であるが異なる記号で置き換える操作である。例えば、個体R1のランダムに選択された位置における論理記号がANDであった場合、これをORに置き換える。選択された位置が論理式である場合には、次の3つのケースに分けて考える。

(a) 比較演算子を別の比較演算子に置き換える
(b) 算術式の原始関数を別の原始関数に置き換える
(c) 算術式の変数を別の変数で置き換える
(d) 算術式の定数を別の定数で置き換える

応用例として、企業の発行する債券の格つけを財務指標を入力として予測する問題、あるいは株価の観測データを用いて将来を予測する問

題がある[34]－[38]。これらについて、GPによるルールを推定する方法により良好な結果を得ている[33]。ここでは詳細は省略する。

## 9　CNNによるリスク拡散分析

### 9．1　CNNによるダイナミックスの表現

ニューロンの平面的、立体的な結合によりシステムを記述する方法としてCNN(Cellular Neural Network)が提案され、カオス的な挙動を含めて解析されている[40]－[45]。CNNの工学的な応用として画像処理が示されているが、進行波を記述するモデルとしても興味あり、例えば、インターネットを介した電子商取引における決済リスクの拡大など、経済社会におけるモデル分析にも有効である[45][9]。

これまでCNNのモデル化では、あらかじめシステム方程式が与えられていることが仮定され、拡散係数と信号の伝搬の問題が議論されているが、一般には、システムからの観測信号だけが与えられて、システム方程式が未知の場合が多く、関数近似により方程式を推定する必要がある。すなわち、実際にデータが与えられた場合に、これを生成するCNNを推定する、いわゆる逆問題を解くことが必要となる。

CNNにおけるシステム方程式をGPにより近似するために、初等演算のほかに区分線形などの関数を準備し、これらと変数、定数を含む木構造によりGPにおける個体を定義する。GPにおいては適合度（fitness）の大きい２つの個体を選択し、適切に定めた位置で個体の交叉処理を行ない、より関数近似能力の高い個体を生成していく。このような方法により、カオス的特性を示すCNNのシステム方程式を含めて近似を行なうことができる。

更に、GPを用いてシステム方程式の近似が求められていることを利用し、フィードバック制御によるカオス同期化法に従ってカオス状態から安定状態へ移行させる方法を提案する。この結果を用いることにより制御入力を推定でき、適切なフィードバック入力を加える同期化制御により、短時間で均衡レベルに制御することができることを示している。

CNNはChua教授等により、非線型素子の回路網により偏微分方程式の体系を表現する方法として提案され、初期の段階では連続時間における定式化がなされ、現在ではコンピュータ処理に適した離散モデルも提案されている。

CNNは１次元、２次元空間において偏微分方程式として表現されるダイナミックスを、セルオートマトンによる表現にかえたものであり、画像処理や図形パターン分類などに応用され、また、そのカオス的な挙動についても研究されている[42]－[44]。CNNにより、これまでの進行波を統一的に表現できるなどの利点は知られている。

CNNには、外部からの信号を加えることを前提とした非自律型CNNと、外力を加えない自律型CNNの２つの類型がある。更に、これらのCNNを記述するダイナミクスを１～３次元空間のいずれで記述するか、ダイナミックスを記述する方程式の数をいくつにするかにより、モデルの形が分類される。

これらの詳細の区分は省略し、以下ではネットワークにおける拡散のモデル化に適したものだけを考察する。セルが格子状に結合した自律的CNNにおいて、空間的な拡散(spatially discrete relation)を考慮した、次のようなモデルを導入する。

$$dX_{ij}/dt=f(X_{ij})+D_{ij}\nabla^2 X_{ij} \quad (27)$$

ここで、変数のベクトルは$X_{ij}=(u_{ij}, v_{ij}, ..)$であるとし、変数$u_{ij}$、$v_{ij}$、..はセル$c(i,j)$の内部状態を表す（添字$i$、$j$は2次元平面における座標）。セルどうしの結合係数はなくなり、拡散の項目だけが考慮される。なお、変数$u_{ij}$、$v_{ij}$、..は、ダイナミックスを記述するのに必要な方程式の変数に対応している。ここで、ラプラシアン$\nabla^2$を偏微分の近似で置き換える。例えば、次のようになる。

$$\nabla^2 u_{ij}\rightarrow u_{i+1\,j}u_{i-1\,j}+u_{i,j+1}+u_{i,j-1}-4u_{ij} \quad (28)$$

## 9．2　CNNによるモデル化の例

CNNのダイナミックスを記述する方程式（状態）を3つに限定した場合を考え、次のような具体例を考察する。

$$du_{ij}/dt=\alpha(v_{ij}-A_u u_{ij}-g(u_{ij}))+ D_u(u_{i+1\,j}+u_{i-1\,j}+u_{i,j+1}+u_{i,j-1}-4u_{ij}) \quad (29)$$

$$dv_{ij}/dt=u_{ij}-v_{ij}+w_{ij}+ D_v(v_{i+1\,j}+v_{i-1\,j}+v_{i,j+1}+v_{i,j-1}-4v_{ij}) \quad (30)$$

$$dw_{ij}/dt=-\beta v_{ij}+D_w(w_{i+1\,j}+w_{i,j-1}+w_{i,j+1}+ w_{i,j-1}-4w_{ij}) \quad (31)$$

$$g(x)=s_1 x+0.5(s_0-s_1)(|x+1|-|x-1|) \quad (32)$$

このモデルは、$u_{ij}$、$v_{ij}$、$w_{ij}$のそれぞれの変数に関して、近隣のセルへの流出（流入）を考慮したモデルであり、セルどうしが、より緊密な提携関係にあるケースを想定している。このモデル（$A_u=1$）をCNN－3とよんでおく。経済モデルでは、例えば、企業や個人などの主体をセルであると考え、変数$u_{ij}$、$v_{ij}$、$w_{ij}$をそれぞれ資本、労働、商品在庫（あるいは単に商品）に対応させ、これらによりその挙動が記述されていると仮定する[10][45]。

更に、式(28)－(32)に示すモデルにおいて、次のように設定したモデルをCNN－1とよんでおく。

$A_u=0, D_v=0, D_w=0$

$$g(x)=0.5[(s_1+s_2)x+(s_0-s_1)(|x-B_1|-|B_1|)+(s_2-s_0)(|x-B2|-|B2|)]+\varepsilon \quad (33)$$

このモデルは、$u_{ij}$にだけ周辺への拡散を仮定しており、式(33)で記述されるCNNは、Chua circuitとよばれている。

## 9．3　GPによるシステム推定の例

以下では、GPによる関数近似の能力を検証するために、すでに述べたの2つのCNN類型について、関数形を推定する[10][45]。まず、CNNによる信号を発生しておいて、特定のセルにおける観測信号とする。この信号をもとにして、GPによりCNNのシステム方程式を推定し、もとの形と比較することにより性能を評価する。推定のシミュレーションの条件は次のようである。

個体プール規模：1000

配列の最大長：変数$u$、$v$、$w$に相当する部分を合計して100以下

最大世代数：500

データ長：20

関数に含まれる演算子:＋、－、×、/、||

$P_M=0.02$

具体的に計算する過程より分かるよう、方程式を解くにはあるセル$c(i,j)$の周辺のセルにおいて発生した変数の値を用いる。従って、例えば２次元のCNNの場合には、100×100の全体のセルについて微分方程式を解いていく必要がある。拡散の係数はGAにより推定されるべき定数となる。

以下のシミュレーションでは、それぞれのCNNについて次のようなパラメータを設定している。

（CNN－１）

$\alpha=9$，$\beta=30$，$s_1=s_2=2/7$

$s_0=-1/7$，$\varepsilon=1/14$，$B_1=-1$，$B_2=1$

（CNN－３）

$\alpha=9$，$\beta=19$，$s_0=-1.143$、$s_1=0.714$

また、シミュレーションの条件については、本論文を通じて次のように仮定する。

解析手法：ルンゲクッタ法

きざみ幅：$h=0.01$

最大ステップ数：100

なお、各個体の適合度を計算するためにルンゲクッタ法の数値積分を行う必要があり、データ数が多い場合には計算時間や計算誤差が問題となる。しかし、GPによる関数近似の利点として、比較的少ないデータ数で十分な近似精度が得られることがある。ここで示したアルゴリズムは、データ数を増加させる必要がある場合に発生する計算時間や計算誤差に関連する問題を回避できる１つの方法ともなっている。

推定の誤差について、次のような式で定義する。

$$n\text{-}rmse=[\sum(x(t)-\hat{x}(t))^2/N_s]^{1/2}/\sigma \qquad (34)$$

ここで、合計は対象とするすべてのセルについて実施し、$\sigma^2$は観測された信号の分散である。

図12には、GPにおける世代数と関数近似の収束の例（$n\text{-}rmse$）を示している。このように、ほぼ130回のGPにより良好な近似を得ている。表７には、このようなシミュレーション実験の結果を示している。表７には、同時に、関数近似により得られる関数形が、ほぼ最終的な形と一致するまでのGP世代数も示している。

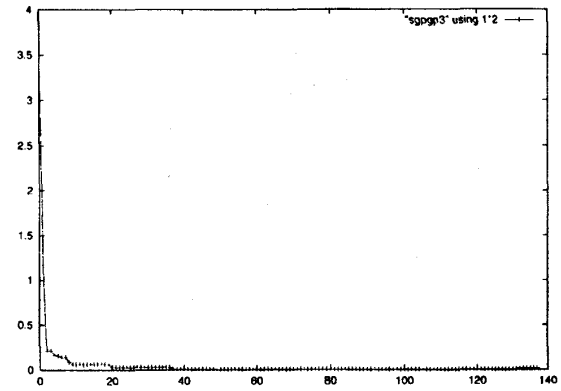



**図12　関数近似誤差とGPの世代数**

**表７　近似誤差**

| name | n-rmse | $N_p$ |
|---|---|---|
| CNN-１ | 0.0015 | 134 |
| CNN-３ | 0.0021 | 153 |

## ９．４　カオスとフィードバック制御

カオス力学系において、ストレンジアトラクタ内に不安定軌道が存在する場合、適切な外力を加えて、安定軌道へと移行させる制御方法が存在する。離散時間システムを対象としたものとしてはOGY法がよく知られ、連続時間システムではPyragasにより提案されたフィードバック制御法がある[13]－[15]。この方法は、一般には周期の不安定軌道を安定化するために用いられるが、均衡点への移行にもそのまま用いることができる[10]。

カオス制御の目的は軌道の安定化であるが、

以下では、不動点あるいはリミットサイクルへの移行（いわゆる同期化：synchronization）について考察する。連続時間システムの場合には、そのシステム方程式$dX(t)/dt=f(X(t))$が得られていれば、目標値との差をフィードバック制御入力として加えることにより、同期化が可能である。目標とする不動点あるいはリミットサイクルを$\hat{X}(t)$とした場合に同期化の制御は$K[\hat{X}(t)-X(t)]$を微分方程式の右辺に加えたフィードバック制御により可能である[19]。

$$dX(t)/dt=f(X(t))+K[\hat{X}(t)-X(t)] \qquad (35)$$

ここで、$K$はフィードバックゲインである。しかし、一般には観測データしか与えられていないため、システムは未知でありフィードバック制御はできない。

しかし、われわれはGPによりシステム方程式の近似形を推定しているので、これを用いた制御が可能となる[20]。また、システム方程式を推定しているので、ゲインを適切に設定することができ、収束を早めることができる。

次のようなダイナミックスを考える。

$$dx(t)/dt=f(x(t))+u(t) \qquad (36)$$

ここで、関数$f(x(t))$は、式（33）におけるセル自身における変数、および周辺のセルにおける変数を含んだ関数である。入力$u(t)$は通常はゼロであるが、不動点あるいはリミットサイクル$\hat{x}(t)$への移行を行う場合に適切な入力$u(t)$を加えることを意味する。特に、入力$u(t)$が次のような形を仮定する。

$$dx(t)/dt=f(x(t))+\lambda[\hat{x}(t)-x(t)] \qquad (37)$$

入力がゼロであるシステム方程式を、GPを用いて推定しておいて、これを$\hat{f}(x(t))$としておく。これらをアルゴリズムとしてまとめると、次のようになる。

（ステップ１）不動点あるいはリミットサイクルの検出

いま、関数の形が推定されているので、これを用いて不動点あるいはリミットサイクル$\hat{x}(t)$を推定しておく。

（ステップ２）制御入力を加える

時刻tに$\hat{x}(t)$に移動するには、フィードバックゲインλを小さい値から一定の間隔で増加させながら

$$dx(t)/dt=\hat{f}(x(t))+\lambda[\hat{x}(t)-x(t)]\text{の結果}\rightarrow\hat{x}(t)) \qquad (38)$$

となるように決めればよい。$x(t)$が$\hat{x}(t)$に近くなると入力$u(t)$は小さくなり、制御が完了すればゼロになる。

（ステップ３）制御の終了の判断

制御を終了させるのは、目的としている均衡点、リミットサイクルと、現在の状態との誤差が、ある設定したしきい値より小さくなった時点とする。状態が不動点、リミットサイクルに移行しない場合には、ステップ２へと戻って制御を繰り返す。

## ９．５　同期化の例題

以下では、CNN－１、CNN－３における制御の例題を示す[10]。式（7）に示すCNN－１においては、すでに述べたように１つの安定状態が存在する。また、式(3)－(6)に示すCNN－３において、拡散項をなくしたシステム方程式はリ

ミットサイクルをもつことが示されている。これらを同期化の目標として採用する。

CNN－１について、制御される前、および制御後のセルの状態を示したのが図13、14、15である。ここで、セルは10×10の範囲で２次元的に配置されているが、この図13、14、15では１列にならべて表示している。最初の図13では、セルの初期状態としてランダムな数値を与えた場合を示している。

次の図14は、セルにおける演算が進行して、セルの固まりごとに、特徴的なパターンを形成している状況を示している。乱数とした与えられた初期状態よりは、まとまりのあるパターンとなってはいるが、不安定な状態である。

最後の図15は、入力印加による制御を実施し安定化させた例であり、このシミュレーション

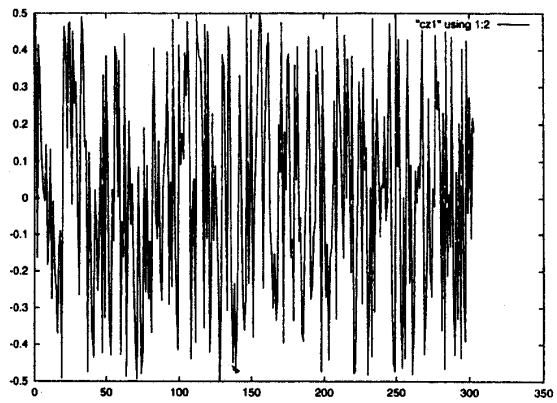



**図13　セルの初期状態**$(u_{ij}, v_{ij}, w_{ij})$

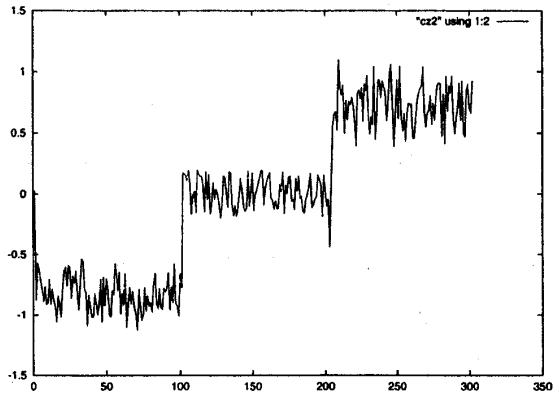



**図14　一定の時間後のセルの状態**$(u_{ij}, v_{ij}, w_{ij})$

においては、制御をはじめて約30ステップ目に収束をしている。これより分かるように、ほぼ、目的とするレベルに制御できている。

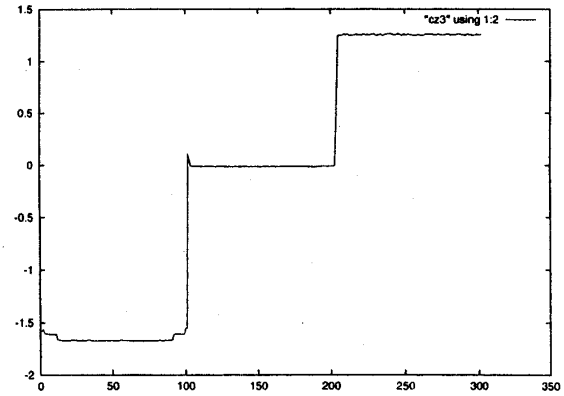



**図15　同期化後のセルの状態**$(u_{ij}, v_{ij}, w_{ij})$

**表８　制御に要するステップ数**

Table8. Result of steps necessary for control

| name | $r_c$ | $N_c$ |
|---|---|---|
| CNN-１ | 0.05 | 6 |
| CNN-３ | 0.08 | 7 |

これを数量的に示すために、制御を開始して以降の状態$x(t)$が、最終的な目標$\hat{x}(t)$にどの程度近くなるかを、次の式で定義する。

$$r_c=[\sum(\hat{x}(t)-x(t))^2/N_c]^{1/2}/\sigma \qquad (26)$$

ここで、$x(t)$は時刻$t$における状態であり、$\hat{x}(t)$は目標としている最終状態である。また、$N_c$は同期化制御に必要なステップ数、$\sigma^2$はデータの分散である。表８には、これらの$r_c$と$N_c$とを示している。

なお、セルのすべてを同期化するのではなく、例えば、$I$おきのセルについてだけ同期化の信号を加える制御方法がある[19]。このような制御についても実施している。この結果を簡単にまとめると、制御が完結するための最小限必要

な同期化制御を加えるべきセル間の間隔$I$は、次のように推定される。

CNN－1 : $I=3$

CNN－3 : $I=4$

## 10 むすび

本論文では、不均衡モデル分析を含めて複雑系理論の提唱する経済モデル分析について、特に遺伝的プログラミング(GP:Genetic Programming)について、これまでの著者の研究と今後の関連性について述べた。具体的には、GPの原理、システム方程式近似と制御、経済動学モデルとカオス、Input Pricingのシステム推定へのGPワークフロー管理システムの設計、プロダクションルールの生成とGP、CNNによるダイナミックスの表現とGPである。

今後、これらの個別分野における解析と応用を進めると同時に、エージェントシステムによる経済社会現象のモデル化、例えば株価や景気変動について考察を行なっていく予定である。

## 参考文献


[1] J.R.Koza:Genatic Programming,MIt Press, 1992

[2] J.Koza: “Genetic programming:A paradigm for genetically breeding populations of computer programs to solve problems”,Report No.STAN-CS-90-1314, Dept.of.Computer Science Stanford University,1990

[3] J.Koza: “Evalution and subsumption using genetic programming”,Proc of the First European Conference on Artificial Life, MIT Press,1991.

[4] J.Koza and J.P.Rice: “Automatic programming of robots by using genetic programming”, Proc. of 10th AAAI,1992.

[5] H.Iba:Genetic Programming (in Japanese) ,Denki-dai Shuppannkai,1996.

[6] M.J.Keith and M.C.Martin: “Genetic programming in C++: Implementation issues”,in (ed) K.E.Kinnerar,Jr., Advance in Genetic Programming MIT Press,1994.

[7] J.R.Koza:G1natic Programming, MIT Press,1992

[8] Y.Ikeda and S.Tokinaga: “Approximation of chaotic dynamics by using smaller number of data based upon the genetic programming”, Trans.IEICE,vol.E83-A,no.8,pp.1599-1607,2000

[9] Y.Ikeda and S.Tokinaga: “Controlling the chaotic dynamics by using approximated system equations obtained by the genetic programming” Trans.IEICE,vol.E84-A,vol.9,pp.2118-2127,2001.

[10] 時永祥三、矢加部正幸: “遺伝的プログラミングを用いたCNNによる拡散モデルの近似とその同期化への応用”、電子情報通信学会論文誌,vol.J85-A,採録決定済,2002.

[11] C.M.Rump and S.Stidham,Jr: “Stability and chaos in input pricing for a service facility with adaptive customer response to congestion”,

Management Science, vol. 44, no. 2., p. 246-261, 1998.

[12] E. Otto, C. Grebogi and J. A. Yorke: "Controlling chaos", Physical Review Letters, vol. 64, no. 11, pp. 1198-1199, 1990.

[13] K. Pyragas: "Continuous control of chaos by self-controlling feedback", Physics Letters, A, vol. 170, pp. 421-428, 1992.

[14] H. Nakajima, H. Ito and Y. Ueda: "Automatic adjustment of delay time and feedback gain in delayed feedback control of chaos", Trans., IEICE, vol. E80-A, no. 9, pp. 1554-1559, 1997.

[15] H. Mori, T. Ushio and S. Kodama: "Controlling chaotic discrete-time systems via nonlinear feedback (in Japanese)", Trans. IEICE, vol. J78-A, no. 7, pp. 798-805, 1995.

[16] Workflow Management Coalition: "The workflow reference model", Document No. WfMC-tC-1006, 1994.

[17] Workflow Management Coalition: "Terminology and Glossary", DocumentNo. WfMC-TC-1011, 1996.

[18] 松井一郎: "業務の連携を自動化、時間短縮と管理を実現、ワークフロー管理ソフトが日本でも利用可能に、日経コンピュータ、1994年5月2日号, pp. 57-67, 1994.

[19] 垂水浩幸、金政ふじ、小笠原章夫: "ワークフロー技術とその応用"、計測と制御, Vol. 34, No. 12, pp. 932-936, 1995.

[20] 「ワークフローで仕事を変える」、日経コンピュータ、1996年3月4日号, pp. 129-142, 1996.

[21] 横井康二、荒尾辰之: "ワークフローシステムによる新契約査定業務の改革"、日立評論, Vol. 77, No. 6, pp. 43-46, 1995.

[22] 國島丈生、横田一正: "Workflow Base: データベース技術に基づくワーフローモデル"、情報処理学会論文誌, Vol. 39, No. 11, pp. 3122-3130, 1986.

[23] 垂水浩幸、喜田弘司、柳生弘之、石黒義英: "エージェントによるワークフローの動的再計画"、情報処理学会論文誌, Vol. 39, No. 7, pp. 2361-2369, 1998.

[24] C. Ellis, et. al.: "Dynamic change within workflow systems", Proc. Conf. on Organizational Computing Systems (COOCS'95), pp. 10-21, ACM, 1995.

[25] M. Reichert: "ADEPT flex-supportingdynamic change of workflow without losing control in workflow management systems", Journal of Intelligent Information Systems, vol. 10, pp. 93-129, 1998.

[26] A. Kumar and J. L. Zhao: "Dynamic routing and operational controls in workflow management systems", Management Science, vol. 45, no. 2, pp. 253-272, 1999.

[27] 陳暁栄、時永祥三: "遺伝的プログラミングを用いたフロー型ジョブの多目的最適化の一手法"、電子情報通信学会技術研究報告、NLP2000-140, PP. 1-8, 2001.

[28] J. R. Quinlan: "Induction of Decisiontree", Machine Learning, 1,

pp. 81-106, 1986.

[29] Y. H. Pao: ***Adaptive Pattern Recognition and Neural Networks,*** Addison-Wesley Publishing Co., Inc., 1989.

[30] F. Allen and R. Karjalainen: "Using genetic programming to find technical trading rules", Technical Report, Wharton School, University of Pennsylvania, 1993.

[31] M. Oussaidene, B. Chopard, O. V. Pictet and M. Tomassini: "Parallel genetic programming: An application to trading model evolution", Proc. of First Annual Conference of Genetic Programming, pp. 357-362, 1996.

[32] M. Oussaidene, B. Chopard, O. V. Pictet and M. Tomassini: "Parallel genetic programming and its application to trading model induction", Parallel Computing, vol. 23, pp. 1183-1198, 1997.

[33] S. Tokinaga and S. Matsuno: "Generating the production rules by using the genetic programming and its application to risk analysis", Proc. ofITC-CSCC'01, vol. 2, pp. 961-964, 2001.

[34] Y. Kishikawa and S. Tokinaga: "Prediction of stock trends by using the wavelet transform and the multi-stage fuzzy inference system optimized by the GA", Trans. IEICE, vol. E83-A, no. 2, pp. 357-366, 2000.

[35] Stock prices in CD-ROM, Toyokeizai Data Bank, 2000.

[36] Japan Credit Rating Agency: Annual Report, 2000.

[37] K. Tan and S. Tokinaga: "Optimization of fuzzy inference rules by using the genetic algorithm and its application to the bond rating", J. of Operations Research Society of Japan, vol. 42, no. 3, pp. 302-315, 1999.

[38] K. Tan and S. Tokinaga: "the design of multi-stage multi-stage fuzzy inference system with smaller number of rules based upon the optimization of rules by using the GA", Trans. IEICE, vol. E82-A, no. 9, pp. 1865-1873, 1999.

[39] Y. Kishikawa and S. Tokinaga: "Approximation of multi-dimensional chaotic dynamics by using the multi-stage fuzzy inference systems and the GA", Trans. IEICE, vol. E84-A, no. 9, pp2128-2137, 2001.

[40] L. O. Chua and L. Young: "Cellular neural network: Theory and practice", IEEE Trans., Cirsuits Syste., vol. 35, no. 10, pp. 1257-1290, 1988.

[41] L. O. Chua and T. Roska: "The CNN paradigm", IEEE Trans., Cirsuits Syste., vol. 40, no. 3, pp. 147-156, 1993.

[42] L. O. Chuaet. al: "Autonomous cellular neural networks: A unified paradigm for pattern formation and active wave propagation" IEEE Trans. Circuits Syst., vol. 42, no. 10, pp. 559-578, 1995.

[43] L. Pivka: "Autowave and spatio-temporal chaos in CNNs-Part I :A tutorial", IEEE Tanns. Circuits Syst., vol. 42, no. 10, pp. 638-649, 1995.

[44] L. Pivka: "Autowave and spatio-temporal chaos in CNNs-Part II:A tutorial", IEEE Tanns. Circuits Syst., vol. 42, no. 10, pp. 650-664, 1995.

[45] G. Hu , L. Pivka and A. L. Zheleznyak: "Synchronization of a one-dimensional array of Chua's circuits by feedback control and noise", IEEE Trans. Cirsuits Syst., vol. 42, no. 10, pp. 736-741, 1995.

[45] 時永祥三、矢加部正幸: "CNNによるネットワーク決済リスク拡散のモデル化とその応用", 電子情報通信学会技術研究報告、NLP-200-141, pp. 9-16, 2001.

[47] R. M. Goodwin: Nonlinear Economic Dynamics, Oxford University Press, 1990.

[48] V. Bohm: "Recurrence in Keynesian macroeconomic models" in G. F. Geronazzo, L. Galeotti (eds) , Nonlinear Dynamics in Economics and the Social Sciences, Springer-Verlag, Berlin, pp. 69-94, 1993

[49] C. M. Rump and S. Stidham, Jr: "Stability and chaos in input pricing for a service facility with adaptive customer response to congestion", Management Science, vol. 44, no. 2., p. 246-261, 1998.

[50] S. Stidham, Jr: "Pricing and capacity decisions for a service facility:Stability and multiple local optima", Management Science, vol. 38, no. 8, p. 1121-1139, 1992.

[51] S. Tokinaga and X. Chen: "Estimation of chaotic dynamics for the input pricing at the service facilities based on the GP and its application to the control", Proc. of 16th Digital Signal Processing Symposium, pp. 747-752, 2001.