修　士　論　文

生体分子ネットワークに適用可能なペトリネットシステムの研究

指導教官　　小長谷明彦　教授

北陸先端科学技術大学院大学
知識科学研究科知識システム基礎学専攻

050049　鈴木　龍司

審査委員：　小長谷　明彦　教授（主査）
佐藤　賢二　助教授
本多　卓也　教授
中森　義輝　教授

2002 年 2 月

目　次

図　目　次

表　目　次

第　1　章
序 論

1.1　　背景と目的

2000 年 6 月、ヒトゲノムのドラフトシーケンスの終了が宣言された。しかしながら、これは単にゲノムの全塩基配列を決定しただけであり、個々の遺伝子がどこにあるか、さらにそのはたらきが何であるかが、直ちに明らかになるわけではない。また、遺伝子の指示によって作成されるタンパク質についても、その構造・機能が明らかになっていないものは多い。ゲノムの情報を真の意味で解読するためには、遺伝子やタンパク質といった個々の部品の集まりから細胞あるいは生物個体といったシステム全体を再構築できるかどうかを調べ、「生命のはたらきをシステムのはたらきとして理解する」合成論のアプローチが必要である。このような合成論的なアプローチは基本的に情報科学の問題として扱われている。つまり、コンピュータを用いてゲノムの情報から生命の情報システムを再構築しようとしている。

このことから、最近では個々の部品とその相互作用を記述するグラフに関する研究が行われている。また、それを共通の方法論とし、分子のネットワークだけではなく、細胞のネットワークとしての脳・神経系、さらには個体のネットワークとしての生態系といった異なるレベルのデータに適用することにより、異なるレベルの生命現象を理解していくことが可能になると期待されている。

本研究では、これらのネットワークのモデル化手法として、ペトリネット(Petri Net)を用いる事とし、このようなモデルに適用可能なペトリネットシステムの開発を目的とする。

1.2　シミュレーションについて

シミュレーションとは、現実に存在する現象を、コンピュータ上で模倣することである。実際の生物実験では、コストや時間がかかりすぎたり、倫理的な問題が発生する可能性がある。シミュレーションではその心配はなく、シミュレーションで得られた知見はグラフ等、容易に可視化出来る等があげられる。また、一度シミュレーションモデルを構築すると、各パラメータや反応経路に変更を施すことにより簡単に仮想実験が行えるところは生物実験にはない利点である。

生体分子ネットワークのシミュレーションは今まで連立微分方程式等を用いて反応速度や、その他、物理的・化学的な数値計算をいくつも並べて行っていた。しかしこの方法では、複雑な系になったとき、現象とそれに関連する事物の関係がわかりにくくなる。また、反応物質の空間的な偏在等を扱えない[24]ため必ずしもよいモデルを作れないなどの問題がある。

1.3　何故ペトリネットを使うか

ペトリネットは、1962 年 C. A. Petri の学位論文[23]によって提唱された。ペトリネットは、生体システムの動作の特徴である並行性、非同期性、非決定性を明示的に表現できる数学モデルである。また、ペトリネットはネットワーク構造とその挙動を視覚的に捉えることができ[1],[2]、これらの特徴から遺伝子ネットワークや代謝経路のモデル化ツールとして優れている[16]。

また、離散値だけでなく連続値を扱うことのできる、ハイブリッドペトリネット[4],[7],[8]を用いることによって、酵素反応など数値計算を必要とする部分を持つような系の記述も可能となった。

本研究ではペトリネットシステムへの位置情報の導入を行うことにより、胚発生など位置情報を必要とするモデルの記述を目指す。こういった特徴を併せ持つペトリネットシステムを開発することにより、生体分子ネットワークを統一的に扱えるシミュレーションツールを開発する。

1.4　本論文の構成

本論文は次のように構成される。第2章では、ペトリネットの概略について述べる。第3章では、最も基本的なペトリネットである、正規化ペトリネットを作成したことについて述べる。第4章では、シミュレーションツールとしての有効性を増すために、システムをハイブリッド化したことについて述べる。第5章では、作成したペトリネットシステムを用いて行った、細胞周期のシミュレーションについて述べる。第6章では、位置情報を扱えるペトリネットシステムを開発し、それを使ったショウジョウバエの初期胚におけるパターン形成のシミュレーションについて述べる。最後に第7章で本論文のまとめを行い、また今後の課題について述べる。

第　2　章
ペトリネットの概略

2.1　　はじめに

ここではこのペトリネットについて入門書[1],[2]をもとに概略的に説明する。

ペトリネット（Petri Net、PN）は多くのシステムに適用可能なグラフィックで数学的なモデル化ツールである。グラフィックなツールとしては、フローチャートやブロックダイヤグラフ・ネットワークと同じようにシステムの静的な構造の可視化手段としてペトリネットを使用することができるだけでなく、ペトリネットの中でトークン(token)を使用することにより、システムの動的な事象をシミュレートすることができる。一方、数学的なツールとしては、システムの挙動を表現する状態方程式や代数方程式その他の数学モデルを立てることが可能である。なお、次節以降で説明するペトリネットは基本的に離散時間で処理が進行するものである。連続時間で処理を行うものについては 4 章で説明する。

2.2　　ペトリネット構造

ペトリネットは構造的には、プレース(place)とトランジション(transition)の 2 種類のノードを持つ 2 部有向グラフである。システムの静的な接続状態を示すペトリネット構造に、動的な性質を示すトークンを導入したのがペトリネットである。プレースとトランジションは基本的にそれぞれ、システムの状態または条件（condition）と、システムの状態遷移を表す事象（event）に対応する。ペトリネットのプレースとトランジションのシステム的解釈[2]を表 2.1 にまとめて示す。

表 2.1　プレースとトランジションのシステム的解釈

入力プレース	トランジション	出力プレース
現在の状態	状態遷移	次の状態
入力データ	処理	出力データ
入力バッファ	プロセッサ	出力バッファ
資源の補足	タスク・ジョブ	資源の解放
前提条件	事象（イベント）	後提条件
条件論理式	論理節	結論論理式

2.2.1　2 部有向グラフと PN 構造

有向グラフ G(V,A)は、いくつかのノードの集合 V と、ノードの順序対、すなわち 2 つのエンドノードで定義されるアーク(arc)の集合 A から構成される。

有向グラフのノード集合 V を V_1、V_2に分割して、どのアークのエンドノードも一方が V_1に、他方が V_2に属するとき 2 部有向グラフと呼ばれる。ペトリネット構造 N は、アンマークト(unmarked)ペトリネットとも呼ばれ、V_1をプレースの集合 P、V_2をトランジションの集合 T とする 2 種類のノードを持つ 2 部有向グラフである。すなわち、アークはプレースからトランジションに向かうアーク{P×T}か、トランジションからプレースへ向かうアーク{T×P}かのどちらかであり、プレースからプレースへや、トランジションからトランジションへ向かうアークは存在しない。　ペトリネット構造（ペトリネットグラフ）のグラフィック表現では、プレースを円または楕円で、トランジションをバーまたは長方形で示している。

ペトリネット構造 N(P,T,A)は以下に示す状態変数を表すプレースの有限集合 P、状態遷移を表すトランジションの有限集合 T、情報や制御の流れを示すアークの有限集合 A から構成される。

$$P = \{p_1, p_2, \cdots, p_n\} = \{p_i \mid 1 \leqq i \leqq |P| = n\}$$

$$T = \{t_1, t_2, \cdots, t_m\} = \{t_j \mid 1 \leqq j \leqq |T| = m\}$$

$$A \subseteq \{P \times T\} \cup \{T \times P\} \qquad (P \cap T = 0,\ P \cup T \neq 0)$$

2.2.2　接続行列と入出力集合

プレース p からトランジション t へのアークがあれば 1、なければ 0 の値を持つ入力接続関数 $W^{-}(p,t)$と、トランジション t からプレース p へのアークがあれば 1、なければ 0 の値を持つ出力接続関数 $W^{+}(p,t)$を導入する。

$$W^{-} \in \{0,1\}^{n \times m} \quad （または\ W^{-} : P \times T \rightarrow \{0,1\}）$$

$$W^{+} \in \{0,1\}^{n \times m} \quad （または\ W^{+} : P \times T \rightarrow \{0,1\}）$$

はそれぞれ入力アーク、出力アークを表す２進ｎ×m 行列であり、これらを使ってペトリネット構造は４項組

$$N(P,T, W^{-}, W^{+})$$

と表現することができる。

トランジションへの入力接続を示す n×m 行列 W^{-}と、トランジションからの出力接続を示す n×m 行列 W^{+}から、接続行列 W を

$$W = W^{+} - W^{-}$$

と定義され、

$$W \in \{-1,0,1\}^{n \times m}$$

となる。

トランジション t への入力プレースの集合、およびｔからの出力プレースの集合はそれぞれ、･t、t･と表現され、

$$I(t) = {\cdot}t = \{p \mid W^{-}(p,t) > 0\}$$

$$O(t) = t{\cdot} = \{p \mid W^{+}(p,t) > 0\}$$

と定義できる。同様にして、プレース p への入力トランジションの出力およびｐからの出力トランジションの集合はそれぞれ、･p、p･と表現され、

$$I(p) = {\cdot}p = \{t \mid W^{+}(p,t) > 0\}$$

$$O(p) = p{\cdot} = \{t \mid W^{-}(p,t) > 0\}$$

と表すことができる。

入出力集合を用いて接続行列を表現すれば、

$$W(p,t) = \begin{cases} -1\ ;\ p \in {}^{\bullet}t、かつ、p \notin t^{\bullet}のとき \\ 1\ ;\ p \in t^{\bullet}、かつ、p \notin {}^{\bullet}tのとき \\ 0\ ;\ その他のとき \end{cases}$$

となる。

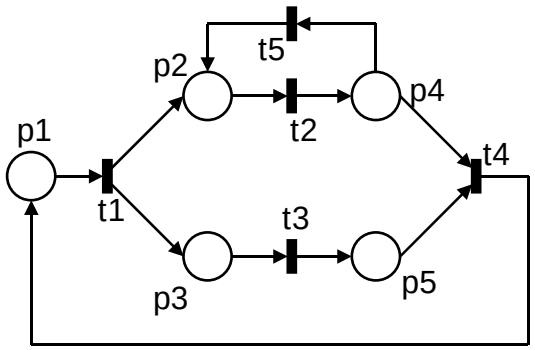


図2.1　ペトリネット構造の例

図 2.1 に示す例で説明しよう。

$$P = \{p1,p2,p3,p4,p5\}$$

$$T = \{t1,t2,t3,t4,t5\}$$

$$W^{-} = \begin{array}{c} \begin{matrix} t1 & t2 & t3 & t4 & t5 \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array} \begin{matrix} p1 \\ p2 \\ p3 \\ p4 \\ p5 \end{matrix}$$

$$W^{+} = \begin{array}{c} \begin{matrix} t1 & t2 & t3 & t4 & t5 \end{matrix} \\ \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array} \begin{matrix} p1 \\ p2 \\ p3 \\ p4 \\ p5 \end{matrix}$$

$$W = \begin{array}{c} \begin{matrix} t1 & t2 & t3 & t4 & t5 \end{matrix} \\ \begin{pmatrix} -1 & 0 & 0 & 1 & 0 \\ 1 & -1 & 0 & 0 & 1 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & -1 \\ 0 & 0 & 1 & -1 & 0 \end{pmatrix} \end{array} \begin{matrix} p1 \\ p2 \\ p3 \\ p4 \\ p5 \end{matrix}$$

$${}^{\bullet}t1=\{p1\},\ {}^{\bullet}t2=\{p2\},\ {}^{\bullet}t3=\{p3\},\ {}^{\bullet}t4=\{p4,p5\},\ {}^{\bullet}t5=\{p4\}$$

$$t1^{\bullet}=\{p2,p3\},\ t2^{\bullet}=\{p4\},\ t3^{\bullet}=\{p5\},\ t4^{\bullet}=\{p1\},\ t5^{\bullet}=\{p2\}$$

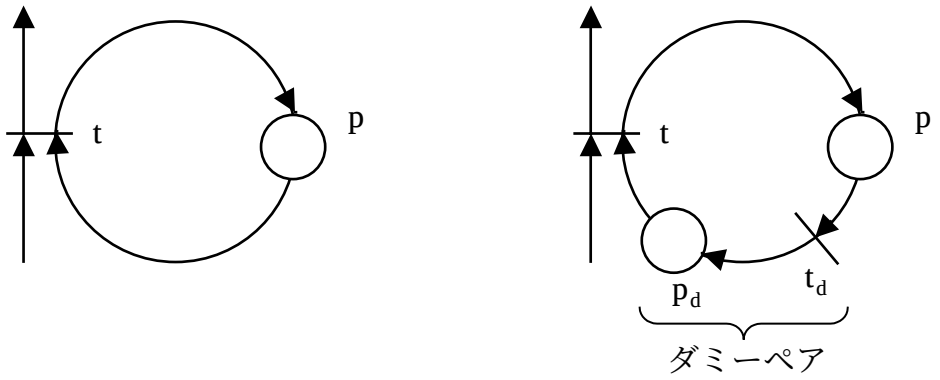
$${}^{\bullet}p1=\{t4\},\ {}^{\bullet}p2=\{t1,t5\},\ {}^{\bullet}p3=\{t1\},\ {}^{\bullet}p4=\{t2\},\ {}^{\bullet}p5=\{t3\}$$

p1・={t1}, p2・={t2}, p3・={t3}, p4・={t4,t5}, p5・={t4}

である。

トランジション t1 のように、複数の出力アークをもつ場合(|t・|> 1)をフォーク(fork)、t4 のように複数の入力アークをもつ場合(|・t|> 1)をジョイン(join)と呼ぶことがある。トランジションは、入力アークのない場合(・t=0)ソース(source)トランジション、出力アークのない場合(t・=0)シンク(sink)トランジションと呼ぶ。同様に、入力トランジションを持たないプレース(・p=0)をソースプレース、出力トランジションを持たないプレース(p・=0)をシンクプレースと呼ぶ。

図 1.2(a)のように、プレース p がトランジション t の入力プレースでかつ出力プレースの場合（トランジション t がプレース p の入力トランジションでかつ出力トランジションの場合）p と t のペアを自己ループ(self-loop)と呼ぶ。そしてペトリネット構造は自己ループを持たない場合を純(pure)、もつ場合を非純(impure)と呼ぶ。



(a) (b)

図2.2　自己ループとダミーペアによるループ化

図 1.2(b)の例に示すように、トランスとプレースのダミーペア(dummy pair)の導入などにより、いっぱんにどんなペトリネット構造も純にできる。

なお、p と t の間に自己ループがある場合には、$W^-(p,t)= 1$、$W^+(p,t)= 1$ となり接続行列の対応する要素 $W(p,t)= 0$ となるから、接続がないのと同じになってしまう。すなわち、非純ペトリネット構造の場合には接続行列から元のペトリネット構造を回復することはできず、情報損失があることに注意する必要がある。

2.3　正規ペトリネット

まず最も基本的なペトリネットである正規ペトリネット（ordinary PN）について説明する。次節以降に示される拡張型のペトリネットもこの正規ペトリネットを基本として導かれる。

2.3.1　トークンとマーキング

前節で示したシステムの静的性質を示すペトリネット構造にシステムの動的性質を示すトークンまたはマーク(mark)を導入する。トークン(マーク)はシステムの動作状態を示す目印であり、システムの実行中に次項に示す規則に従ってネット中を移動する。移動により各プレースのトークン数が変化することにより、システムの状態が変化する。

トークンはプレース中の黒丸あるいは正の整数で表され、各プレースにトークンを割り当てることをマーキング M という。マーキングは各プレース中のトークンの分布、すなわちシステムの状態を示し、各プレースに何個トークンがあるかを示す長さ n のベクトル

$$M = (M(p_1),M(p_2),\cdots, M(p_i) ,\cdots, M(p_n)) = (m_1,m_2,\cdots,m_i,\cdots,m_n)$$

で表す。すなわちマーキング M の i 番目の要素 m_i は、プレース p_1 中のトークンの数 $M(p_1)$を表す。M はプレースの集合 N＝{0,1,2,⋯}にマッピング(mapping)するマッピング関数とも定義できる。

$$M：P \rightarrow N\ (M \in N^n)$$

マーキングの初期割り当てを初期マーキング(initial marking)M_0 と呼ぶ。図 2.1 のペトリネット構造のプレース p1 に 1 個のトークンを割り当てた図 2.3(a)の初期マーキングは M_0=(1,0,0,0,0)である。

マークされたマークト(marked)ペトリネット構造をペトリネット･システムあるいは単にペトリネット(PN)と呼び、ペトリネット構造 N にトークンの分散初期状態を示す初期マーキング M_0($M_0 \in N^n$)を加えて,

$$PN = (N, M_0)$$

と表される。

2.3.2　発火規則

システムの動的性質は、ペトリネットの状態を示すマーキングを各トランジションの発火によって変化させることにより表すことができる。

ある時点においてトランジションは、そのすべての入力プレースが少なくとも 1 つのトークンを持つとき発火可能になる。

$$\forall p \in \cdot t、\ M(p) > 0（または \forall p \in P、M(p) \geqq W^{-}(p,t)）$$

発火可能なトランジションが実際に発火するかどうかは、そのトランジションに対応する事象が実際に起こるかどうかによるが、発火そのものは瞬間的に起こると考える。なお、入力を持たないソーストランジションは常に発火可能である。

マーキング M において発火可能なトランジションが発火すると、そのすべての入力プレースからトークンを１個ずつ取り去って（前提条件の消費）、その出力プレースのすべてに１個ずつトークンを送り出す（後提条件の生成）その結果新しいマーキング M’が得られる。

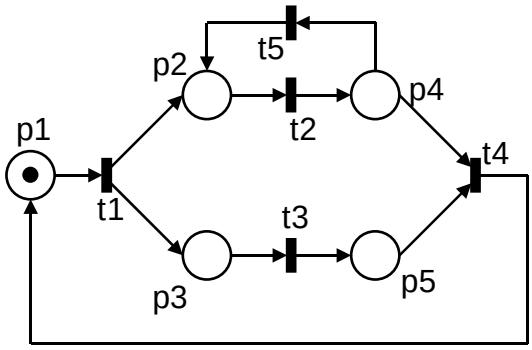
$$M'(p) = M(p) - W^{-}(p,t) + W^{+}(p,t) = M(p) + W(p,t)$$

図 2.3(a)に示す初期マーキング M0(1,0,0,0,0)では、トランジション t1 だけが発火可能であり、t1 が発火すると同図(b)に示すように、p1 のトークンは取り除かれ、p2 と p3 に１個ずつトークンが送り出され、新しいマーキング M1(0,1,1,0,0)となる。

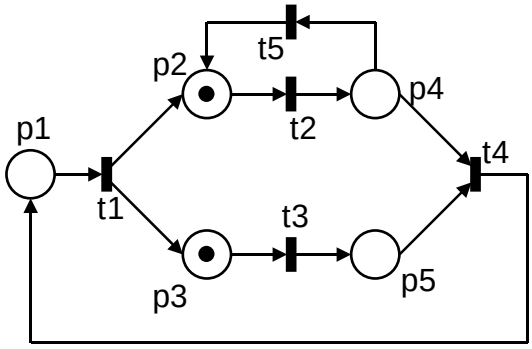
2.3.3　並行・競合などの基本動作

図 2.3(b)では、t2 と t3 がともに発火可能状態になる。この 2 つのトランジションは入力プレースを共有しないので独立に発火できる。すなわち、p2→t2→p4 の経路と p3→t3→p5 の経路は並行動作が可能である。その結果、マーキングは図 2.3(c)に示す M_2(0,0,0,1,1)となる。

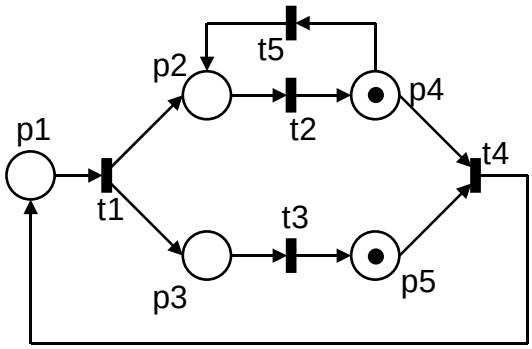
図 2.3(c)では、t4 と t5 は入力プレースとして p4 を共有しているから、この場合には t4 と t5 は同時に発火することはできない。すなわちトークンは分割不能であり、ただひとつのトランジションによってしか取り除かれない。したがって、先に発火したトランジションが p4 のトークンを取り除き、その結果、もう一方のトークンは発火不能になってしまう。



(a) 初期マーキング
M_0 =(1,0,0,0,0)



(b) t1発火後のマーキング
M_1 =(0,1,1,0,0)
（t2,t3が独立に発火可能な並行動作可能状態を示す）



(c) t2,t3発火後のマーキング
M_2 =(0,0,0,1,1)
（t4とt5がp4のトークンを奪い合う競合状態を示す）

図2.3 ペトリネットの動作例（並行と競合）

この状態は、衝突または競合を表している。このような場合、どちらのトランジションが発火するかはまったく任意であり、非決定性を表している。

このように 2 つ以上のトランジションが共通の入力プレースを持つ場合、構造的競合という。構造的競合があっても、マーキングによって実際に競合が起こる図 2.3(c)

実効競合と、実際には競合が起こらない非実効競合がある。たとえば、図 2.3(c)において M(p4)=2 の場合には、構造的競合があっても実際には競合は起こらず、非実効競合である。

2.4　拡張型ペトリネット

ここでは、正規ペトリネットの機能拡張について説明する。

2.4.1　多重アーク

あるプレースが複数個のトークンを持った場合、トランジションの発火によって、それらのトークンをいくつかまとまった数だけ移動させることが必要なときがある。これは図 2.4(a)に示すように多重アーク(multiple arc)を導入することで解決できる。また、多重アークは図 2.4(b)に示すように、単一アークに多重度(multiplicity)をあらわす重みを加えて表現する。重みが１の場合には記入しない。

多重アークをまったくもたない正規ペトリネットに対して、このようにアークに重みを導入したペトリネットをプレーストランジションネット（P/T ネット）、一般化ペトリネット(generalized PN)という。正規ペトリネットと一般化ペトリネットは、モデルの効率や利便性に違いはあるが、本質的に同じモデル化能力をもつ。

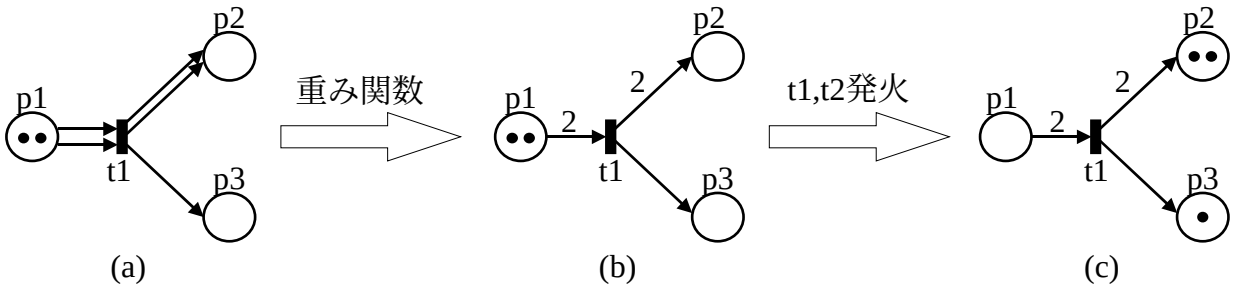


図2.4　多重アークと重み関数による表示

2.4.2　有限容量ネット

実際の物理的なシステムのモデル化では、各プレースが保持できるトークンの数に上限があるのが普通である。そのため、トランジションが発火したくても、出力プレースの容量制限のために発火不能となる場合も生まれてくる。このプレースの容量制限を容量関数（capacity function）K で表し、各プレースの容量をK(p)で表す。

有限容量ネットでは、前項で述べたトランジション発火可能条件に。「t の各出力プレースの、発火後のトークン数が K(p)を超えない」という条件を加える必要がある。したがって有限容量ペトリネットの発火可能条件は、

$$\forall p \in {}^{\bullet}t,\ M(p) \geqq W^{-}(p,t)$$

$$\forall p \in t^{\bullet},\ M(p)+W^{+}(p,t) \leqq K(p)$$

となる。

次の 2 ステップからなる補プレース変換（complementary-place transformation）によって、すべての有限容量ネット(N, M_0)は、等価な（すべての発火系列の集合が同じ）無限容量ネット(N', M_0')に変換できる。したがって今後特に断らない限り、無限容量ネットを考えることにする。

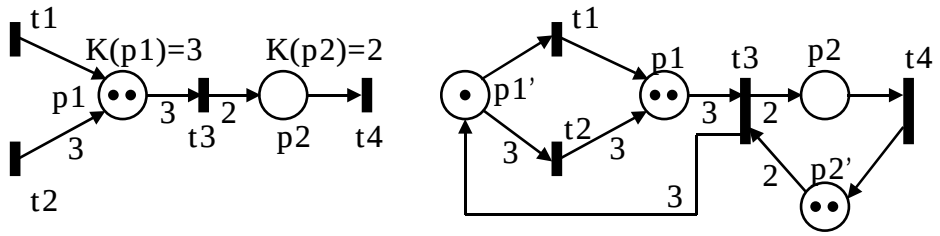
ステップ 1：各プレースに補プレース p'を加え、p'の初期マーキングを

$$M_0'(p') = K(p) - M_0(p)$$

とする。

ステップ 2：各トランジション t とある補プレース p'の間に、プレース p 中のトークンとその補プレース p'中のトークンの和がトランジション t の発火の前後で p の容量 K(p) に等しくなるように新しいアークを引く。すなわち、アーク(t, p)に対して同じ重みのアーク(p', t)を引き、アーク(p, t)に対して同じ重みのアーク(t, p')を引けばよい。

図 2.5 に有限容量ネットを補プレース変換した例を示す。

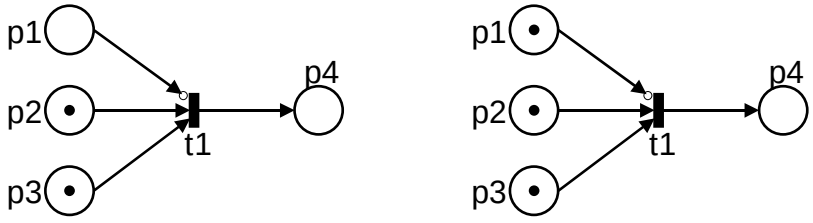


(a)　有限容量ネット　　(b)　無限容量ネット

図2.5　補プレース変換の例

2.4.3　抑止アーク

トランジションが発火可能となるには、そのすべての入力プレースにトークンがあることであった。しかし抑止アーク(inhibitor arc)により、入力プレースにトークンがあれば発火を抑制し（図 2.5(b)）、トークンがなければ発火可能とする（図 2.5(a)）ことができる。



(a)　発火可能　　(b)　発火不能

図2.6　抑止アーク

抑止アークは図 2.5 で示すようにトランジション側に小さい丸印をつけた矢印あるいは線で表す。また、トランジションが発火しても抑止アークを通してトークンは移動しない。抑止アークの導入により、「プレースにトークンがない」というゼロテストが可能となる。一般化ペトリネットの場合は、抑止アークの重みよりプレース中のトークンの数が小さければ発火可能になる。したがって、「ある定数より小」の検出が可能になる。

2.4.4　優先順位ペトリネット

優先順位ペトリネットは優先順位(priority)をトランジション付与したペトリネットで、複数のトランジションが発火可能な競合状態に陥った際、優先順位の高いトランジションから順に発火するものである。

優先順位ペトリネットを使ってプレース p1 内にトークンがあるかをテストすることができる。これを図 2.6 に示す。プレース p2 にトークンを一個投入し、トランジション t1 の優先順位をトランジション p2 より高くしておくことで右側にあるプレース p3、p4 のいずれかに、p1 のマーキングに依存してトークンが現れる。これは、トランジション t1 は p1 にトークンがあるときにだけ発火できるからである。一方、t2 は p1 が空で発火できないときに限って発火できる。したがって、トークンが p3、p4 のどちらに移動したかによってプレース p1 のトークンの有無がわかる。

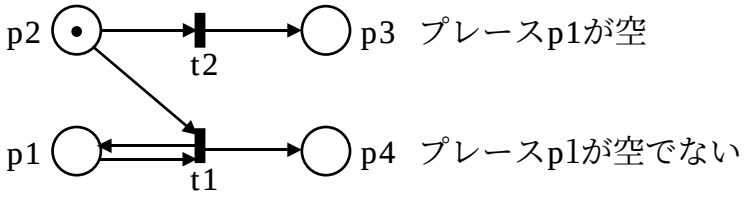


図2.7　優先順位ペトリネットによるゼロテスト

前項であげた抑止アークを導入したペトリネットと優先順位ペトリネットは、ゼロテストが可能なことからチューリングマシンの計算能力をシミュレートすることが可能である[3]。したがって、ゼロテスト可能なペトリネットはいかなるシステムといえどもモデル化できるようなモデル化の方法となる。

2.4.5　連続・ハイブリッドペトリネット

連続ペトリネット(continuous PN)はトークンおよびアークの重みに実数値を取ることができる。連続ペトリネットは、化学物質の濃度などの連続量を表現する際に有効である。一方、遺伝子の転写活性などは離散的なトークンを利用したほうが記述しやすい。ハイブリッドペトリネットは、連続値と離散値の両方を扱うことができ、生体分子ネットワークの記述に広く受け入れられている。

ハイブリッドペトリネットについては第 4 章で詳しく説明する。

第　3　章
正規ペトリネットの実装

3.1　　はじめに

第 3 章では、すべてのペトリネットの基礎となる正規ペトリネットのシステム開発に関して、その設計方針、プログラム実装について説明し、作成したシステムに例題を計算させ、その結果をもとにシステムを検証する。

3.5 項以降では、作成したシステムに対して 2 章で示した拡張のうち、「多重アーク」、「有限容量ネット」、「抑止アーク」、「優先順位ペトリネット」の拡張を行いそれについて検証を行う。

3.2　　設計方針

本システムは、開発言語として C++を使用する。これは、ペトリネットシステムがプレース、トランジション、アーク、トークンという部品から構成されており、これらの部品をそれぞれ「クラス」とみなすことは妥当だと考えられるからである。ただし、本システムではトークンについてはプレースの値としてのみ使用されるので特にクラスは設けなかった。

図 3.1 に本システムのクラスの関係を示す。

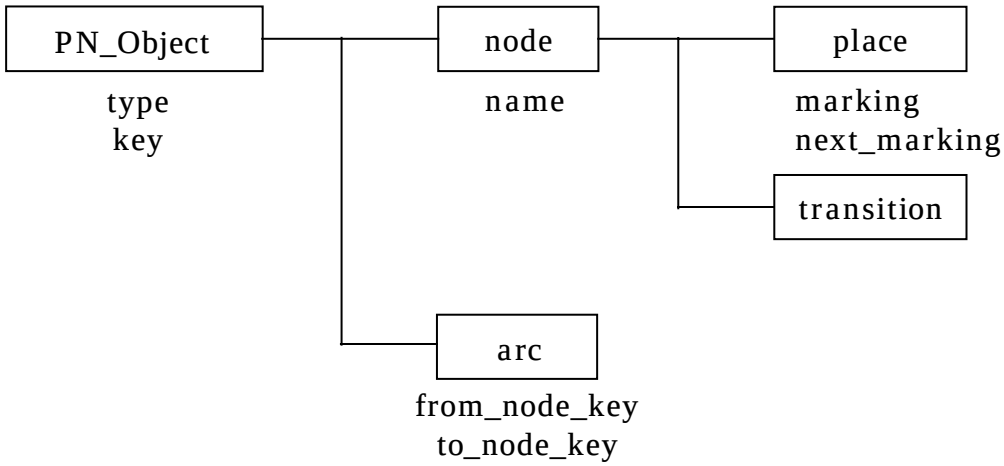


図3.1　クラス相関図Ⅰ

四角に囲まれたのがクラス名、その下にかかれているのがそのクラスのメンバー変数である。クラスは左から右に継承されている。つまり「PN_Object」クラスを基底クラスとし、「node」クラス、「arc」クラスはその派生クラスとなる。そして「place」クラス、「transition」クラスは「node」クラスの派生クラスとなる。

本システムにおいて実行の際作成されるオブジェクトのクラスは「place」、「transition」、「arc」の 3 種類であり、「PN_Object」クラスや「node」クラスは基底クラスとしてのみ使用される。

次に、メンバー変数について説明する。メンバー変数「type」はそれぞれのオブジェクトのタイプを示し、オブジェクトのクラス名を格納する。「key」は 1 つ 1 つのオブジェクトを識別するための ID 番号である。「name」はプレースやトランジションの名前である。「marking」はプレース内のトークン(マーク)の数を示し、「next_marking」は計算の際、ステップごとに変化するトークンを一時的に格納する。「from_node_key」、「to_node_key」はアークの接続元オブジェクトと接続先オブジェクトの「key」を格納する。

基底クラスのメンバー変数をその派生クラスも持っているので、たとえば「place」クラスは、「type」、「key」、「name」、「marking」、「next_marking」をそのメンバー変数として持つことになる。

3.3　実装

ここでは、正規ペトリネットシステムの実装について説明する。

まず、正規ペトリネットの計算アルゴリズムについて簡単にまとめると以下のようになる。

1. ペトリネット構造を記述し、それに初期マーキングを加える。
2. 任意のトランジションについて、発火可能か調べる。
3. すべてのトランジションについて 2 を行う。

4a. トランジションが発火可能ならば、そのすべての入力プレースからトークンを１個ずつ減らし、発火。

4b. 発火後、その出力プレースのすべてに１個ずつトークンを送り出す。

5. 4a、4b をすべての発火可能トランジションについて行う。
6. 新しいマーキングが得られる。
7. 任意の回数だけ 2 に戻る。

本システムでは上記のアルゴリズムを以下のように実装する。。

i. 各オブジェクトの作成およびパラメータの設定。（1）

ii. 任意のトランジションについて、発火可能か調べる。（2）

iii. トランジションが発火可能ならば、そのすべての入力プレースから「marking」を１個ずつ減らし、発火。(4a)

iv. 発火後、そのすべての出力プレースの「next_marking」を１個ずつ増やす。(4b)

v. すべてのトランジションについて ii、iii、iv を行う。（3)、（5）

vi. すべてのプレースについて「marking」と「next_marking」を足し合わせ新しいマーキングを得、それをファイルに出力。（6）

vii. 任意の回数（必要なステップ数）だけ ii に戻る。（7）

この中で出てくる「marking」と「next_marking」はクラス「place」のメンバー変数

のことである。また、行末にある括弧の数字は、正規ペトリネットのアルゴリズムのどの段階にあたるのかを示している。

この 2 つのアルゴリズムを見比べると、トークンの移動が行われる段階は違うが大筋で同じ事が行われていることがわかる。ただし、ペトリネット構造に競合があった場合、発火するトランジションは任意に決められるはずだが、本システムでは先に処理されたトランジションが発火することになる。

上記のそれぞれの処理が本システムのどこで行われるかを図 3.2 に示す。

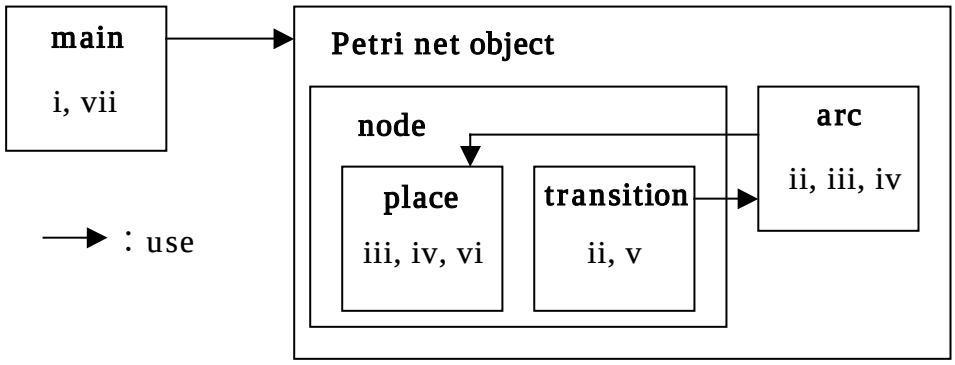


図3.2　本システムの仕様

ii、iii、iv は重複しているが、これは、発火の処理をプレース、トランジション、アークの各オブジェクトに分散化したためである。

3.4　例題

ここでは、ここまでの段階で実装したシステムを第 2 章の図 2.1 や図 2.3 で示したペトリネットについて、計算してみる。その結果は次のようになった。

	p1	*p2*	*p3*	*p4*	*p5*
0	*1*	*0*	*0*	*0*	*0*
1	*0*	*1*	*1*	*0*	*0*
2	*0*	*0*	*0*	*1*	*1*
3	*1*	*0*	*0*	*0*	*0*

第 1 列の値はステップ数を示している。これは第 2 章で得た結果である $M_0(1,0,0,0,0)$、$M_1(0,1,1,0,0)$、$M_2(0,0,0,1,1)$と 2 ステップ目までは等しい。第 3 ステップについては、このシステムでは t4 が先に処理されるため t4 が発火し t5 は発火しない。これは初期化の段階で「transition」オブジェクト t4 を先に作成しているからであり、t5 を先に作成していれば t5 が発火し t4 は発火不能になるようにすることができる。

しかし、この方法では大きなペトリネットを扱うときにはとても不便になってしまう。そこで対策として、競合のないようなペトリネットにする、あるいはシステムを優先順位ペトリネットに拡張することがあげられる。すべての問題に対して競合のないペトリネットを作ることは一般に不可能であるし問題のモデル化にも悪影響を与えかねない。そこで本システムではペトリネットを拡張していくことにした。

3.5 簡単な拡張

2 章および前節であげたように、正規ペトリネットではモデル化できない問題、あるいはモデル化するのが難しい問題がある。そこで、この節ではそのような問題に対応するために 2 章で示した拡張のうち、「多重アーク」、「有限容量ネット」、「抑止アーク」、「優先順位ペトリネット」の拡張を行う。

3.5.1 多重アーク

多重アークは、2 章で示したように複数のアークをまとめることでモデルの簡素化を図っている。本システムでは、「arc」クラスに「weight」メンバー変数を導入することで多重アークを実現する。これは、単に複数のアークをまとめるだけでなく、次章で示すような連続量を扱うペトリネットに対してもアークの重み（weight）を連続量で表現することで対応できる。

3.5.2 有限容量ネット

有限容量ネットは各プレースが保持できるトークンの数を有限にしたものである。本システムでは、「place」クラスに「capacity」メンバー変数を導入することで有限容量ネットを実現する。この「capacity」に保持できるトークン数の上限を格納し、

発火可能か調べる際（処理 ii）に「marking」＋「weight」が「capacity」を超えるときは発火しないようにする。無限容量ネットの場合には「capacity」= - 1 とした。

3.5.3　抑止アークとテストアーク

抑止アークは、入力プレースにトークンがあれば発火を抑制し、トークンがなければ発火可能とする。そして、トランジションが発火してもトークンの移動は起こらないという性質のアークである。

これはアークの「type」に「inhibitor」を追加し、発火可能か調べる際（処理 ii）にアークの「type」によりその処理を方法を変えることで対応する。また、トランジションが発火する際に「inhibitor」によって接続している入力プレースのトークンの移動（処理 iii）は行わないようにする。抑止アークでないアークの「type」を「normal」とする。

発火可能か調べる際には「normal」と同じ処理をするが発火後のトークンの移動を行いたくないときがある。これを今までのペトリネットで実現すると、自己ループを描くことになる。そこでアークの「type」に「test_arc」を加え、処理 ii は「normal」と同じ、処理 iii は「inhibitor」と同じ処理を行うようにする。これによりダミーペアなどの余計な接続を作らずにペトリネット構造を純なものにしておくことができる。テストアークは図 3.4 で示すように破線の矢印で表すことにする。

抑止アークと次項であげる優先順位ペトリネットの導入は、2 章で示したように多くの問題をモデル化可能とする。

3.5.4　優先順位ペトリネット

優先順位ペトリネットは優先順位をトランジション付与したペトリネットで、複数のトランジションが発火可能な競合状態に陥った際、優先順位の高いトランジションから順に発火するものである。

本システムでは「transition」クラスに「priority」メンバー変数を加えることにより優先順位ペトリネットを実現する。処理 i と処理 ii の間に「priority」が高い順にトランジションの発火をチェックする順番を変更し、以後の発火チェック（処理 iii）はこの順番に従うようにする。

ここで、2 章で見た優先順位ペトリネットを使った p1 にトークンが存在するかどうかのゼロテストを行ってみる。図 3.3 は、図 2.6 で示されたゼロテストの図をテストアークを使って書き直したものである。

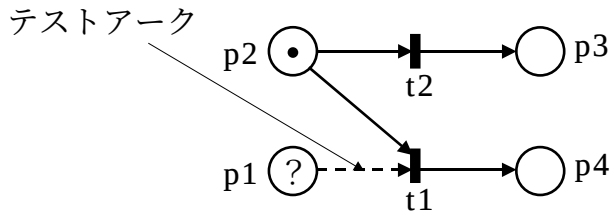


図3.3　優先順位ペトリネットによるゼロテストⅡ

p1 にトークンがあった場合の結果を以下に示す。

	p1	*p2*	*p3*	*p4*
0	*1*	*1*	*0*	*0*
1	*1*	*0*	*0*	*1*

次に p1 トークンがなかった場合の結果を示す。

	p1	*p2*	*p3*	*p4*
0	*0*	*1*	*0*	*0*
1	*0*	*0*	*1*	*0*

この結果は望むものであり、本システムが正常に動いていることがわかる。

3.6　おわりに

この章では、システムの基本となる正規ペトリネットを作成し、その後、「多重アーク」、「抑止アーク」、「優先順位ペトリネット」といった拡張を施していった。この拡張により、図 3.1 に示したクラスの関係は図 3.4 のようになった。

図 3.4 において追加されたメンバー変数「weight」、「priority」はそれぞれ前節までに説明したペトリネットの拡張のために追加されたものである。また「arc」オブジェクトの「type」として「inhibitor」、「test_arc」を追加し、正規ペトリネットで用いられていたアークを「normal」とした。

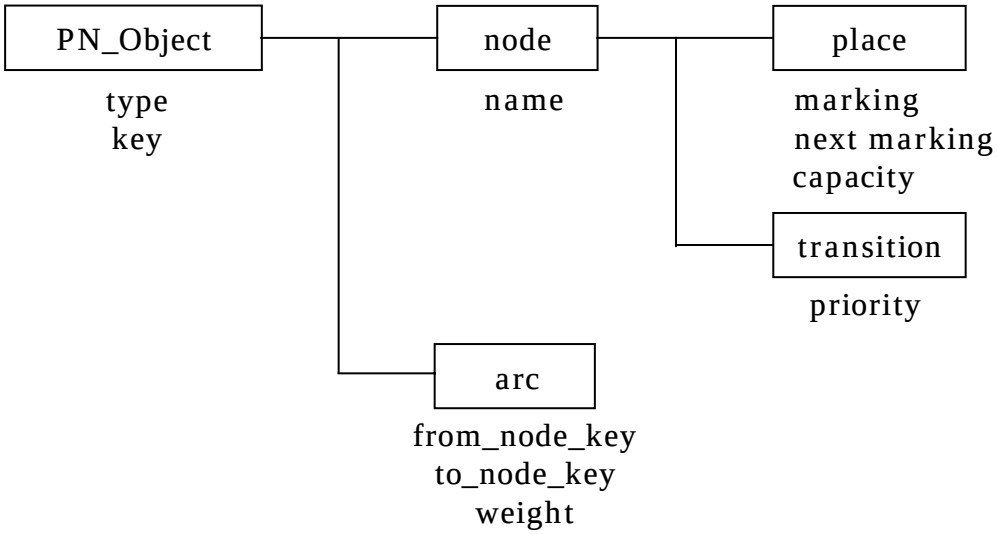


図3.4　クラス相関図Ⅱ

これに伴い、3.3 節で示したアルゴリズムを以下のように変更した。

i.　各オブジェクトの作成およびパラメータの設定。
　i'.　トランジションを「priority」の高い順にソーティング。
ii.　ソーティングされた順にトランジションを、発火可能か調べる。
　ii'.　アークの「type」が「normal」、「test_arc」なら従来通りの発火規則、「inhibitor」なら逆となる。
　ii".　「capacity」が－1 なら発火可能。「capacity」が－1 以外で「marking」＋「weight」が「capacity」を超えたら発火不能。
iii'.　トランジションが発火可能ならば、「normal」アークで接続されているすべての入力プレースから「marking」を「weight」分だけ減らし、発火。
iv'.　発火後、「normal」アークで接続されているすべての出力プレースの

「next_marking」を「weight」分だけ増やす。

v. すべてのトランジションについて ii、iii、iv を行う。

vi. すべてのプレースについて「marking」と「next_marking」を足し合わせ新しいマーキングを得、それをファイルに出力。

vii. 任意の回数（必要なステップ数）だけ ii に戻る。

処理 i’、ii’、ii”、iii’、iv’が新たに加わったり変更された処理である。処理 i’は優先順位ペトリネットへの拡張のため、処理 ii’は抑止アークの導入のため、処理 ii”は有限容量ネットに対応するため、処理 iii’、iv’は多重アーク、抑止アーク、テストアークの導入のために変更した。

第　4　章
連続・ハイブリッドペトリネットの実装

4.1　　はじめに

前章で作成したペトリネットシステムが、トークンおよびアークの重みとして正の整数値しか取ることを許されていなかったのを、実数値を取ることが可能な連続ペトリネット、ハイブリッドペトリネットについて文献[1],[4],[5]を元に説明し、システムの拡張を行っていく。

4.1.1　連続ペトリネット

連続ペトリネットは、R. David と H. Alla によって 1987 年に提案された[7]、トークンおよびアークの重みとして実数値をとることを許したペトリネットである。連続ペトリネットが提案された動機の一つに、トークン数がかなり大きい場合に可到達マーキング数（初期マーキングから発火を繰り返す間に取り得るマーキングの総数）の爆発により事実上解析が不可能になることが挙げられる。たとえば、生産システムでは各バッファに蓄えられるパーツの量を各プレースでのトークン数としたとき、バッファの容量がかなり大きい場合には、可到達マーキング数が爆発的に増大する。そこで、パーツの個数を整数ではなく実数で表現し、パーツの流れをフローとして近似して解析を行うほうが実用上有効となる。

4.1.2　ハイブリッドペトリネット

生産システムにおいてパーツの流れなどを連続量で近似するのことは有効であるが、機械の立ち上げ、停止などを表現するには離散的なトークンを利用したほうがよ

い。そこで、連続値を取るトークンと離散値しか取れないトークンをともに持つペトリネットが提案され[7],[8]、それをハイブリッドペトリネットという。

ハイブリッドペトリネットは、プレーストランジションネットと連続ペトリネットを組み合わせたもので、離散トークンを持つ離散プレース、連続トークンを持つ連続プレース、瞬時に発火する離散トランジション、連続発火する連続トランジションからなる。これらをノードとする 2 部有向グラフによって、ハイブリッドペトリネットは定義される。また、これらを区別するために図 4.1 で示す記号で表すことにする。

(a) (b) (c) (d)

図4.1 ハイブリッドペトリネットのノード：
(a)離散プレース、(b)連続プレース
(c)離散トランジション、(d)連続トランジション

4.2 設計方針

連続ペトリネット、ハイブリッドペトリネットには、時間付きと時間なしがあるが、本システムは時間なしの連続ペトリネットとした。時間付きペトリネットとは、発火条件を満たしたトランジションの発火を遅延させることによって時間の概念を導入したものであり、これを利用した連続ペトリネットハイブリッドペトリネットについては、文献[4]に説明を譲る。

本システムでは、時間なしの連続ペトリネット、ハイブリッドペトリネットとしているが、これは、今までの１ステップをいくつかの小ステップに分割することで、マーキングの連続的な変化を近似している。

図 4.2 に作成したペトリネットシステムのクラスの相関を示す。この図から、クラス「place」はその下にクラス「discrete_place」(離散プレース)と、クラス「continuous_place」(連続プレース)を派生し、「transition」クラスは、「discrete_transition」(離散トランジション)クラスと「continuous_transition」(連

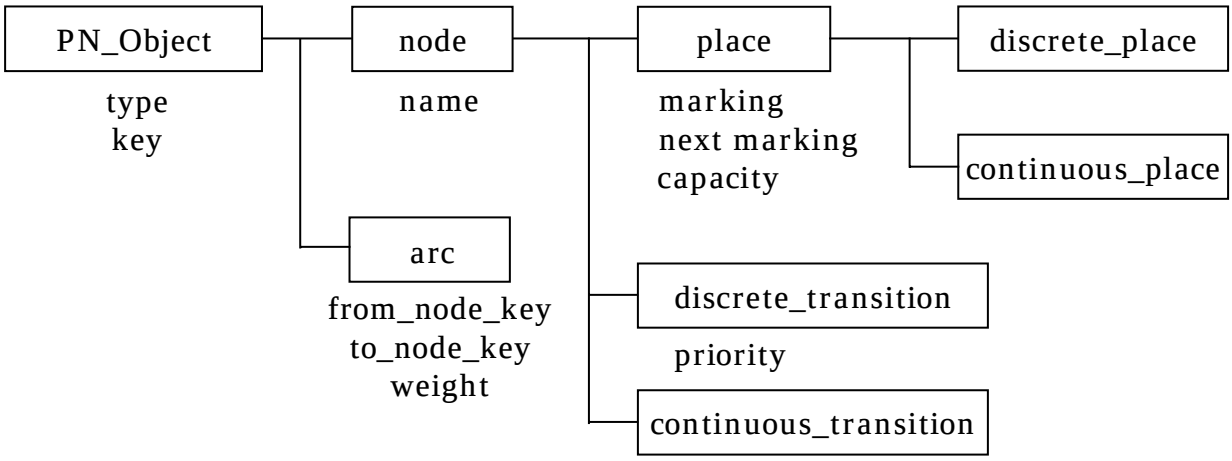


図4.2　クラス相関図III

続トランジション)クラスの２つのクラスに分割されている。この違いは、離散プレースと連続プレースは扱えるトークンが整数か実数かの違いはあるが動作としては同じであるのに対して、離散トランジションと連続トランジションでは扱えるトークンの違いだけでなくその動作にも違いがあるからである。具体的には、連続トランジションは分割された小ステップごとに発火条件が満たされていれば発火するが、離散トランジションは、発火条件が満たされた瞬間瞬時に発火するようになっているからである。

図 4.2 からも分かるように、本システムは離散部分と連続部分の両方を含むハイブリッドペトリネットシステムである。したがって、今後は特に断りがない限りハイブリッドペトリネットシステムとして論を進める。

4.3　実装

ここでは、ハイブリッドペトリネットシステムの実装について説明する。表 4.1 は離散トランジション、連続トランジションのそれぞれについて、入力プレース、出力プレースとして離散、連続どちらのプレースを取るか、またそのとき接続するアークのタイプは何かを示したものである。

表 4.1　プレースとトランジションの接続とその時のアークタイプ

	入力		出力	
	離散プレース	連続プレース	離散プレース	連続プレース
離散トランジション	normal inhibitor test_arc	normal inhibitor test_arc	normal	normal
連続トランジション	inhibitor test_arc	normal inhibitor test_arc	なし	normal

表 4.1 より、出力アークとしては「normal」タイプだけであるが、これは「inhibitor」、「test_arc」ともにトークンの移動が行われないため出力アークとしては無効であるからである。また、連続トランジションと離散プレースの接続において、入力側では「inhibitor」、「test_arc」、出力側では接続できるアークはなしとなっているが、これは、離散プレースのトークンは離散値しか取れず連続的な変化が認められていないため、連続トランジションと離散プレースの接続はトークンが移動しないものとするためである。

以下にハイブリッドペトリネットシステムのアルゴリズムを示す。

i. 各オブジェクトの作成およびパラメータの設定。
ii. 離散トランジションを、発火可能か調べる。
iii. トランジションが発火可能ならば、「normal」アークで接続されているすべての入力プレースから「marking」を「weight」分だけ減らし、発火。
iv. 発火後、「normal」アークで接続されているすべての出力プレースの「next_marking」を「weight」分だけ増やす。
v. すべての離散トランジションについて ii、iii、iv を行う。
vi. すべてのプレースについて「marking」と「next_marking」を足し合わせ新しいマーキングを得、それをファイルに出力。
vii. 発火する離散プレースがなくなるまで ii にもどる。

viii. 連続プレースを、発火可能か調べる。

ix. トランジションが発火可能ならば、「normal」アークで接続されているすべての入力プレースから「marking」を「weight」÷ステップの刻み数減らし、発火。

x. 発火後、「normal」アークで接続されているすべての出力プレースの「next_marking」を「weight」÷ステップの刻み数増やす。

xi. すべての連続トランジションについて viii、ix、x を行う。

xii. すべてのプレースについて「marking」と「next_marking」を足し合わせ新しいマーキングを得、それをファイルに出力。

xiii. 任意の回数（必要なステップ数×ステップの刻み数）だけ ii に戻る。

処理 ii から処理 vii は離散部分、処理 viii から処理 xii は連続部分の処理である。ここから、条件を満たした瞬間発火する離散部分が連続部分に優先して処理される事がわかる。

4.4 例題

ここでは、開発したハイブリッドペトリネットシステムを用いて図 4.3 に示すハイブリッドペトリネットをとく。

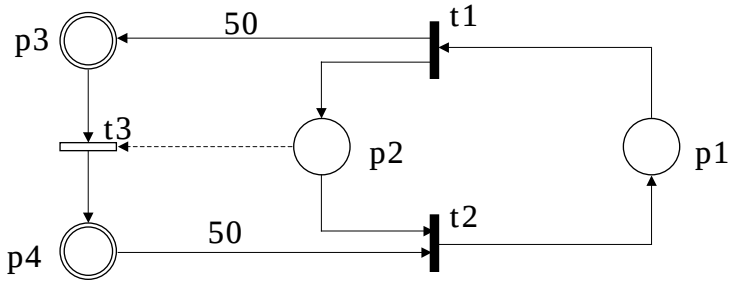


図4.3 ハイブリッドペトリネットの例

これは、1 ロットが 50 個の部品をロット単位で生産する機械のモデルである。離散プレース p1、p2 は作業が休止中と実行中を表し、連続プレース p3 と p4 内のトークン数は未生産の部品および生産済みの部品を表す。離散トランジション t1、t2 は生

産の開始と終了を意味し、連続トランジション t3 は部品の生産を表す。

初期マーキングとして、M_0=(1,0,0,0)を与え、ステップの刻み数を 10（刻み幅 0.1）として計算を行った結果を図 4.4 に示す。

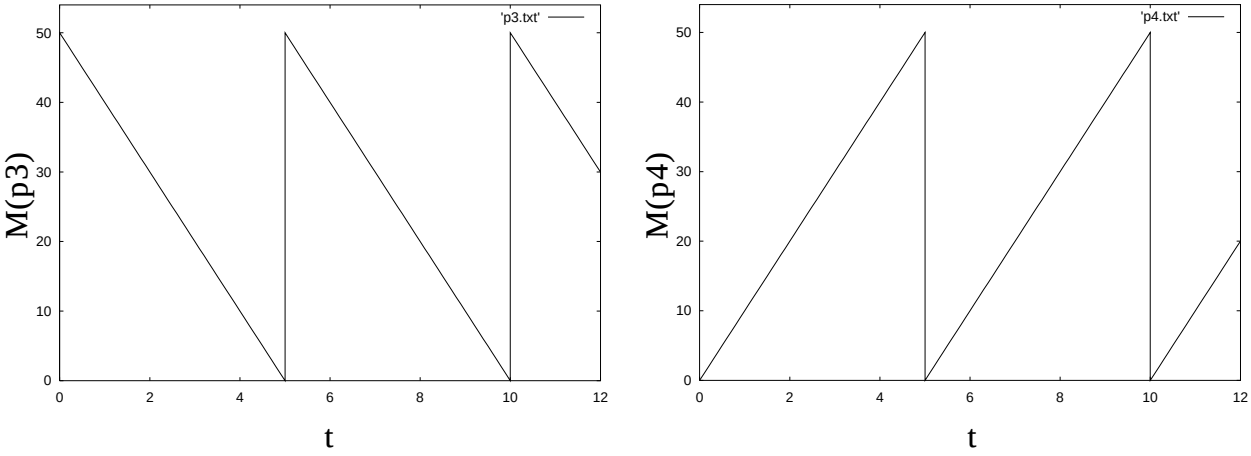


図4.4　例題のマーキングM(p3)、M(p4)の挙動

時刻 t＝0 で p1 にトークンがあることから離散トランジション t1 は即座に発火して M_0=(0,1,50,0)となり、連続トランジション t3 の連続発火が開始、p3 と p4 の挙動は図 4.4 のように変化する。t＝5 のとき M(p3)=0 となり t3 の発火は終了する。それと同時に、離散トランジション t2 の発火条件が満たされるため、t2 が発火し M(p1)=1、M(p2)=0、M(p4)=0 となる。この時点ですでに t1 の発火条件が満たされるため t1 は即座に発火して、M_5=(0,1,50,0)となる。以下同様の振る舞いを繰り返される。

4.5　拡張

ここまで、トークンの移動量はアークの重みによって決定されてきた。しかし、一般的な化学反応や、生体の酵素反応速度論に見られるように入力側の量や濃度などによって、発火速度を変化させたいときがある。ここでは、そのような要望を満たすため、ハイブリッドペトリネットシステムをさらに拡張する。

4.5.1　常微分方程式の数値解法

物理学の法則や工学的な数理モデルは、微分方程式の形であらわされることが多い。なぜかというと、微分とは「変化を数学的にあらわしたもの」だからである。したがって、何か変化するものを予測しようという場合には、ほとんど必ず微分方程式が関係してくると言ってよい

生体内での重要な化学反応のひとつである酵素反応も、そのもっとも基本的なミカエリス・メンテン(Michaelis-Menten)の速度式に見られるように酵素や基質等の濃度をその変化の時間微分である(濃度変化の)速度から求められる[24]。

以下の常微分方程式の説明は文献[5]に拠った。常微分方程式の問題は、常に１階の微分方程式の組に置き換えて考えることができる。たとえば２階の微分方程式

$$\frac{d^2y}{dx^2} + q(x)\frac{dy}{dx} = r(x)$$

は新しい変数 z を補助的に用いて、連立の１階微分方程式

$$\frac{dy}{dx} = z(x)$$
$$\frac{dz}{dx} = r(x) - q(x)z(x)$$

に書き直すことができる。これは、どのような常微分方程式についても例外なく当てはまる手順の例である。新しい変数を選ぶには、本来の変数の微分になるようにすればよい。

このように、一般の常微分方程式の問題は、関数 y_i、i = 1,2,…,N に対して

$$\frac{dy_i(x)}{dx} = f_i'(x, y_1, ..., y_N) \qquad (4.1)$$

の一般形をした N 元連立１階微分方程式の帰着する。

常微分方程式は、その方程式だけで完全に問題が定まるものではない。問題を数値的にどのようにして解くか決定付ける、重要なものとして境界条件の性質がある。境界条件とは(4.1)の関数 y_i の値についての条件を代数式で表したものである。一般にそれらは離散的な特定の各点で満たされるが、それらの点の間の領域では成り立たない。つまり境界条件は微分方程式によって自動的に保存されるものではない。境界条件は、変数がある数値をとるよう求めるだけの単純なものから、変数間に非線形の関係式が

成り立つよう求める複雑なものまである。

この境界条件は大きく分類すると二つに分けられる。

a. 初期値問題(initial value problems)では、すべての y_i について出発点 x_i での値を与え、終点 x_f または（例えば、表の形で表したときのように）離散的に並んだ点での y_i の値を求めることを目的とする。
b. 一方、二点境界値問題(two-point boundary value problems)では、境界条件を２点以上の x で定める。普通は、出発点 x_s と終点 x_f で定める。

ここでは、初期値問題だけを考えることにする。初期値問題を解くルーチンの基礎にある考え方は次のようになる。式(4.1)において dy と dx を有限な Δy と Δx に書き換え、式の両辺に Δx をかける。こうすることにより、独立変数 x が「刻み幅」 Δx だけ移動したときの関数の変化に対応する代数式が与えられる。刻み幅を非常に小さくすれば、元の微分方程式のよい近似となる。この方法をそのまま実行すればオイラー法（Euler method）になるが、この方法は精度に問題があるため実際に用いられることはあまりない。しかしオイラー法の概念そのものは重要であり、実際に用いられる解法のすべては同じ考えに行き着く。その考えとは、式(4.1)の右辺の微分に刻み幅をかけたものに相当する小さな変化を関数に加えることである。

本システムは、常微分方程式の初期値問題の数値解法としてルンゲ・クッタ法（Runge-Kutta method）[25]を採用することにした。ルンゲ・クッタ法は、いくつかのオイラー型のステップから得られる情報を結び付け、その情報を使ってテイラー(Taylor)級数展開の、ある高次の次数まで一致させることにより、１つの区間に渡る解を求める方法である。この方法によって「刻み幅」 Δx が同じ大きさの場合、オイラー法に比べ格段に高い精度が得られる。

本システムにおいてルンゲ・クッタ法を採用したのは、この解法は早い（効率のよい）解法ではないが、どのような問題に対しても充分な精度の解が得られる解法であるからである。次節では、オイラー法の問題点を指摘するとともにそれを解決するルンゲクッタ法について詳しく説明する。

4.5.2　オイラー法の問題点

オイラー法の公式は

$$y_{n+1} = y_n + hf'(x_n, y_n) \qquad (4.2)$$

であり、これはx_nから $x_{n+1}=x_n+h$ へと解を進展させるものである。公式は非対称である。すなわち、図 4.5 に示すように解を区間 h にわたって進展させるが、区間の出発点だけの微分値を用いている。このことは、級数展開でも証明できるようにステップの誤差が補正項 $hf'(x_n,y_n)$より h について１次少ないだけであることを意味する。つまり、式(4.2)は $O(h^2)$の誤差が加算される。

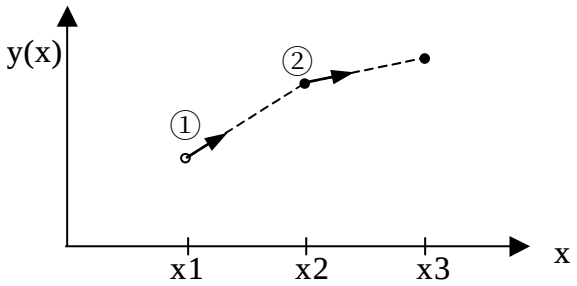


図4.5　オイラー法

オイラー法が実用に向かない理由として、同じ刻み幅で実行したとき他のより良質の方法と比べてあまり正確でないことがあげられる。

4.5.3　ルンゲ・クッタ法

そこで、式(4.2)のようなステップを用いて、区間の中点まで「仮」のステップをとることを考える。そして、その中点での x と y の値を使って、区間全体にわたる「真」のステップを求める。図 4.6 はこの方法を示すものである。

これを式で表すと、

$$\begin{aligned} k_1 &= hf'(x_n, y_n) \\ k_2 &= hf'(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}) \\ y_{n+1} &= y_n + k_2 + O(h^3) \end{aligned} \qquad (4.3)$$

となる。誤差項が示すように、この対称化により 1 時の誤差項は消え、方法は２次に

なる（誤差項が $O(h^{n+1})$のとき、方法は n 次であると呼ぶ)。実際、式(4.3)は２次のルンゲ・クッタ法もしくは中点法と呼ばれる。

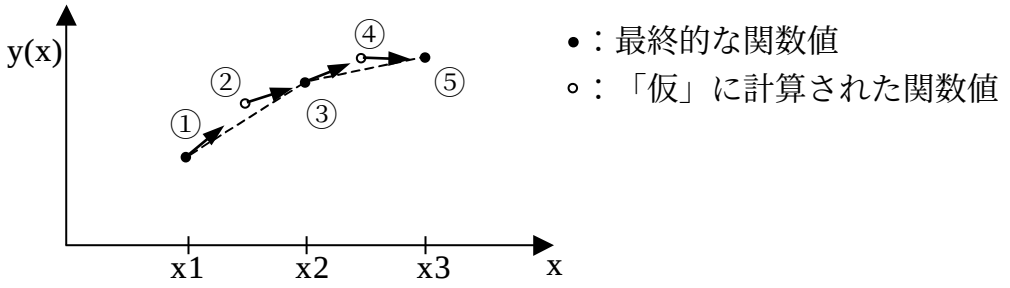


図4.6　２次のルンゲ・クッタ法(中点法)

ここでさらに、「仮」のステップを増やすことを考える。図 4.7 は４次のルンゲ・クッタ法を示している。この図より微分がステップごとに 4 回ずつ行われているのが分かる。そのうち、出発点 y_n で 1 回、仮の中点で２回、仮の終点で１回である。これらの微分の計算を経て、最後に関数値 y_{n+1} が得られる。

これを式で表すと以下のようになる。

$$
\begin{aligned}
k_1 &= hf'(x_n, y_n) \\
k_2 &= hf'(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}) \\
k_3 &= hf'(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}) \\
k_4 &= hf'(x_n + h, y_n + k_3) \\
y_{n+1} &= y_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + (h^5)
\end{aligned}
\qquad (4.4)
$$

4 次のルンゲ・クッタ法では、ステップ幅 h ごとに式(4.1)の右辺について計算が 4 回必要である。この方法が中点法より勝るためには、式(4.4)で少なくとも 2 倍の大きさのステップ幅をとっても同じ精度が得られなければならない。実際には、たいていはそうであるが、必ずというわけではない。つまり、高次であることは必ずしも高精度を意味するものではない。ただし、「一般に」４次のルンゲ・クッタ法は２次より優れているといえる。また、4 次のルンゲ・クッタ法は一般に、より高次のルンゲ・クッタ法より優れていることが知られている。これは、４次より高い次数 M に対し、

関数の計算は M より多い回数を必要としてしまう。このように、４次はルンゲ・クッタ法の次数の１つの分岐点となっている。

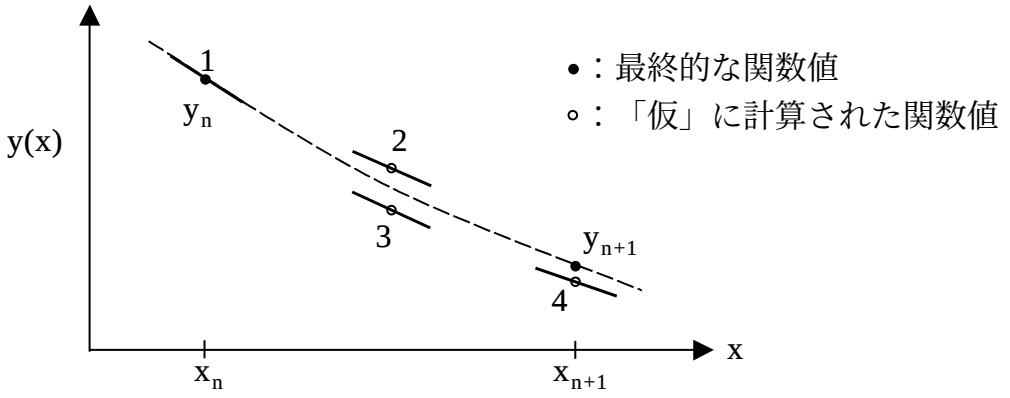


図4.7　4次のルンゲ・クッタ法

4.5.3　実装

本システムでは、連続トランジションの発火速度を４次のルンゲ・クッタ法を用いて求めることにした。そこで、クラス「continuous_transition」に $f'(x_n,y_n)$を示すメンバー変数「speed_function」と計算時にプレースを一意に定めるためクラス「place」にメンバー変数「variable_name」を加えた。

このときのクラスの相関図を図 4.8 に示す。

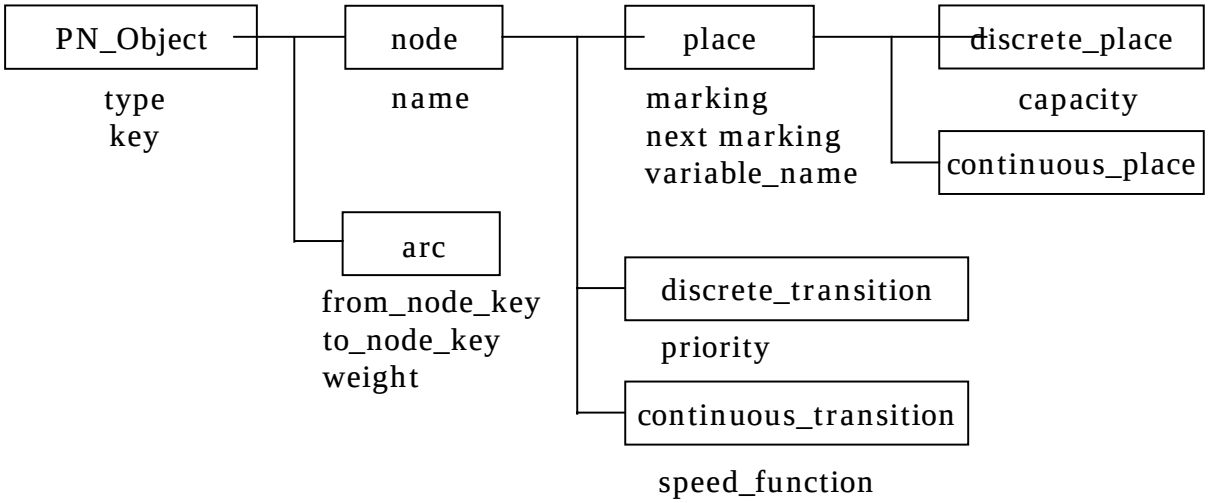


図4.8　クラス相関図Ⅳ

「variable_name」はプレースごとにユニークに決められ、「speed_function」内で各プレースの値を求めるときに使われる。「speed_function」では、加減乗除の四則演算のほかに表 4.2 で示す関数が使用できる。

表 4.2　speed_function で使用できる関数

関数	関数の機能
Sin(x)	x ラジアンのサインを計算
Cos(x)	x ラジアンのコサインを計算
Tan(x)	x ラジアンのタンジェントを計算
Cot(x)	x ラジアンのコタンジェントを計算
Exp(x)	e の x 乗を計算
SQR(x)	x の２乗を計算
Sqrt(x)	x のルート（２乗根）を計算
Sign(x)	x が正なら１、負なら-1、ゼロならば 0 を返す
ABS(x)	x の絶対値を返す

4.6　例題Ⅱ

ここでは、連続トランジションの発火速度が「speed_function」により動的に決定される系を例題として解く。

また、参考として、オイラー法を用いて計算する「Visual Object Net++」[9],[10],[11](以下 VON)と、適応刻み幅制御(adaptive stepsize control)を行う陰的(implicit)ルンゲ・クッタ法[6]を用いて計算する「MATLAB」を用いて同じ計算を行ってみた。

4.6.1　一般的な系

ここでは、簡単な例として、外力のない相互作用する剛体の挙動についての連立微分方程式を解く。式は次のようになる。

$$y1' = 0.1*y2*y3$$

$$y2' = -0.1*y1*y3$$

y3' = -0.051*y1*y2

これをペトリネットで表すと図 4.9 のようになる。

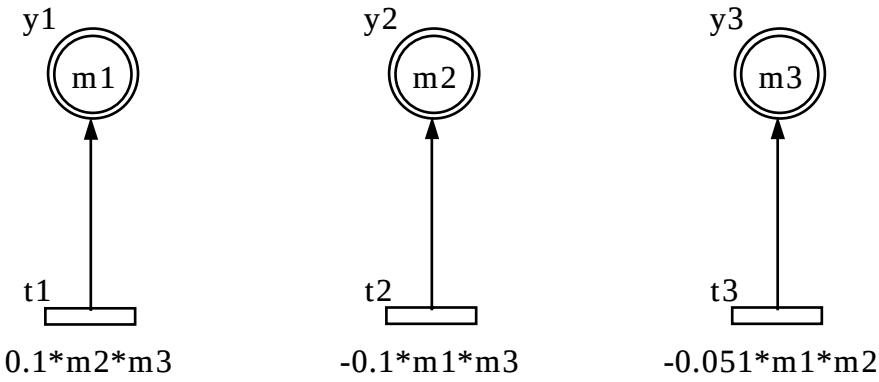


図4.9　ペトリネットを用いた外力のない剛体の挙動についての微分方程式系の表現

ここで、m1、m2、m3 は各プレースの「variable_name」である。これを初期マーキング M_0=(0,1,1)で解いた結果が図 4.10 である。

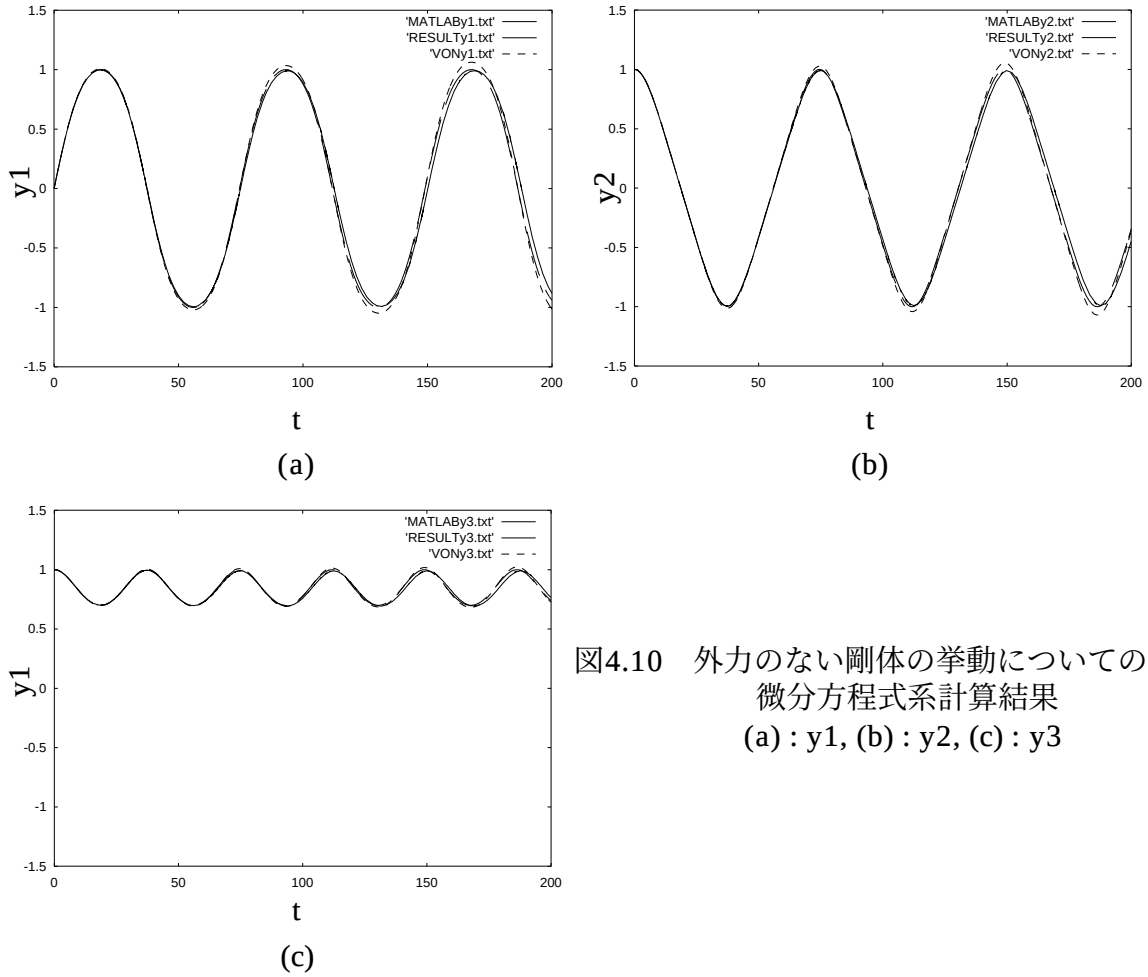


図4.10　外力のない剛体の挙動についての
微分方程式系計算結果
(a) : y1, (b) : y2, (c) : y3

この図からも分かるように点線で表されているオイラー法を用いた VON は周期を繰り返すごとに他の 2 つの計算結果から離れている。図 4.11 は t＝3000 まで計算したときの y1 の極大値を示したものである。ここでは先に述べた傾向が顕著に表れ、オイラー法による計算結果は発散に向かっている事がわかる。ここから、オイラー法の計算精度が高くないことが分かる。それに対し、4 次のルンゲ・クッタ法を使用した本システムと適応刻み幅制御ルンゲ・クッタ法を使用した MATLAB は、振幅が 2（極大値は 1）のままで推移しており高い精度を保っているといえる。

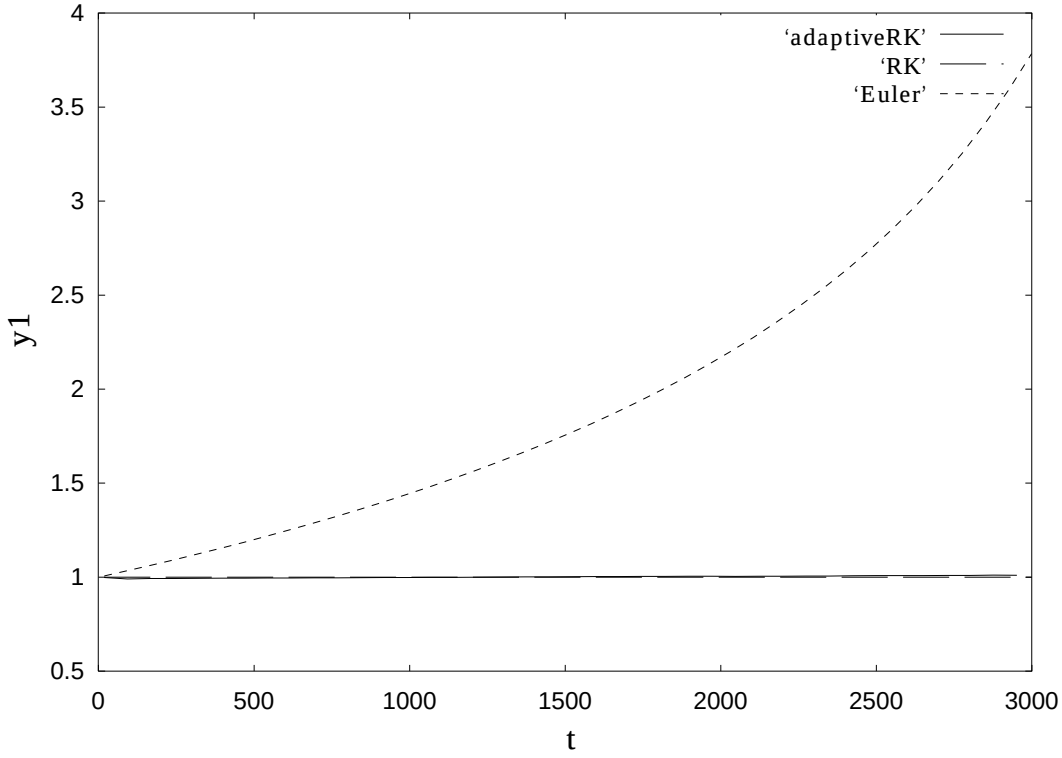


図 4.11　y1 の極大値の時系列

（オイラー法による計算結果の振幅は指数的に発散する。）

4.6.2　硬い(stiff)系

2 つ以上の微分方程式を扱っていると、硬い(stiff)連立微分方程式が出てくることがある。この現象は、従属変数の変化にかかわる独立変数が、非常に異なるスケール（大きさの程度）を持つときに生じる。

ここでは硬い連立微分方程式の例として van der Pol 方程式を解く。式は次のよう

になる。

$$y1' = 0.01*y2$$

$$y2' = (1-y1^2)*y2 - y1*y2$$

これをペトリネットで表示すると図 4.11 のようになる。

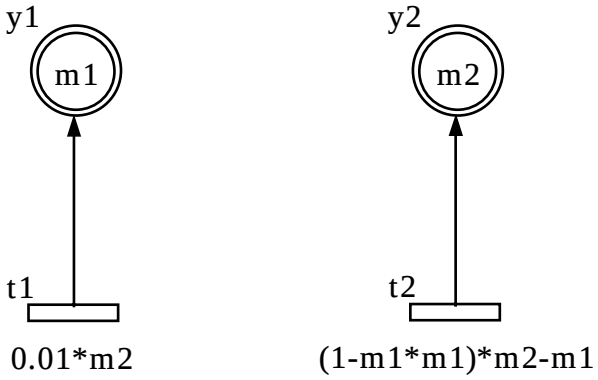


図4.12　ペトリネットを用いた硬い系(van der Pol方程式)の表示

これを初期マーキング M_0=(0,1)で解いた結果が図 4.12 である。

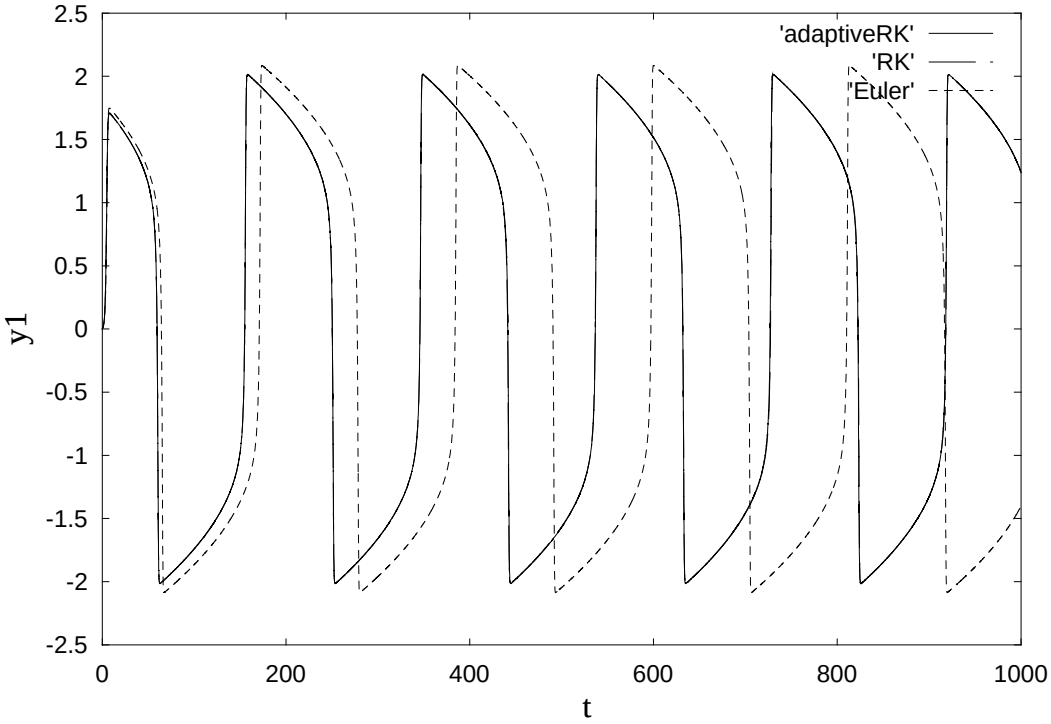


図 4.13　硬い系の計算結果（y1 の時系列）

（オイラー法による計算結果は周期が次第にずれていく。）

この図から、点線で表されているオイラー法を用いた VON は周期を繰り返すごとに他の 2 つの計算結果から離れている様子が見て取れる。ここからオイラー法の計算精度が高くないことだけでなく硬い系においては安定ではないことが分かる。それに対し、本システムは硬い系の計算にも充分耐えられる能力を持っているといえる。

図 4.14 は系の硬さを変えたときのオイラー法とルンゲクッタ法の周期の差がどのようになるかを示したものである。

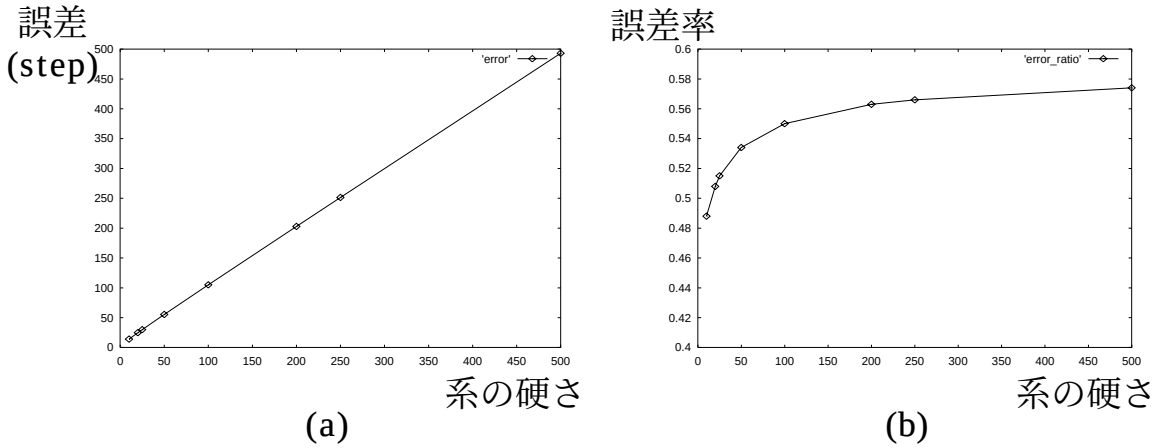


図 4.14　系の硬さを変えたときオイラー法とルンゲ・クッタ法の誤差と誤差率
（誤差としてオイラー法とルンゲ・クッタ法の第 5 周期の極大値が何ステップ目に表れるかの差をとった。誤差率はそれぞれの系の硬さに応じた周期の長さで誤差を割ることにより求めたものである。）

(a)は、van del Pol 方程式の y1’と y2’の係数の比を 10 倍から 500 倍にしたときのオイラー法とルンゲ・クッタ法の誤差（単位：step）を示したものである。横軸は入出力の係数の比を系の硬さとして示している。この図から、系の硬さに比例して誤差が増加しているのが分かる。(b)は先に求めた誤差を、それぞれの系の硬さに応じた周期の長さで誤差を割ることにより求めた誤差率を示したものである。誤差率は系の硬さが低いときに急激に上昇するが、系の硬さがある程度以上になるとほぼ平衡状態となっている。また、系の硬さが 50 程度で誤差率が 0.5 になっていることから、系の硬さがそれほど高くなくてもオイラー法の周期のずれは非常に大きいことが分かる。

4.7　　おわりに

この章では、まず第３章までに作成したペトリネットシステムを拡張し、連続量、離散量ともに扱えるハイブリッドペトリネットシステムを作成した。

そこからさらに、今までアークの重みによって決定されてきたトークンの移動量を、一般的な化学反応や、生体の酵素反応に見られるように入力側の量や濃度などによって、発火速度を変化させられるように拡張した。

計算方法として、４次のルンゲ・クッタ法を使用してこれを例題にかけた結果、良好な結果を得ることができた。

第　5　章
細胞周期のシミュレーション

5.1　　はじめに

ハイブリッドペトリネットを用いると、タンパク質分子などをプレースで表現し、その相互作用などをトランジションで表すことにより、生体分子ネットワークを容易に記述することができる[9],[13],[14],[16]。

生体分子ネットワークのシミュレーションにペトリネットを用いることの利点は、グラフィカルな表現による直感的な理解のしやすさと、タンパク質の濃度と遺伝子の発現制御の関係をデータフローと制御フローにより統一的に記述できる点にある。この章では生体分子ネットワークのひとつである細胞周期調節系(cell-cycle control system)を、作成したペトリネットシステムでシミュレーションを行った。

5.2　　細胞周期調節系

ここでは文献[15]を参考に細胞周期調節系についての説明を行う。細胞周期とは、細胞が分裂する周期のことでありあらゆる生物が増える基本で、酵母や細菌などの単細胞生物は細胞分裂ごとに新しい個体が１つ増える。多細胞生物では新しい個体を作るため、また、個体が成長するために細胞分裂を繰り返さなければならないし、成体になってからも消耗や損傷あるいはプログラムされた細胞死を通じて失われる細胞を補うのに細胞分裂は不可欠である。細胞周期の長さは細胞の種類によって大きく異なる。ハエの胚では最も短く 8 分間程度だが、哺乳類の肝細胞では 1 年以上にも及ぶ。図 5.1 は約 24 時間周期で細胞分裂する細胞の細胞周期の区分を示したものである。

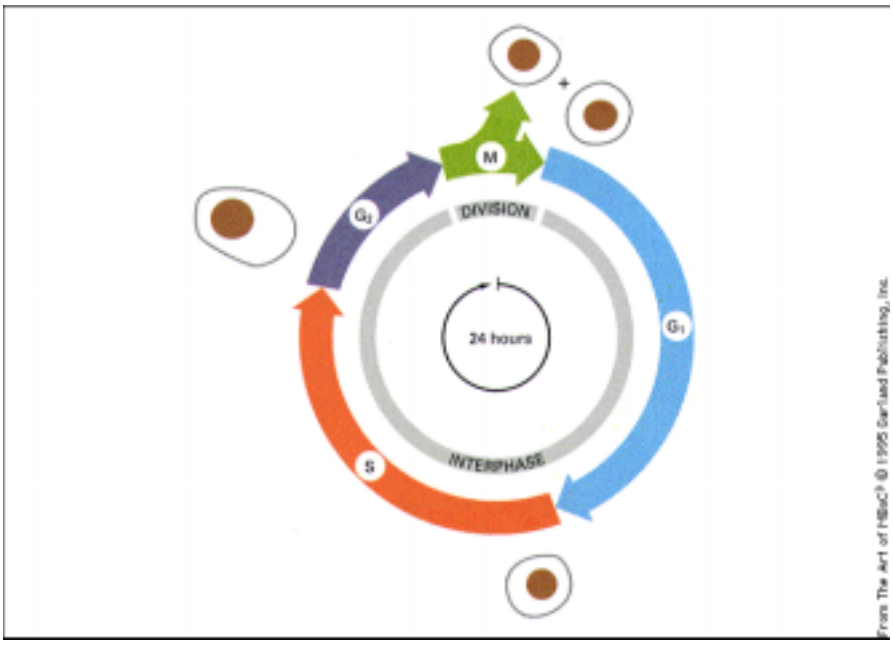


図 5.1　標準的な真核細胞に見られる細胞周期の 4 つの区分(文献[15]より)

図 5.1 に示されているように、細胞周期は 4 つの段階に分かれている。有糸分裂（核の分裂）と細胞の分裂が見られる M 期(M phase, M = mitotic)、核 DNA の複製が行われる S 期(S phase, S = synthesis)、有糸分裂の完了から DNA 合成までの間を G_1 期(G_1 phase, G = gap)、DNA 合成の終了から有糸分裂の開始までの間を G_2 期(G_2 phase)である。G_1 も G_2 も、次の段階に進むための準備をすると同時に、前の時期の出来事が正確になされたかを判定している（チェックポイント機構）。

図 5.1 に示したような、約 24 時間を細胞周期とするようなものとして、哺乳類の癌細胞などが上げられるが、この場合、M 期は 0.5-1 時間、S 期は 2-4 時間、G_2 期は 4-6 時間、残りの約 15 時間は G_1 期である。

先にも述べたように、細胞周期の長さは細胞の種類によって大きく異なるが、例えば受精直後の初期胚では、図 5.2 に見られるように G_1、G_2 期が極端に短く 8～60 分程度の細胞周期であり、逆に神経細胞や肝細胞のように G_1 期が非常に長いため細胞周期が長くなっているものもある。G_1、G_2 期はともに細胞の成長・維持に必要な時間であり、初期胚のように G_1、G_2 期が極端に短い細胞周期では、分裂で生じた娘細胞は親の細胞の半分の大きさになり初期胚の大きさは変わらない。

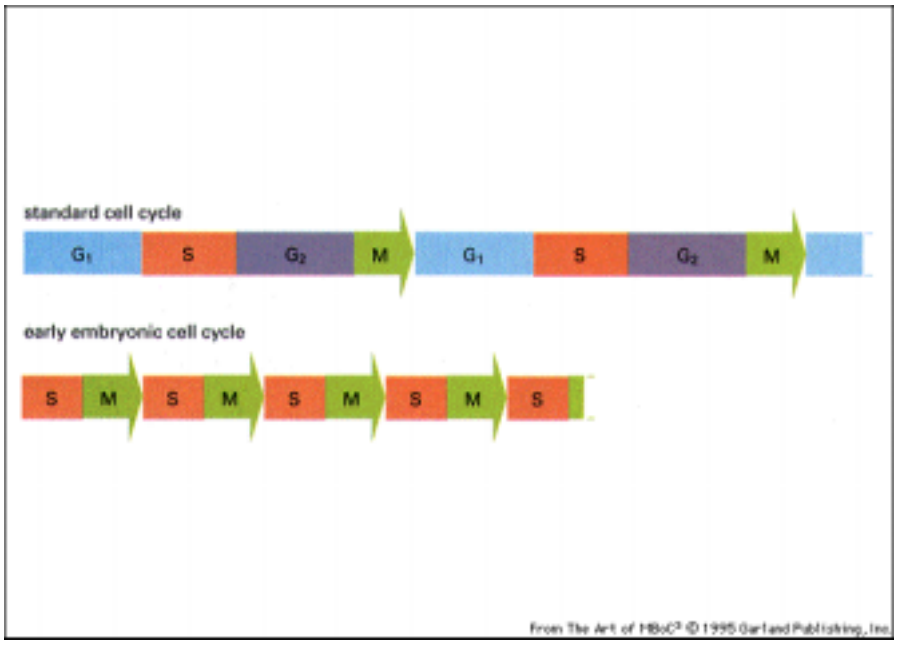


図 5.2　標準的な細胞周期と初期胚の細胞周期の比較(文献[15]より)

細胞周期調節系は 2 群のタンパクファミリーからできている。第一はサイクリン依存性キナーゼ(cyclin-dependent protein kinase : Cdk)ファミリーで、特定のタンパク質のリン酸化を介して、下流の過程を誘導する。

第二はサイクリン(cyclin)と呼ばれる、活性化タンパクファミリーで Cdk と結合してこのキナーゼが適切な標的タンパクをリン酸化する能力を調節する。サイクリン - Cdk 複合体の周期的な組み立て、活性化、分解は細胞周期のかなめをなしている。サイクリンと呼ばれるのは、その合成と分解が周期的に起こるためである。サイクリンは 2 種類に分けることができる。有糸分裂サイクリン(mitotic cyclin)は G_2 期に Cdk 分子と結合して有糸分裂への進行に働き、G_1(G_1 cyclin)サイクリンは G_1 期に Cdk 分子と結合して S 期への進行に働く。

M 期を制御する M 期促進因子(M-phase-promoting facter : MPF)は、Cdc2 と呼ばれるサイクリン依存性キナーゼ(Cdk)とサイクリン B(Cyclin B)と呼ばれる有糸分裂サイクリンからなっている(図 5.3)。

Cdc2 は細胞周期を通じて一定量存在し、リン酸化、脱リン酸化の影響を受ける。これに対して、サイクリン B は、MPF 活性が最も高い M 期にタンパク量が最も大きくなり、逆に MPF 活性が低い G_1 期には発言が見られないなど細胞周期全体を通じ

て量的制限を受ける。このような２元的制御は細胞周期特異的キナーゼの特徴で、他の Cdk でも同じような制御を受けるものがある。

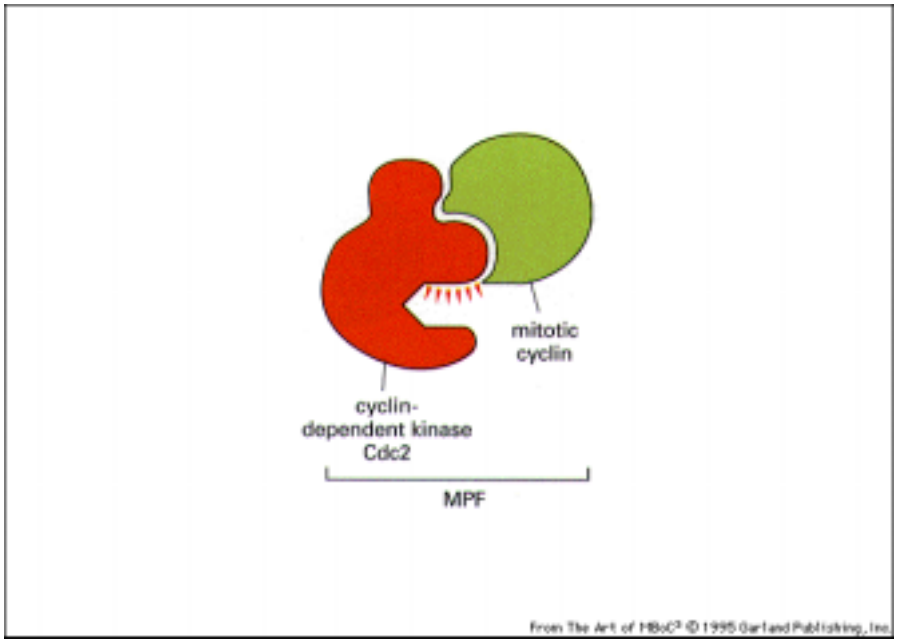


図 5.3　MPF の 2 つの主要サブユニット(文献[15]より)

サイクリン依存性キナーゼ Cdc2 はチロシン 14(T)、15(Y)とスレオニン 160(T)という 3 つのリン酸化部位を持つ。14(T)、15(Y)部位のリン酸化は Wee1 キナーゼによって、160(T)部位のリン酸化はスレオニンキナーゼ(Cdc2-activating Kinase：CAK)によって行われる。CAK によるリン酸化は Cdc2 の活性化型のリン酸化であるが、Wee1 による 14(T)、15(Y)部位のリン酸化は抑制的リン酸化である。MPF が活性化するためには 160(T)部位のリン酸化と 14(T)、15(Y)部位の脱リン酸化が必要である。脱リン酸化は Cdc25 フォスファターゼによって時期特異的になされる。Cdc25 には Cdc25A、Cdc25B、Cdc25C の 3 種類のサブタイプがある。このうち MPF の活性化に関与するのは Cdc25B と Cdc25C である。また、160(T)部位の脱リン酸化は G_1 期に見られるが、未知の IHP フォスファターゼによってなされることが予測されている。以上より MPF は細胞周期遷移中、リン酸化状態の違いにより 4 つの状態をとり得る。

Cdc25B はサイクリンに似た量的変化を示し、フォスファターゼとしての活性はタンパク量に比例している。これに対して Cdc25C は細胞周期を通じてタンパク量の変

化は見られないが、MPF によってリン酸化されないとフォスファターゼとしての活性は上がらない。G_2期からM期への遷移の際、最初にMPFを活性化するのはCdc25Bであるが、いったんMPFが活性化されると、それが極少量であってもCdc25Cがリン酸化されフォスファターゼ活性が上昇し、最終的に活性化型MPFの量が一気に上昇する。

これに対して Wee1 は 14(T)、15(Y)部位のリン酸化を行っているキナーゼだが、Cdc25 とは反対に、脱リン酸化型が活性化型である。MPF によるリン酸化は Wee1 の活性レベルを下げる。以上より MPF の活性化には正のフィードバックが存在する。したがって、M 期に入り MPF の活性がある程度進むとそこから急激に活性化型 MPF の濃度が上昇する。

先ほどサイクリン B は細胞周期を通じて量的制限を受けると述べたが、M 期に活性化されていた MPF は M 期後半より制御因子のサイクリン B が特異的なユビキチン依存性のタンパク分解を受け消失する。活性化型 MPF は M 期に中間酵素(Intermediary Enzymes : IE)をリン酸化し、この IE が分裂時期特異的ユビキチンリガーゼ APC(Anaphase-promoting complex)を活性化する。APC は活性を持つと、サイクリン B に ATP 依存的にユビキチンタンパクを付加し、ユビキチン化されたサイクリン B はプロテアソームによって分解される。これはサイクリン B の負のフィードバックであり、M 期が終了すると速やかに MPF 活性が低下する理由となっている[12],[15]。

以上が MPF を中心とする細胞周期の概略である。細胞周期の生物システムは、遺伝子の転写調節やシグナル伝達に比べて、細胞周期を制御するほとんどの因子がフィードバックを持ち、また、全体としてループ状の伝達体系をもつことが特徴である。すなわち、タンパクのリサイクリング機構によって最終的にシグナルが周期的に何度も現れるという特徴をもっている。シグナル伝達などの一方向性的な伝達系とは異なるシステムとなっている。

5.3 細胞周期モデル

細胞周期をモデル化する試みはこれまでに何度となくなされており、Novak と

Tyson らにより、MPF に関わる状態遷移過程が一連の非線形常微分方程式に定式化された例もある[17]。しかし、新たな細胞周期制御因子も数多く同定されつつあり、また、彼らのモデルはアフリカツノガエルの卵（初期胚）の細胞周期をもとにしたものであり、これを細胞シミュレーションにそのまま使用するのはいくつか問題がある。先に述べた通り Cdc2－CyclinB 複合体は、リン酸化部位の違いにより 4 つの状態を取りうるが、彼らのモデルはこの 4 状態を自由に行き来できる。これは in vitro（試験管内）においては正しいが、in vivo（生体内）においては不適切なものである。なぜなら、生体細胞内では実際に起こらない反応が記述されていることがあげられる。例えば、時間的・空間的な理由で反応分子そのものが会合しない場合などである。また、Cdc2- CyclinB 複合体で実細胞内には有り得ない結合がモデル化されていることもあげられる。

このモデルに対し、吉岡らは生物学的な検証を加え、実際の細胞内で起きている現象を可能な限り忠実に反映するように整理し、新たなモデルを構築した[12]。図 5.4 はそのモデルを示したものである。

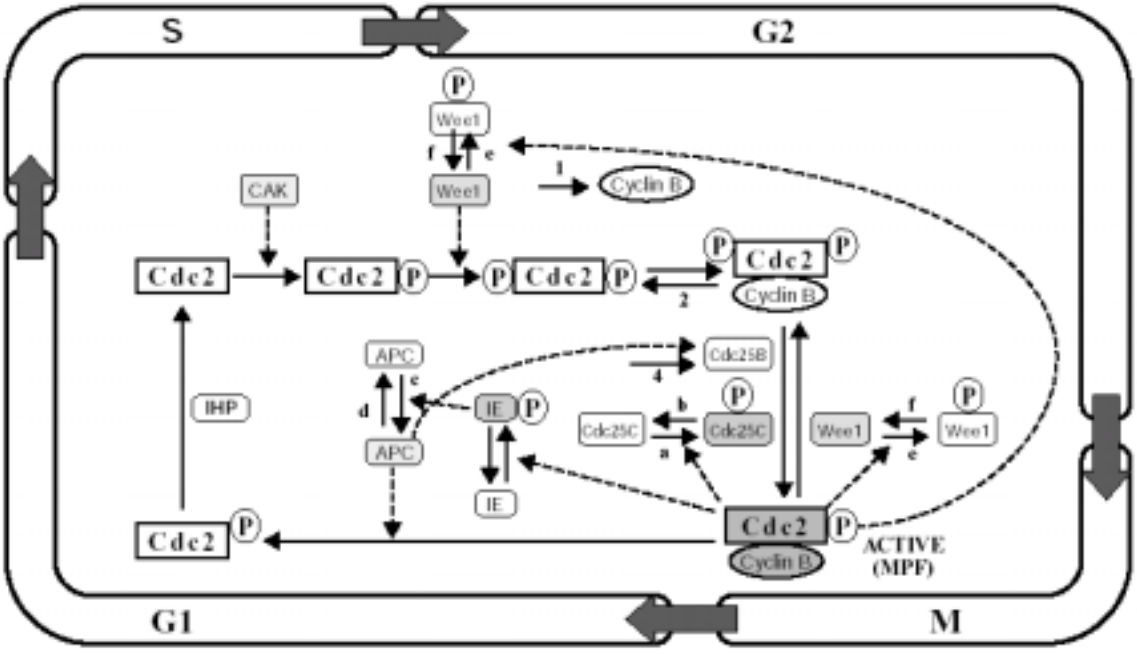


図 5.4　Cdc2 を中心とした細胞周期モデル

上のモデルをペトリネットで記述したものを図 5.5 に示す。

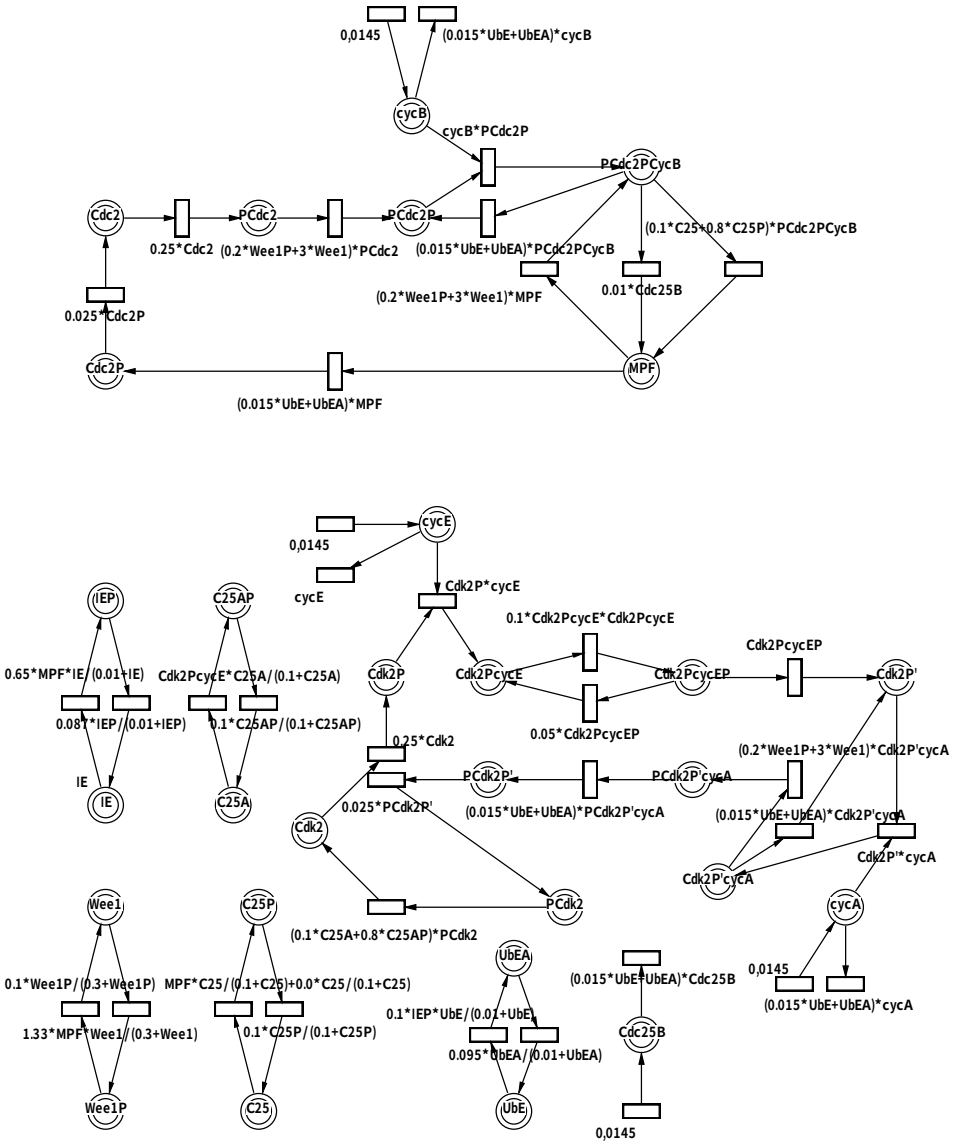


図 5.5　ペトリネットによる細胞周期モデルの記述

図 5.5 上部の、ループ構造となっている部分が図 5.4 で示した Cdc2 を中心とした細胞周期モデルのペトリネットによる記述で、下部は、そのモデルに関連するタンパクキナーゼや活性化タンパクを記述したものである。ここで、各反応の速度定数は文献[17],[18],[19],[20]を参考にしている。

5.4　シミュレーション

前項で示したモデルを本システムで動作させた。図 5.6 は MPF と有糸分裂サイクリンであるサイクリン B の濃度変化状態を示したものである。

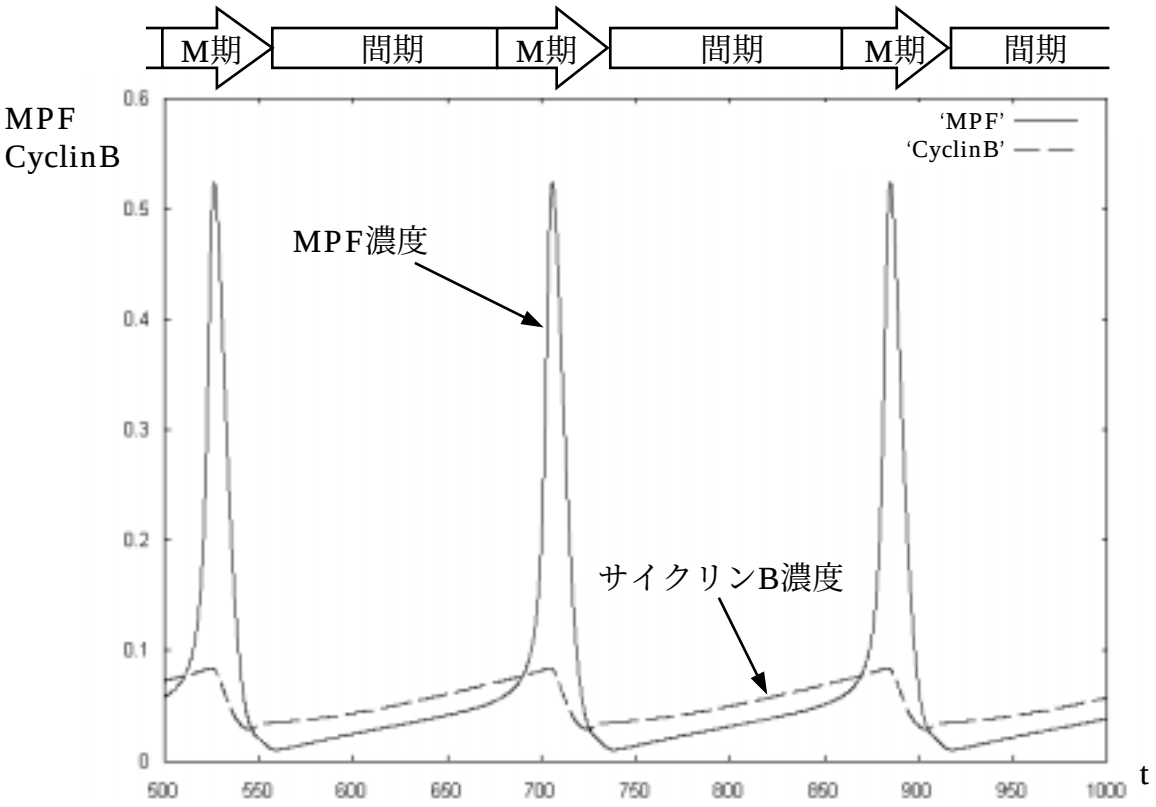


図 5.6　通常の細胞周期における MPF とサイクリン B の濃度の時系列

この図から、5.2 項で述べた MPF とサイクリンの傾向を見ることができる。

一度シミュレーションモデルを構築すると、各パラメータや反応経路の変更により簡単に仮想実験が行えるところは生物実験にはない利点である。

例として、文献[21]によると、大腸癌患者で Cdc25B が過剰に発現しており、このような患者は術後生存期間が短く、Cdc25B の過剰発現は予後不良因子であると報告されている。これをコンピュータ上で再現するため、Cdc25B の発現量を 2 倍として Cdc25B が過剰発現した場合を想定したシミュレーションを行った。図 5.7 はその結

果である。破線が、Cdc25B を過剰発現させた場合の結果である。

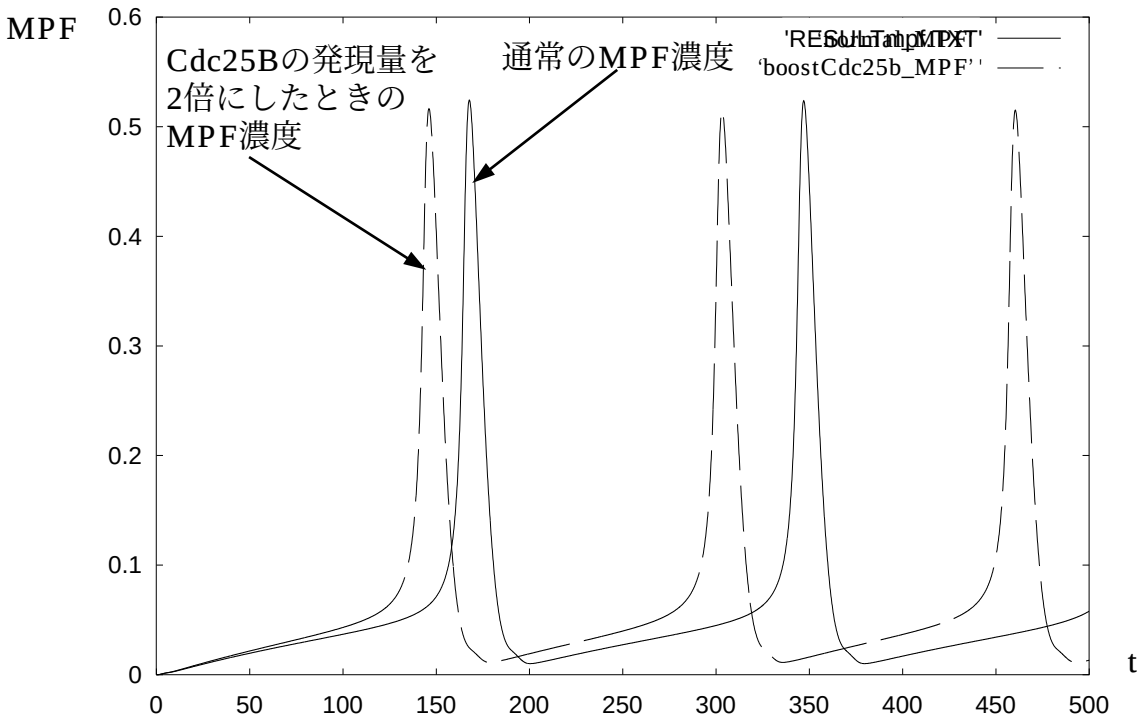


図 5.7　Cdc25B の発現量を増やした場合と通常の場合の MPF 濃度の時系列
（Cdc25B の発現量を増やした場合 MPF 活性の周期が通常より短くなる。）

図 5.7 に示す結果より Cdc25B の発現量を増やした場合、MPF 活性の周期が通常の場合より短くなっておりこのことから細胞の分裂周期が短くなることが予想される。したがって、文献[21]の Cdc25B の過剰発現は予後不良因子であるとの報告をシミュレーションの結果は支持している。

5.5　結論

この章では、開発したハイブリッドペトリネットシステムを用いて、生体分子ネットワークのシミュレーションを行った。

モデルとしては、細胞周期調節系に関する文献[12]のモデルを用いた。そのシミュ

レーション結果と、思考実験として Cdc25B を過剰発現させた場合のシミュレーションの結果を得、これらを生物学的知見と比較して矛盾なく意味付けをすることができた。

第　6　章
位置情報付きペトリネット

6.1　　はじめに

従来のペトリネットを用いた生体分子ネットワークのモデルでは、その反応過程を記述することに終始しており空間的な要素を排除していた。しかし実際には反応物質が偏在し、反応も全体で一斉に起こるのではなくある地点から次第に拡散していくなど空間的な要素を含んでいる。

この章では、空間的な要素を含む系のシミュレーションを行うためにペトリネットシステムを拡張して、位置情報付きのペトリネットシステムを作成する。そして、このシステムを用いて空間的な要素を含む、ショウジョウバエの胚発生におけるパターン形成についてのシミュレーションを行う。

6.2　　拡張

ペトリネットシステムに位置情報をもたせるために、対象となる系の空間を格子状に分割することを想定し、また、タンパク質の濃度や遺伝子発現量を表すプレースのマーキングを整数や実数の単純な値から整数や実数の配列にかえ、この配列のそれぞれの値を分割された空間の一つ一つに対応させることで空間の概念を導入する。図 6.1 はこれを示したものである。

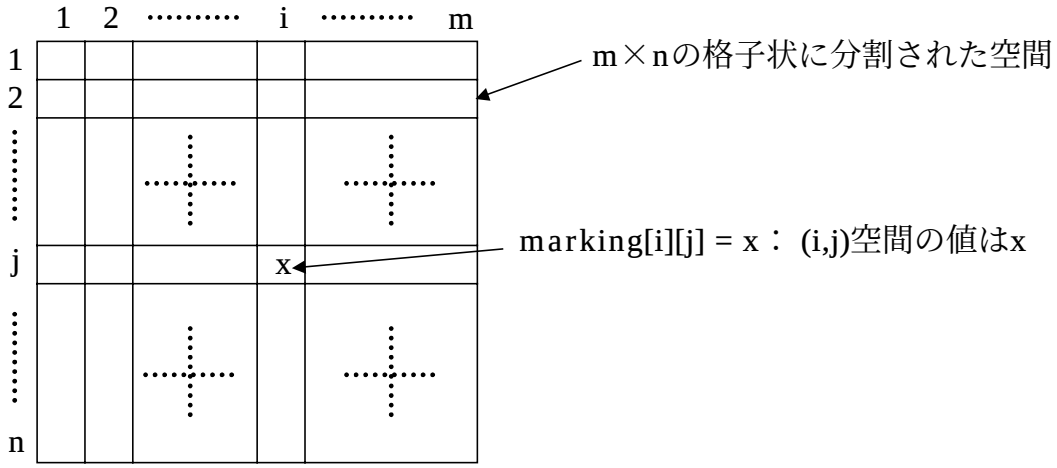


図6.1　位置情報導入の概念

トークンは 1 ステップごとに最近傍（上下左右）に移動可能である。この処理により、物質の拡散を表現することを可能とした。

6.3　ショウジョウバエの胚発生

ここでは、文献[15],[26],[27]および[28]を参考に、ショウジョウバエの胚発生に関して説明する。

ショウジョウバエの卵は長さ約 400μm幅約 160μmであり、極性を持っている。昆虫ではよく見られることだが、卵の発生通常と異なり、細胞分裂なしに核が分裂を繰り返し多核細胞となる。初期の核分裂は同調的に起こり、約 8 分に一回と非常に速い。最初の 9 回の分裂でできた核のほとんどが卵の中心から表面へ移動して一層に並ぶ。これを多核性胞胚(syncytial blastoderm)と呼ぶ。さらに 4 回核が分裂すると、卵の表面から内側に向かって細胞膜が形成されて核を包みこみ、多核性胞胚は約 6000 個の細胞からなる細胞性胞胚に変わる（図 6.2)。

多核性胞胚期では共有する細胞質に多数の核を持つ単一の巨大細胞になっている。しかし、細胞質は均質ではなく胚の長軸方向に沿って不均一に分布した遺伝子調節タンパク質を含んでいる。その不均一な分布によって、肺のある部分と別の部分とを区別する位置情報が与えられ、異なる遺伝子を発現するようになる。

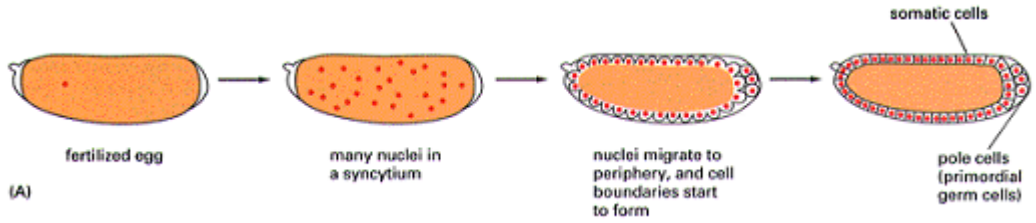


図 6.2　受精から細胞性胞胚期までの卵の発生（文献[15]より）

細胞内の位置を決めるには 2 つの座標が必要であり、これに対応して卵極性遺伝子(egg-polarity gene)も大きく 2 つに分かれている。2 つのグループは、発生開始の際にそれぞれ独立に働いて、胚の主要な 2 本の体軸、前後軸と背腹軸を決定する。これらの遺伝子は卵内にモルフォゲン(molphogen)勾配を設定して胚の空間座標を決定する。

卵極性遺伝子は、卵形成の時期に母親のゲノムから転写され、その産物は受精後すぐに働き始める。したがって、胚の表現系は胚自身の持つ父方と母方の遺伝子の組み合わせではなく、母親の持つ遺伝子（卵の遺伝子）によって決まる。このような様式で発現する遺伝子を母系効果遺伝子(maternal-effect gene)と呼ぶ。

母系効果遺伝子には bicoid、nanos、hunchback、caudal 等がある。hunchback と caudal の mRNA は胚全体に分布しているが bicoid は胚前部、nanos は胚後部に局在している。そして、bicoid は caudal の発現を抑制し、hunchback の発現を促進する。それに対し nanos は hunchback の発現を抑制する。このような遺伝子の相互作用を通じて、初期胚において 4 つのタンパク質の勾配が形成される。

この勾配にしたがって一連の体節ができる過程には多数の分節遺伝子(segmentation gene)が関与している。分節遺伝子は卵極性遺伝子よりも後の時期、つまり胚が母親由来の mRNA を利用するのではなく、自身のゲノムから転写を行うようになってから作用する。このような遺伝子を接合体効果遺伝子(zygotic-effect gene)と呼ぶ。

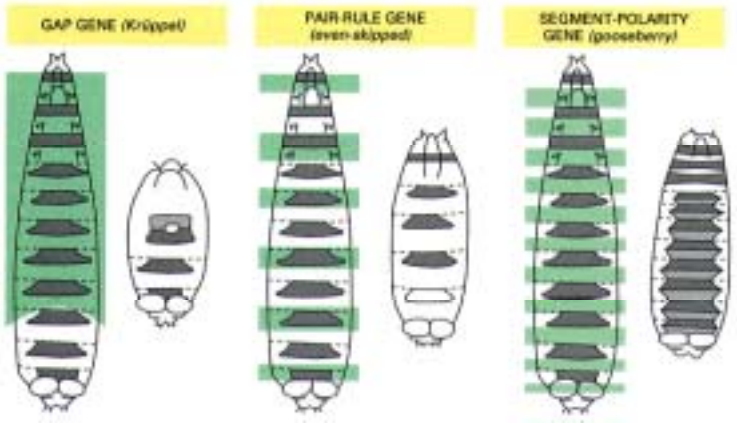
分節遺伝子は、その変異体の表現型とそれがどの段階で作用するかによって大きく 3 群に分類される。最初に作用するのがギャップ遺伝子で、その産物は胚を大雑把に分割する。この遺伝子には、hunchback、Krupple、knirp、giant 等がある。ギャップ遺伝子に変異が生じると 1 ヶ所あるいは数ヶ所で体節が欠損する（図 6.3(a)）。

次に働く遺伝子は、ペア・ルール遺伝子(pair-rule gene)である。この遺伝子によっ

て胚に体節の最初の兆しが現れる。ペア・ルール遺伝子には、hairy、even-skipped(eve)、fushi tarazu(ftz)等がある。この遺伝子に変異が生じると、体節が１つおきに欠失して体節が通常の半分の数しかない胚になる（図 6.3(b)）。

受精後 2、3 時間以内にギャップ遺伝子とペア・ルール遺伝子は順次活性化される。その産物である mRNA が最初に示すパターンは、最終的な像に大まかにしか似ていないが、短時間のうちに何回か調節が起こって、あいまいだった像が規則正しい縞模様になる（図 6.4）。しかしこの系自体は不安定な一時的なものであり、胚発生が進むとギャップ遺伝子とペア・ルール遺伝子による規則正しい分節パターンは消失する。しかし、これらの作用によって胞胚の細胞に永続的な位置標識が植え付けられる。この位置標識の働きで幼虫や成虫が体節構造を維持できる。

最後に、セグメント・ポラリティ遺伝子(segment-polarity gene)が働く。この遺伝子には、gooseberry、engrailed 等がある。この遺伝子に変異が生じると、体節の数は正常だが各体節の一部が欠失し、変わりに残りの部分が鏡像になって重複した構造をもつ幼虫ができる（図 6.3(c)）。



(a)　　(b)　　(c)

図 6.4　3 種類の分節遺伝子に生じた変異の表現系の例（文献[15]より）

胚発生の初期段階に見られる分節化それに伴うパターン形成はこれらの遺伝子の相互作用によって起こる。

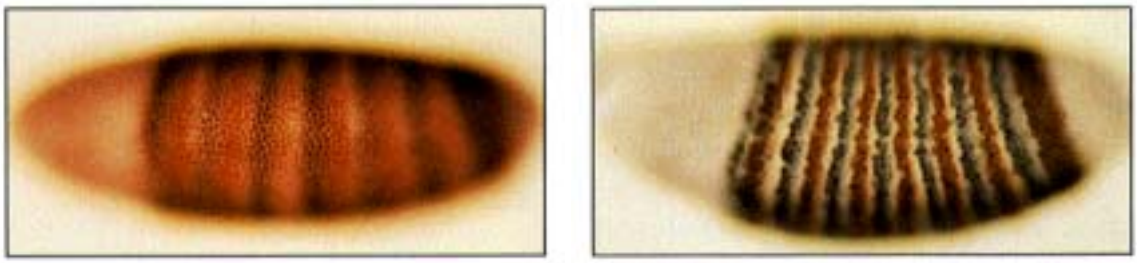


図 6.4　ショウジョウバエ胞胚での ftz 遺伝子と eve 遺伝子による縞模様の形成
（文献[15]より）

6.4　　胚発生時におけるパターン形成のモデル化

ショウジョウバエの分節化に関するシミュレーションはいくつもある[28],[29],[30],[31]が、ここでは文献[31]のモデルをペトリネットを用いて実行する。

文献[31]のモデルは、いくつかの遺伝子調節タンパク質の濃度情報が収められている「細胞」と、その細胞の 1 次元配列である「胚」を場として、遺伝子調節タンパク質の合成、拡散および削除が行われる。遺伝子調節タンパク質がある遺伝子に対して行う抑制あるいは活性化を if-then ルールを用いて表現している。例えば、

(IF (> (ConcentrationOf protein3) 13.611979) protein1)　（文献[31]より）

は「もしタンパク質 3 の濃度が 13.611979 以上ならば、タンパク質 1 を 1 単位合成せよ」と解釈する。

このシミュレーションの大まかな流れは以下のようになる。

1. 胚の初期化。
2. 「合成‐拡散‐削除」ステップと呼ぶ処理を、500 ステップ繰り返す。
   (a) タンパク質の合成：胚の各細胞において if-then ルールの集合が評価され、タンパク質が合成される。
   (b) タンパク質の拡散：肺は細胞の１次元配列であり、各細胞は各遺伝子調節タンパク質の１／４づつを両隣の細胞に渡す。
   (c) タンパク質の削除：各細胞で、各タンパク質の量を１単位削除する。

今回行うシミュレーションは、初期値として母系効果遺伝子である bicoid タンパク質および nanos タンパク質の濃度勾配が与えられる。このシミュレーションの目的は、与えられた初期値からルールを繰り返し実行することで eve 遺伝子の濃度分布を 4 本の縞模様とすることである。適用する if-then ルール（文献[31]）の一部を次に示す。

(IF (IsConcentrationBetween 1 5.776688 33.021667) protein6)
(IF (IsConcentrationBetween 1 10.724397 6.103053) protein5)
(IF (IsConcentrationBetween 0 10.794619 9.350450) protein8)
(IF (IsConcentrationBetween 0 10.794619 10.025200) protein8)
(IF (IsConcentrationBetween 1 5.307470 37.478039) protein6)

例えば、一番目のルールは「タンパク質 1 の濃度が 5.776688 と 33.021667 の間であればタンパク質 6 を 1 単位合成せよ」と解釈する。このルールをペトリネットで記述したものを図 6.5 に示す。

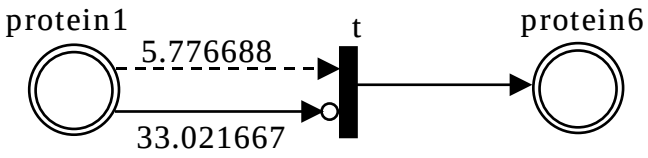


図 6.5　if-then ルールのペトリネットによる記述

これは、タンパク質 1 の濃度が「test_arc」の重み 5.776688 以上で、かつ、「inhibitor」の重み 33.021667 以上でない（33.021667 より低い）ならばトランジション t が発火しタンパク質 6 が 1 増加する。なお、この処理において入力側プレースである protein1 の値に変化はない。

図 6.6 は、先にその一部を示した今回行うシミュレーションの if-then ルールのすべてと、削除ステージをペトリネットで記述したものである。この図では、見やすいように if-then ルールの条件部分を点線矢印で表している。出力のないトランジションが削除ステージを示している。このトランジションは if-then ルールで使用するトランジションより優先度を下げており、合成ステージと混同されることはない。

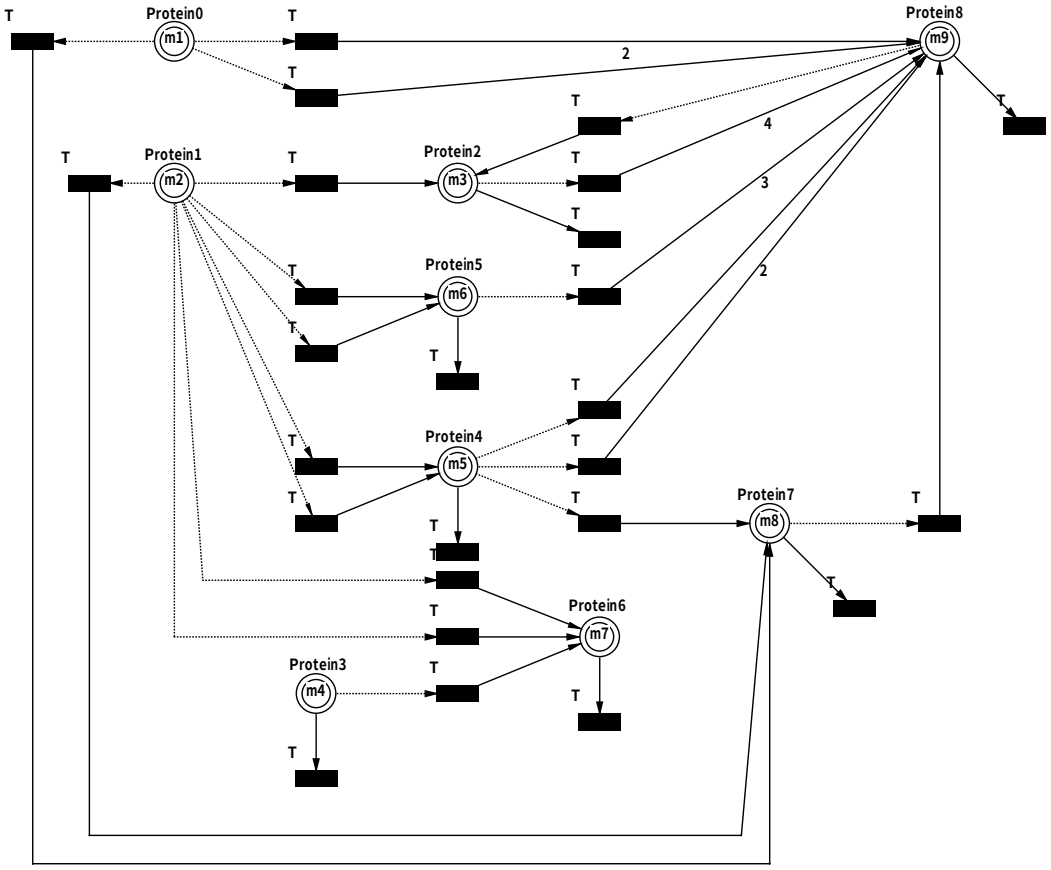


図 6.6　ショウジョウバエの胚発生モデルのペトリネットによる記述

6.5　シミュレーション

ここでは前項で示したモデルのシミュレーションを行う。図 6.7 は初期値として与えられた母系効果遺伝子である bicoid タンパク質および nanos タンパク質の濃度勾配を示したものである。横軸は胚の前後軸を示し、縦軸はタンパク質濃度を示している。この図からは 140 個の細胞において、bicoid タンパク質の濃度の分布が前から後ろへ緩やかな勾配を描き、nanos タンパク質の濃度の分布が後ろから前へ緩やかな勾配を描いている様子が見て取れる。

図 6.8 は与えられた初期値を元にシミュレーションを行い、その 500 ステップ目の結果を示してたものである。この図において protein8 が eve 遺伝子に相当するものである。この図から eve 遺伝子が飛び飛びで 4 ヶ所発現している様子が見て取れる。

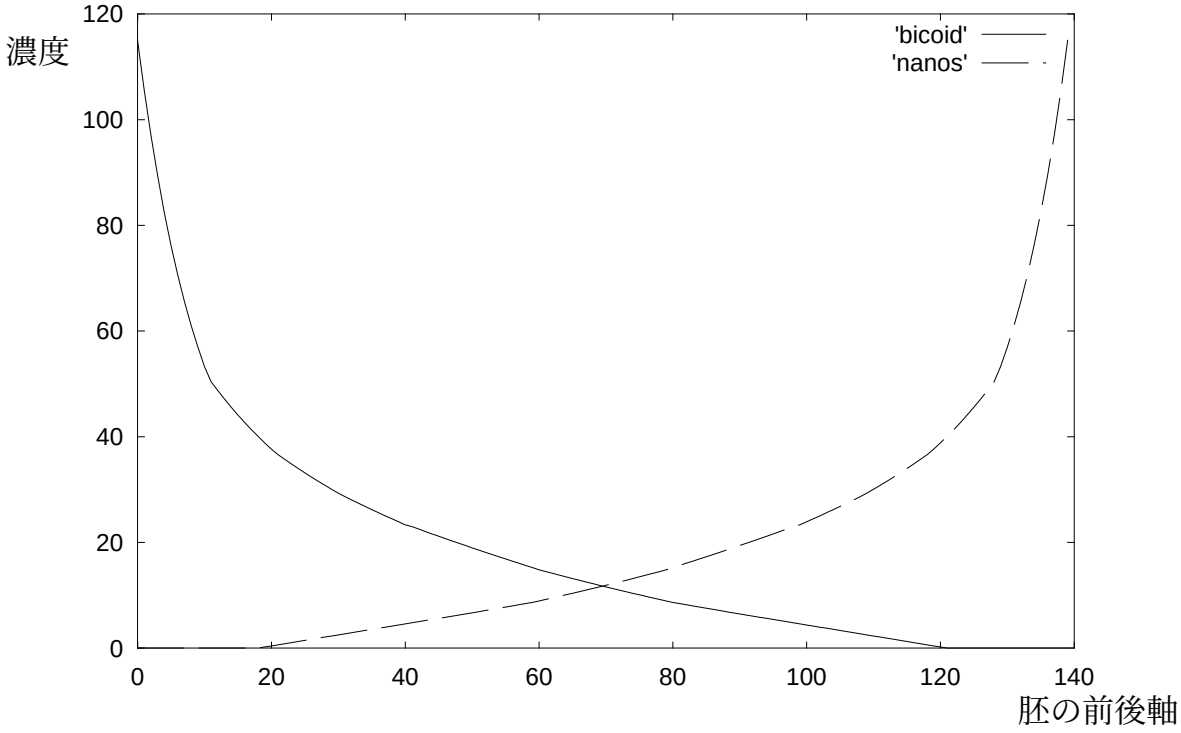


図6.7　初期値として与えられる遺伝子調節タンパク質bicoidとnanosの分布
（文献[31]より）

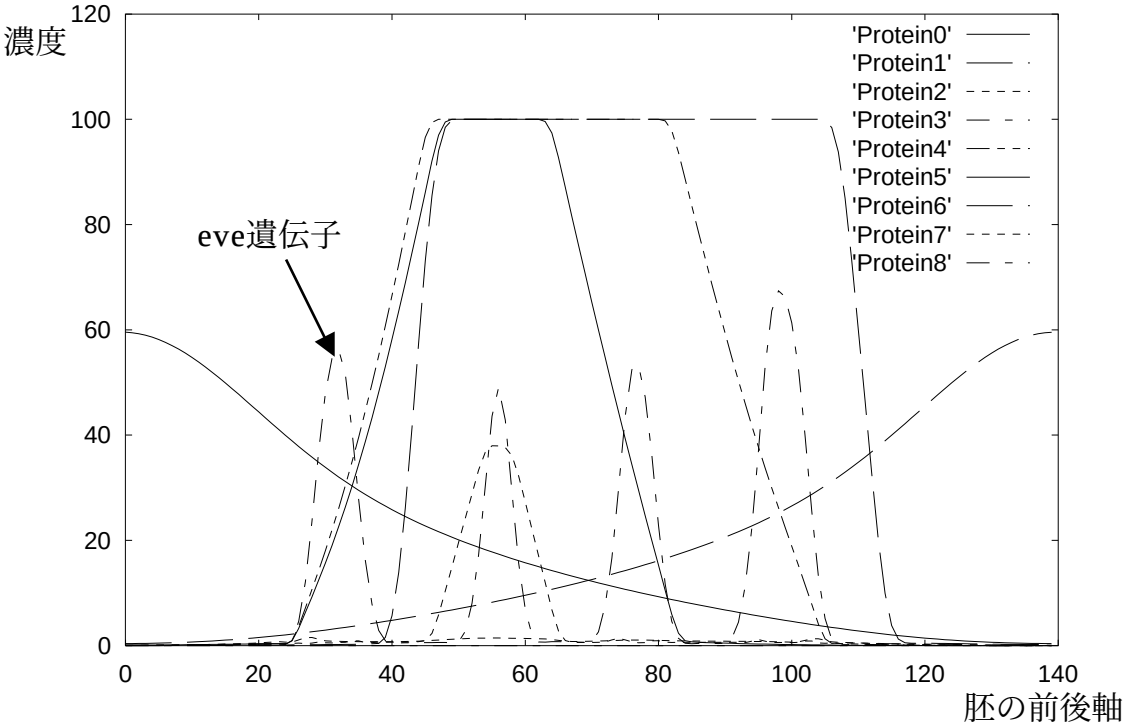


図6.8　500ステップ目の各遺伝子調節タンパク質の分布

これより、このシミュレーションの目的である eve 遺伝子の発現状態に 4 本の縞模様を描かせることに成功したことがわかる。

次に、このモデルが位置シグナル強度のわずかな差異によってパターン形成が阻害されないかを検証した。この検証は初期値として与えられる bicoid および nanos の濃度分布を変化させ、eve 遺伝子の縞模様が何本でるかを確認することで行った。

bicoid タンパク質および nanos タンパク質の初期分布をともに変化させた場合(a)、bicoid タンパク質の初期分布を変化させた場合(b)、nanos タンパク質の初期分布を変化させた場合(c)について表 6.1 に示す。

表 6.1　初期分布を変化させた結果

(初期値×(100%＋変化の割合)を新たな初期値としてシミュレーションを行った。)

(a) bicoidとnanosをともに変化させた場合

変化の割合	縞模様の本数
15%	3本
10%	4本
5%	4本
2.5%	4本
0%	4本
-2.5%	4本
-5%	3本

(b) bicoidを変化させた場合

変化の割合	縞模様の本数
25%	3本
20%	4本
15%	4本
10%	4本
5%	4本
2.5%	4本
0%	4本
-2.5%	4本
-5%	4本
-10%	4本
-15%	3本

(c) nanosを変化させた場合

変化の割合	縞模様の本数
25%	3本
20%	4本
15%	4本
10%	4本
5%	4本
2.5%	4本
0%	4本
-2.5%	4本
-5%	3本

表 6.1 に示されているように、bicoid タンパク質および nanos タンパク質の初期分布をともに 10%から-2.5%変化させた場合でも eve 遺伝子は 4 本縞を形成した。さらに bicoid タンパク質の初期分布を変化させた場合は 20%から-10%、nanos タンパク質の初期分布を変化させた場合は 20%から-2.5%の範囲内で eve 遺伝子が 4 本縞を形成する事を確認した。このことから、このモデルは位置シグナル強度のわずかな差異によってパターン形成が阻害されない頑健なパターン形成の仕組みを実現していることが分かる。

6.6 結論

この章では、開発したハイブリッドペトリネットシステムに対してさらに位置情報を扱えるような拡張を施した。

このシステムを用いてショウジョウバエの胚発生のシミュレーションを行った。モデルとしては、文献[31]のモデルを参考にペトリネットで記述したものを用いた。その結果、母系効果遺伝子の bicoid および nanos の初期分布から、遺伝子制御ネットワークを介して、ペア・ルール遺伝子の 1 つである even-skipped 遺伝子の発現状態に 4 本の縞模様を描かせることができた。また、bicoid と nanos の初期分布を変化させた際の eve 遺伝子の発現を検証したところ、このモデルは位置シグナル強度のわずかな差異によってパターン形成が阻害されない頑健なパターン形成の仕組みを実現していることが確認できた。

第 7 章
結 論

7.1 結論

本研究では、生体分子ネットワークに適用可能なペトリネットシステムを構築した。このペトリネットシステムは、トークンおよびアークの重みとして離散値と実数値の両方を取ることが可能なハイブリッドペトリネットシステムであり、また、トークンの移動に際して、一般的な化学反応や、酵素反応に見られるように入力側の量や濃度などによって発火速度を変化させられるように拡張したものである。さらに、空間的な要素を含む系のシミュレーションを行うための拡張を施した、位置情報付きハイブリッドペトリネットシステムである。

構築したシステムの挙動をいくつかの例題を解くことで検証した。その後、生体分子ネットワークのひとつである細胞周期調節系についてのシミュレーションを行い、最後に、空間的な要素を含むショウジョウバエの胚発生におけるパターン形成についてのシミュレーションを行った。以下にこれらの結果から判明したことを示す。

- 例題として行ったいくつかの数値計算において、既存のツールと比較して本システムの精度や安定性といった面での優位性を確認できた。

- 細胞周期調節系のシミュレーションにおいては、細胞の分裂が見られる M 期を促進する MPF の周期的な活性が見られ、細胞周期をコンピュータ上に再現することができた。また、MPF の活性に関わる Cdc25B フォスファターゼの発現量を

調節することで、MPF の活性周期を変化させることができた。これは、生物学的に確認されている事実と一致する。

- 胚発生のシミュレーションにおいては母系効果遺伝子の bicoid と nanos の初期分布から、複数の遺伝子によって構成される遺伝子制御ネットワークを介して、ペア・ルール遺伝子の 1 つである even-skipped 遺伝子による縞模様の形成を確認できた。

- 細胞周期のシミュレーションは酵素反応速度論を使用した数値計算モデルであり、胚発生のシミュレーションは if-then ルールの集合から構成され、さらにはセルオートマトンのような振る舞いも行うモデルである。本システムはこのように性質の違うモデルでさえも統一的に扱えることを示し、その有効性を確認できた。

7.2 今後の展開

本システムで改良および追加を行う必要があると考えられる点を以下に示す。

- 数値計算部分の改良

  5 章で行った細胞周期のシミュレーションでは、3000 ステップの計算を行うのに 30 分程度の時間がかかっている。これには 2 つの問題が関わっていると考えられる。1 つは、速度式や数値の読み込みがあるごとにそれらをメンバ変数として持つオブジェクトが呼び出され、その後でそれぞれのメンバ変数（速度式や数値）が読み込まれるためだと考えられる。この問題は、ポインタなどを使いメンバ変数に直接アクセスできるようにすれば飛躍的に速くなるがオブジェクト指向の考え方からは外れてしまう。もう 1 つが、数値計算法の問題である。本システムでは常微分方程式の解法としてルンゲ･クッタ法を用いているが、この手法は（刻み幅にもよるが）精度は高いが速度は遅い。これを他の手法（適応刻み幅制御ルンゲ・クッタ法など）に変更することで高速化を図る。

- GUI の作成

  ペトリネットの特徴として、システムとその部品の相互関係を視覚的に捉えることができるというのがあげられるが、本システムにはペトリネットをグラフィカルに作成するエディタのようなものがない。そこで、このエディタの作成が必要だと考えられる。

謝 辞

本研究を進めるにあたり、終始暖かく御指導をいただきました北陸先端科学技術大学院大学遺伝子知識システム論講座 小長谷 明彦 教授に厚く御礼申し上げます。

また、さまざまな面で御教授いただきました北陸先端科学技術大学院大学遺伝子知識システム論講座 佐藤 賢二 助教授に深く感謝いたします。

北陸先端科学技術大学院大学遺伝子知識システム論講座の助手である Xavier Defago 先生、山本知幸先生には論文に関する数々のご助言をいただき深く感謝いたします。

北陸先端科学技術大学院大学複雑系解析論講座 中森 義輝 教授には副テーマで熱心に御指導をいただき深く感謝いたします。

また、日頃よりお世話になりました当研究室の皆様に心より感謝いたします。

参 考 文 献

[1] 奥川峻史, “ペトリネットの基礎”, 共立出版, 1995.

[2] 村田忠夫, ”ペトリネットの解析と応用”, 近代科学社, 92.

[3] M. Hack, “Decidability Questions for Petri Nets”, Ph.D. dissertation, Department of Electrical Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1975

[4] 潮俊光, “ハイブリッドペトリネット”, 計測と制御, 第38巻, 第3号, pp.169-175, 1999.

[5] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Frannery, “NUMERICAL RECIPES in C : the art of scientific computing”, Cambridge University Press, 1997.

[7] R. David and H. Alla, “Continuous Petri Nets”, 8th Eulopean Workshop on Application and theory of Petri Nets, pp.275-294, 1987

[8] J. L. Bail, H. Alla, and R. David, “Hybrid Petri Nets”, proc. European Control Canference, pp.1472-1477, 1991.

[9] http://GenomicObject.Net/

[10] http://www.systemtechnik.tu-ilmenau.de/~drath/visual_E.htm

[11] R. Drath, “Hybrid object net : an object oriented concept for modeling complex hybrid system”, proc. 3rd International Conference on Automation of Mixed Processes, pp437-442, 1998.

[12] 吉岡隆, 小谷秀示, 小長谷明彦, “高等動物体細胞周期のペトリネットによるモデル化について”, Computer Todey, No.108, pp26-33,2002.

[13] H. Matuno, A. Doi, M. Nagasaki and S. Miyano, “Hybrid Petri net representation of gene regulatory network”, Proc. Pacific Symposium on

Biocomputing 5, pp.338-349,2000.

[14] P. J. E. Goss and J. Peccoud, “quantitative modeling of stochastic systems in molecular biology by stochastic Petri nets”, proc. Natl. Acad. Sci. USA, Vol..95, pp6750-6755, 1998.

[15] B. Alberts, D. Bray, J. Lewis, M. Raff, K. Roberts and J. D. Watoson, “Molecular Biology of The Cell”, Garland Publishing, Inc., 3rd edition, 1994.(邦訳：”細胞の分子生物学第３版”, 中村圭子, 藤山秋佐夫, 松原謙一監訳, 教育社, 1995.

[16] V. N. Reddy M. L. Mavrovouniotis and M. N. Liebman, “Petri Net Representations in Metabolic Pathways”, proc. ISMB-93, MIT Press, 1993.

[17] B. Novak and J. J. Tyson, “Numerical analysis of a comprehensive model of M-phase control in Xenopus oocyte extracts and intact embryos”, J. Cell Sci., 106, p.1153-1168, 1993.

[18] D. Gonze and A. Goldbeter, “A Model for a Network of Phosphorylation-dephosphorylation Cycles Displaying the Dynamics of Dominoes and Clocks”, J. theor. Biol., 210, pp.167-186, 2001.

[19] J. J. Tyson and B. Novak, “Regulation of the Eukaryotic Cell Cycle : Molecular Antagonism, Hysteresis, and Irreversible Transitions”, J. theor. Biol., 210, pp.249-263, 2001.

[20] M. Kaern and A. Hunding, “Dynamins of the Cell Cycle Engine : Cdk2-kinase and the Transition into Mitosis”, J. theor. Biol., 193, pp.47-57,1998.

[21] I. Takemasa, et al., “Overexpression of CDC25B Phosphatase as a Novel Marker of Poor Prognosis of Human Colorectal Carcinoma”, Cancer Reserch, 60, pp.3043-3050,2000.

[22] M. Chen, et al., “Absence of Apparent Phenotype in Mice Lacking Cdc25C Protein Phosphatase”, Molecilar and Cellular Bioloy, Vol..21, No.12, pp.3853-3861, 2001.

[23] C. A. Petri, “Kommunukation mit Automaten”, PhD dissertation, University of Bonn, Bonn, West Germany, 1962.(In German) (“Communication with Automata”, G.F. Greene ,Jr. transrator, Supplement to Tech. Rep. RADC-65-

337, 1, Rome Air Development Center, CriffissAir Force Base, N. Y., 1965.)
[24] 大西正健, “酵素反応速度論実験入門：生物化学実験法 21”, 学会出版センター, 1987.
[25] 矢嶋信男, “常微分方程式：理工系の数学入門コース”, 岩波書店, 1989.
[26] S. F. Gilbert, “The genetics of axis specificant in Drosophila”, Developmental Biology, Chapter 14, SinauerAssociates Inc., 1997.
[27] J. Reinitz and D. H. Sharp, “Mechanism of eve stripe formation”, Mechanisms of Development, Vol..49, pp.133-158, 1995.
[28] 濱橋秀互, 京田耕司, “ショウジョウバエのシミュレーション”, システムバイオロジーの展望：生物学の新しいアプローチ, pp.155-164, シュプリンガー・フェアラーク東京株式会社, 2001.
[29] 近藤滋, “コンピュータを使って発生を考える”, 細胞工学, Vol..15, No..6, pp.817-824, 1996.
[30] M. Arita, “SIMFLY2: Simulation of a Fly Embryo”, proc. 6th Genome Informatics Workshop, pp.29-38, Universal Academy Press, 1995.
[31] 萩原茂, “胚発生のパターン形成の仕組みの創発に関わる研究”, 修士論文, 知識科学研究科, 北陸先端科学技術大学院大学, 2000.

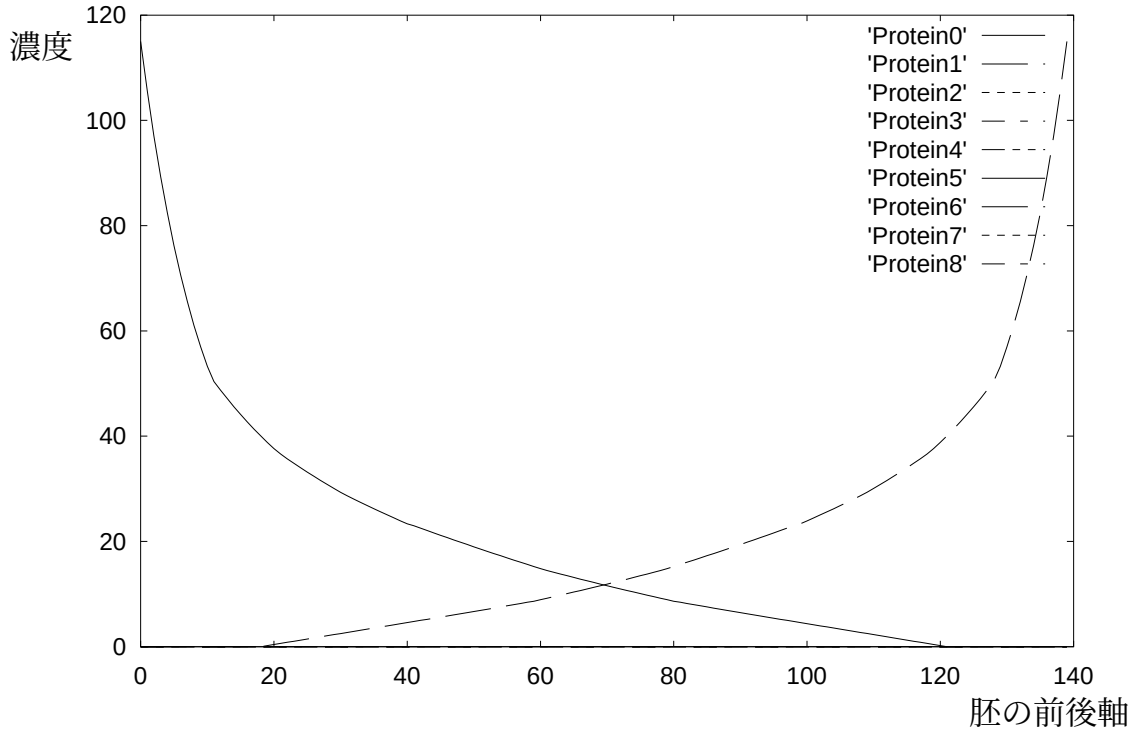
付 録

文献[31]より、第 6 章で行ったショウジョウバエの胚発生におけるパターン形成についてのシミュレーションのモデルとなった if-then ルールの集合を以下に示す。

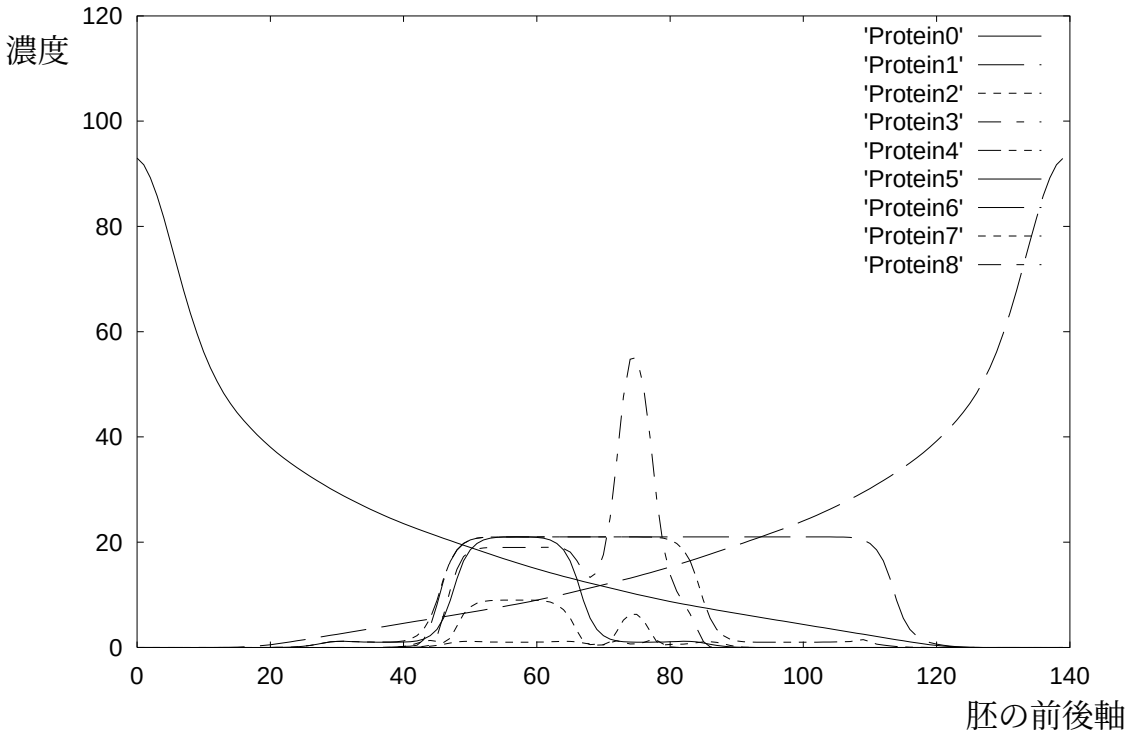
(IF (IsConcentrationBetween 1 5.776688 33.021667) protein6)
(IF (IsConcentrationBetween 1 10.724397 6.103053) protein5)
(IF (IsConcentrationBetween 0 10.794619 9.350450) protein8)
(IF (IsConcentrationBetween 0 10.794619 10.025200) protein8)
(IF (IsConcentrationBetween 1 5.307470 37.478039) protein6)
(IF (IsConcentrationBetween 1 17.059525 2.244593) protein4)
(IF (IsConcentrationBetween 1 5.776688 30.096888) protein4)
(IF (IsConcentrationBetween 1 10.724397 6.103053) protein2)
(IF (IsConcentrationBetween 4 28.167505 36.275078) protein8)
(IF (IsConcentrationBetween 2 37.478039 51.445824) protein8)
(IF (IsConcentrationBetween 7 62.635323 0.242824) protein8)
(IF (IsConcentrationBetween 4 2.447898 10.794619) protein7)
(IF (IsConcentrationBetween 1 17.059525 2.244593) protein5)
(IF (IsConcentrationBetween 0 10.794619 9.350450) protein8)
(IF (IsConcentrationBetween 4 11.277724 34.004467) protein8)
(IF (IsConcentrationBetween 4 11.277724 34.004467) protein8)
(IF (IsConcentrationBetween 0 9.350450 11.277724) protein7)
(IF (IsConcentrationBetween 1 10.724397 6.103053) protein7)
(IF (IsConcentrationBetween 3 32.326157 2.058443) protein6)

(IF (IsConcentrationBetween 8 38.440750 24.766159) protein2)
(IF (IsConcentrationBetween 2 37.478039 51.445824) protein8)
(IF (IsConcentrationBetween 2 37.478039 51.445824) protein8)
(IF (IsConcentrationBetween 5 20.157558 21.375805) protein8)
(IF (IsConcentrationBetween 5 20.157558 21.375805) protein8)
(IF (IsConcentrationBetween 5 20.157558 21.375805) protein8)
(IF (IsConcentrationBetween 2 37.478039 51.445824) protein8)

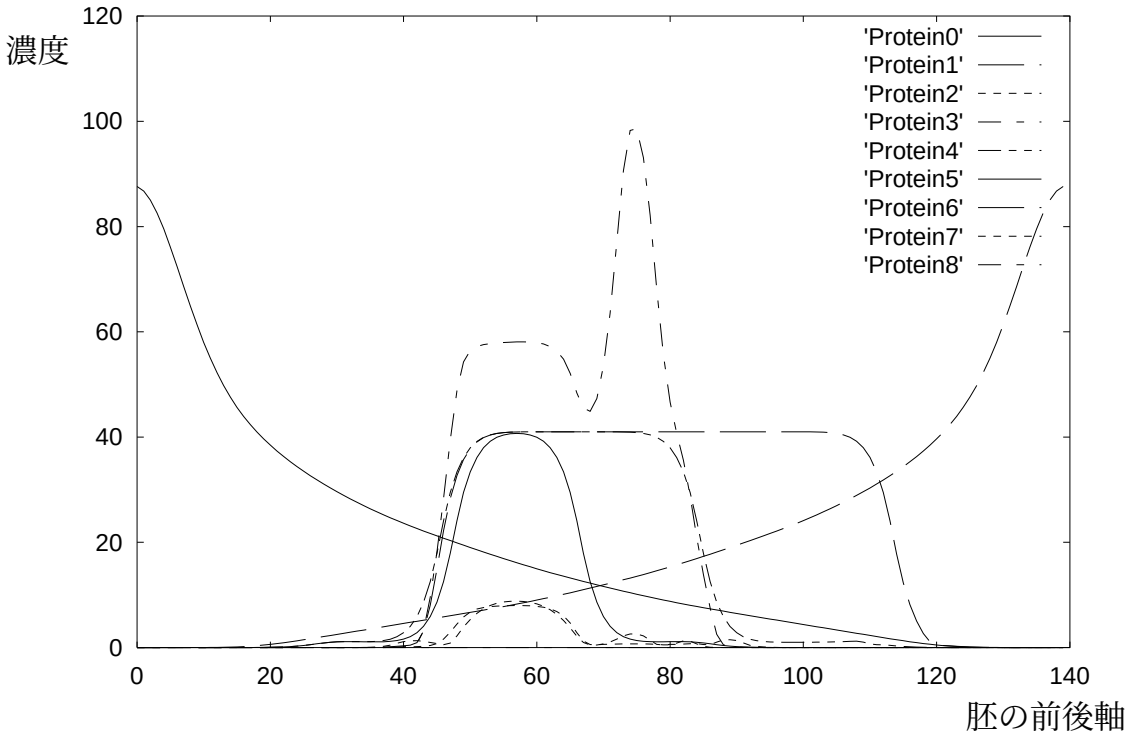
以下では上述のモデルを使ったシミュレーションの結果を、20 ステップごとの遺伝子調節タンパク質の分布図で示す。なお、”protein0”は bicoid、”protein1”は nanos、”protein8”は eve を示している。



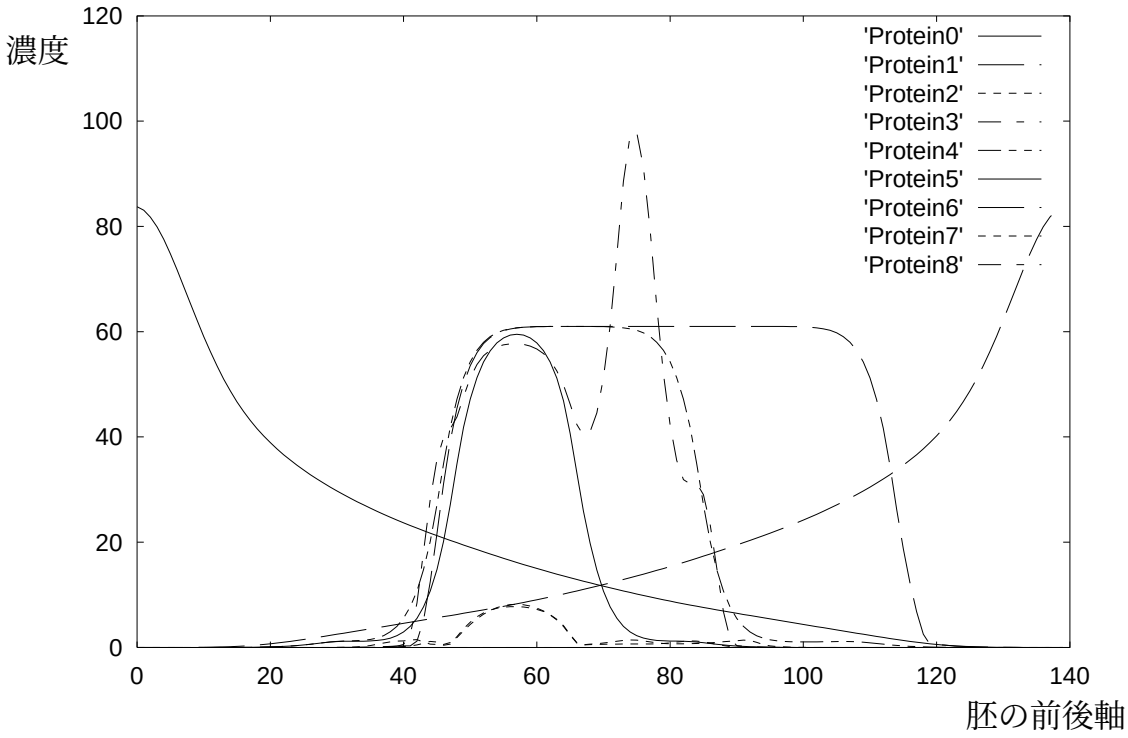
ステップ0 時の各遺伝子調節タンパク質の分布



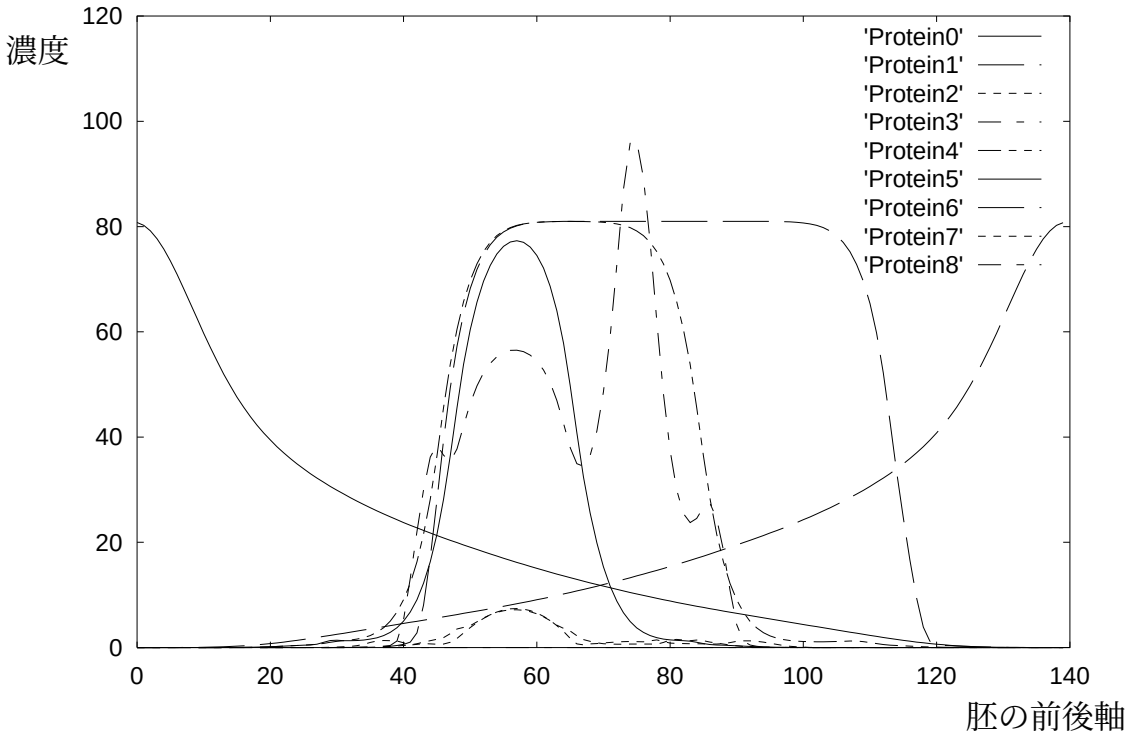
ステップ20 時の各遺伝子調節タンパク質の分布



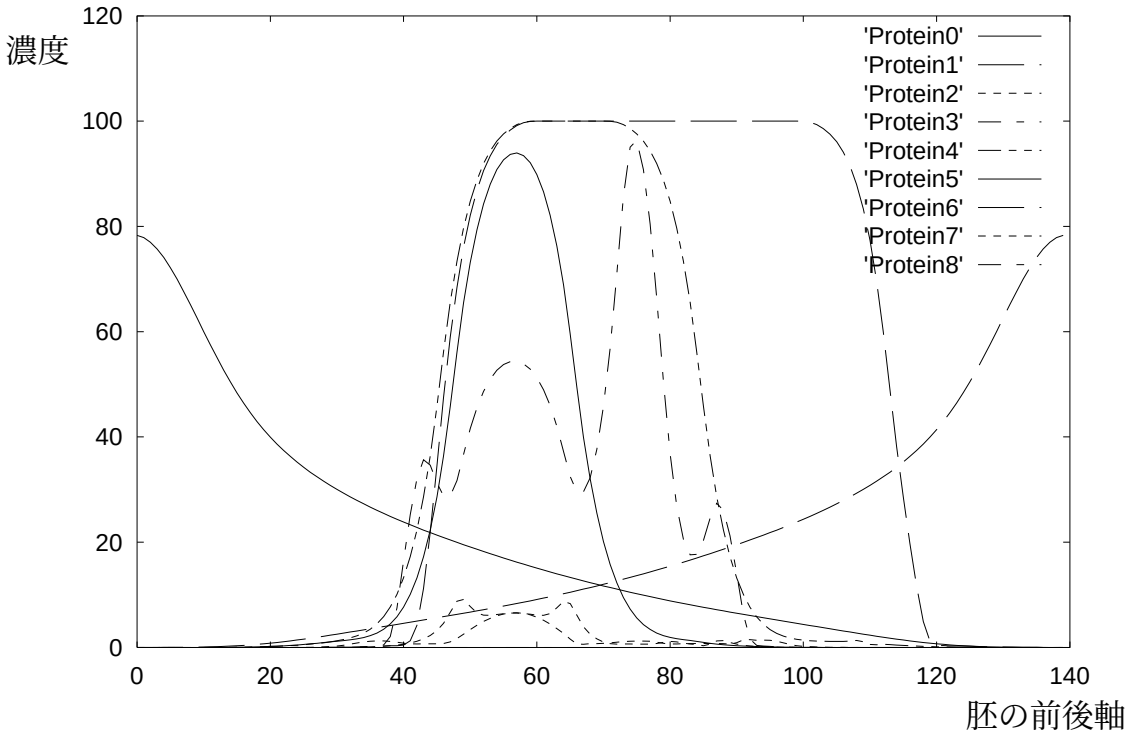
ステップ40 時の各遺伝子調節タンパク質の分布



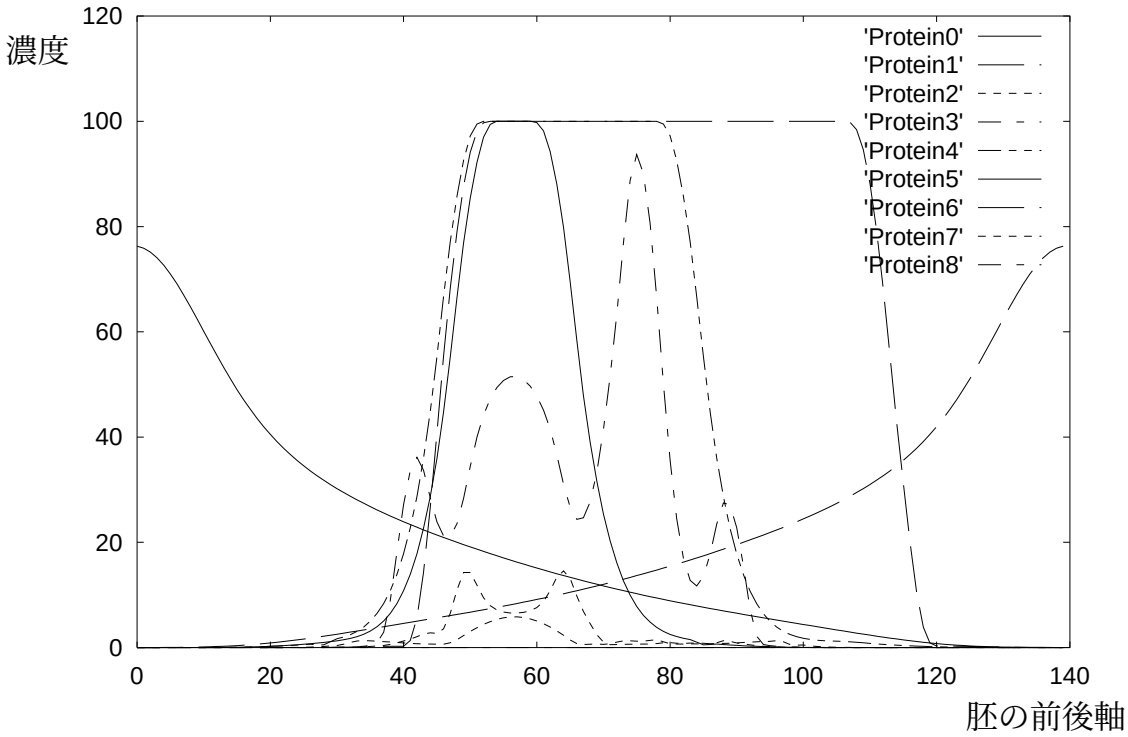
ステップ60 時の各遺伝子調節タンパク質の分布



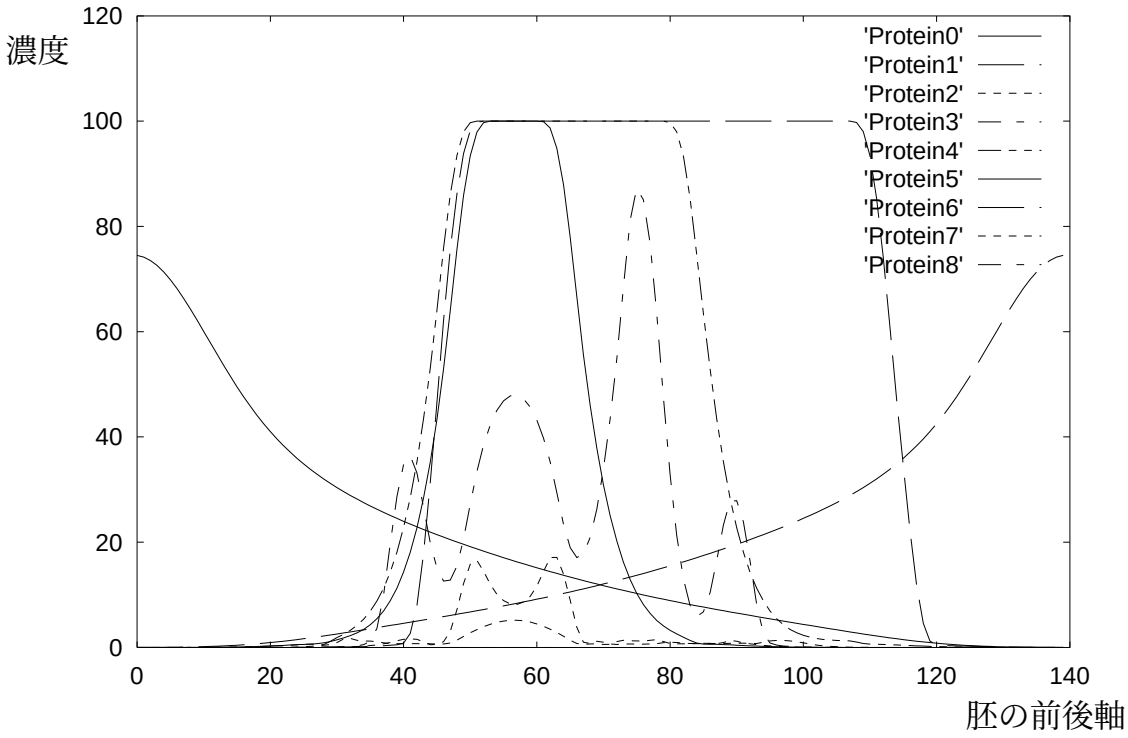
ステップ80 時の各遺伝子調節タンパク質の分布



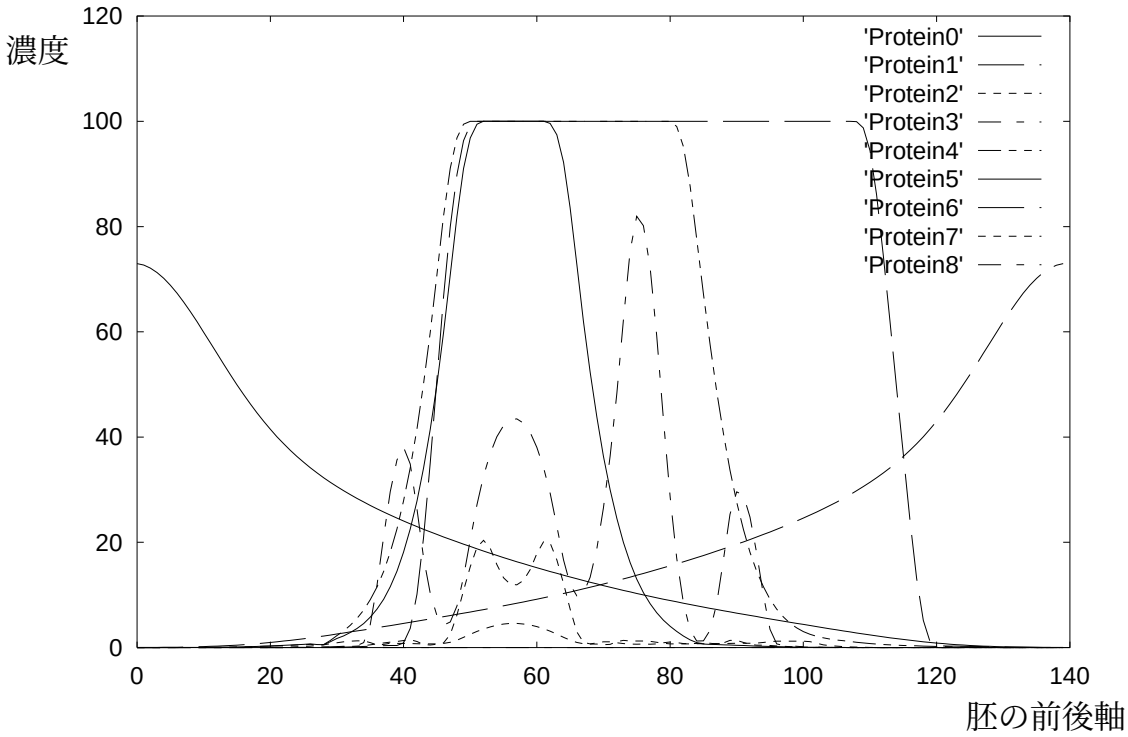
ステップ100 時の各遺伝子調節タンパク質の分布



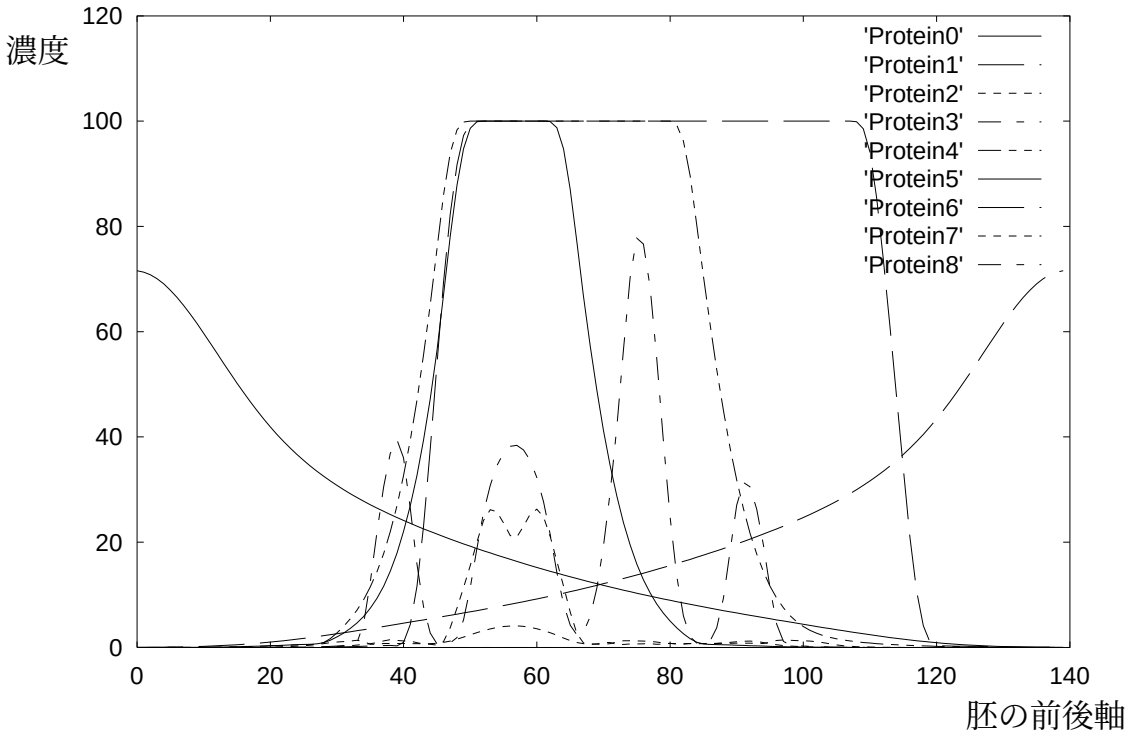
ステップ120 時の各遺伝子調節タンパク質の分布



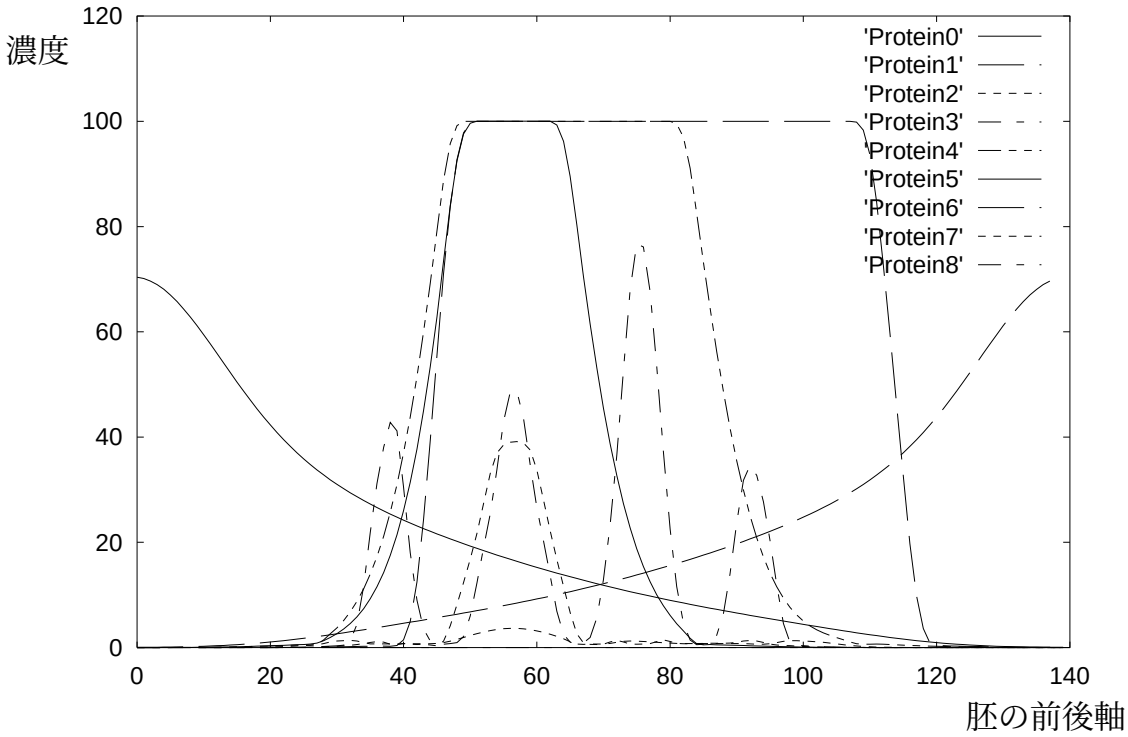
ステップ140 時の各遺伝子調節タンパク質の分布



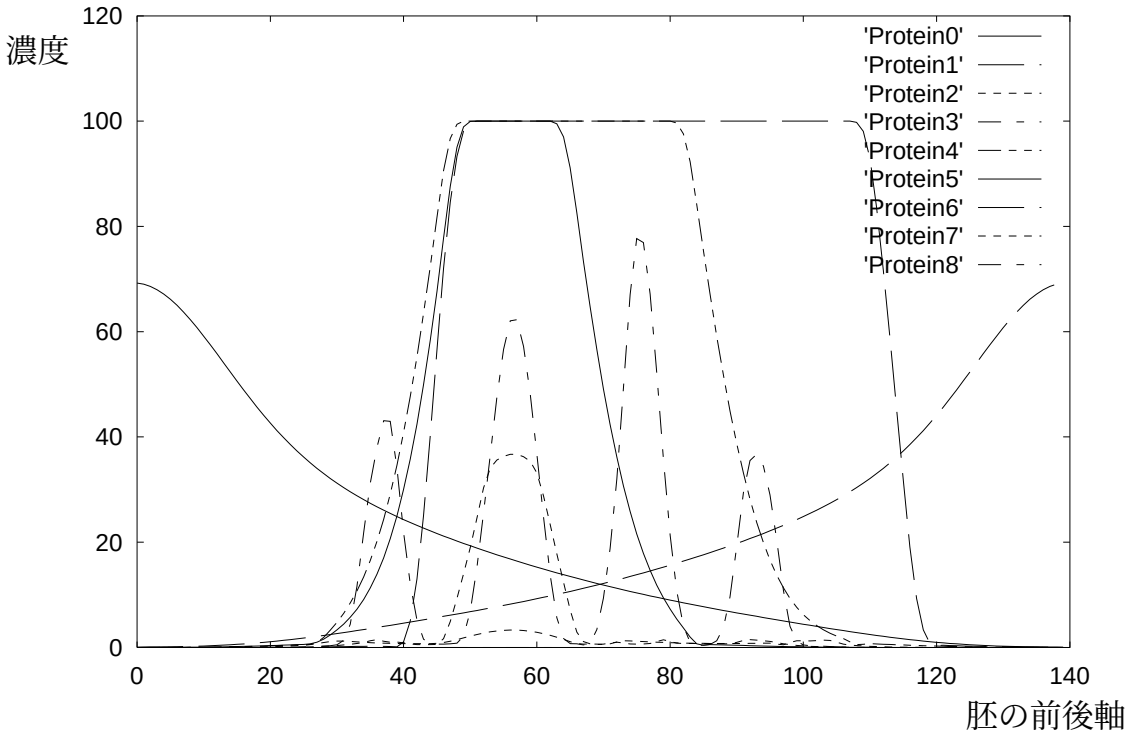
ステップ160 時の各遺伝子調節タンパク質の分布



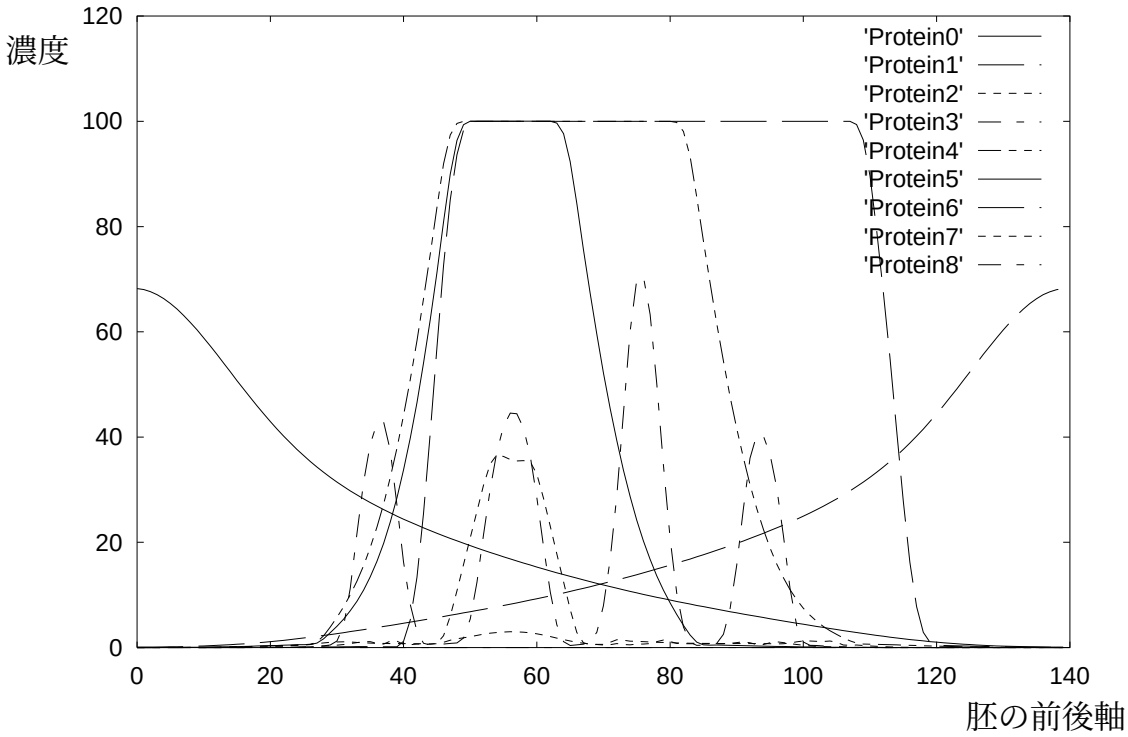
ステップ180 時の各遺伝子調節タンパク質の分布



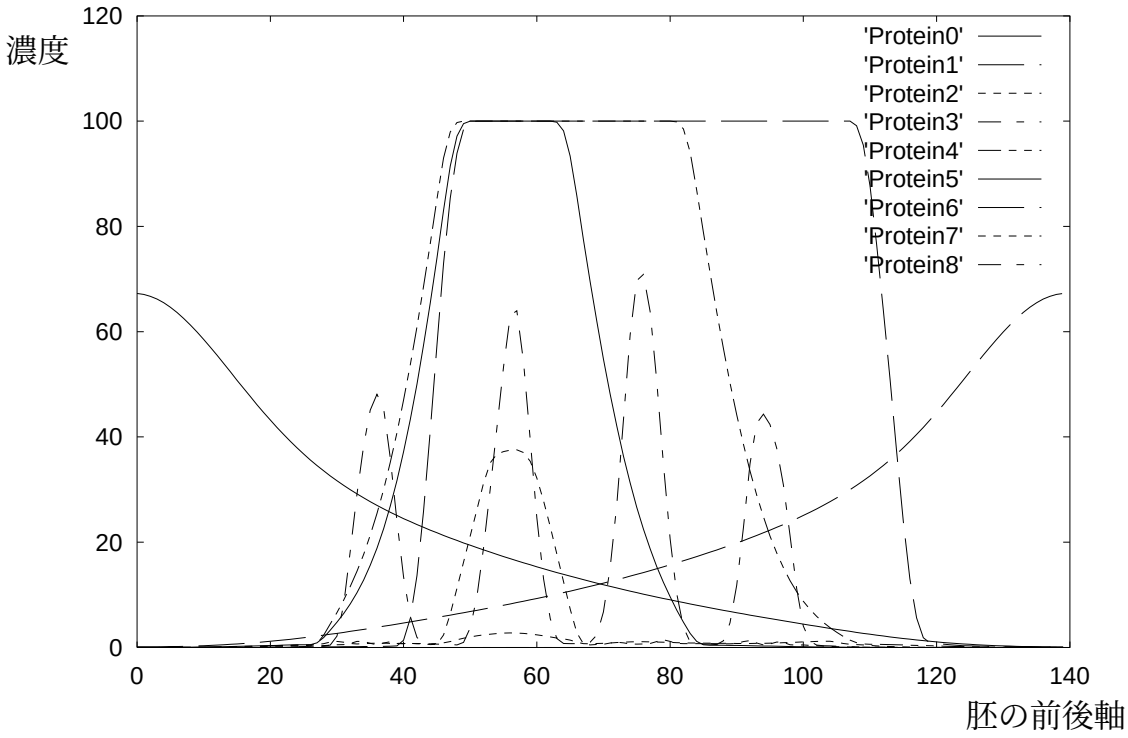
ステップ200 時の各遺伝子調節タンパク質の分布



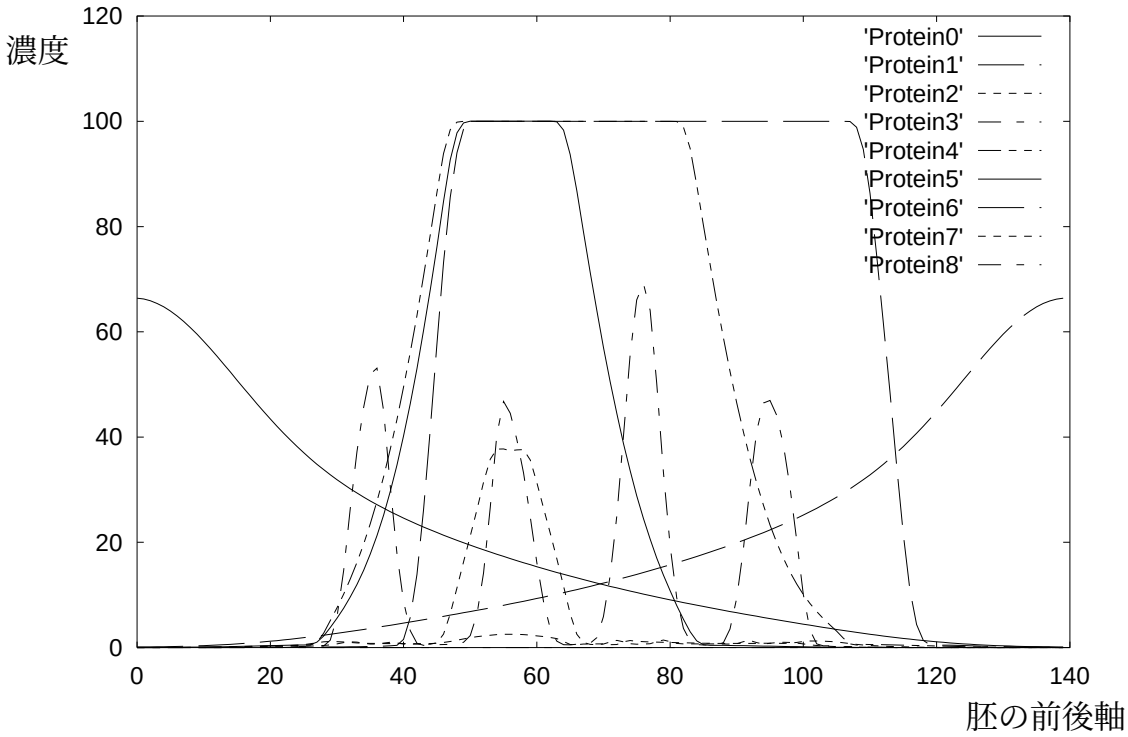
ステップ220 時の各遺伝子調節タンパク質の分布



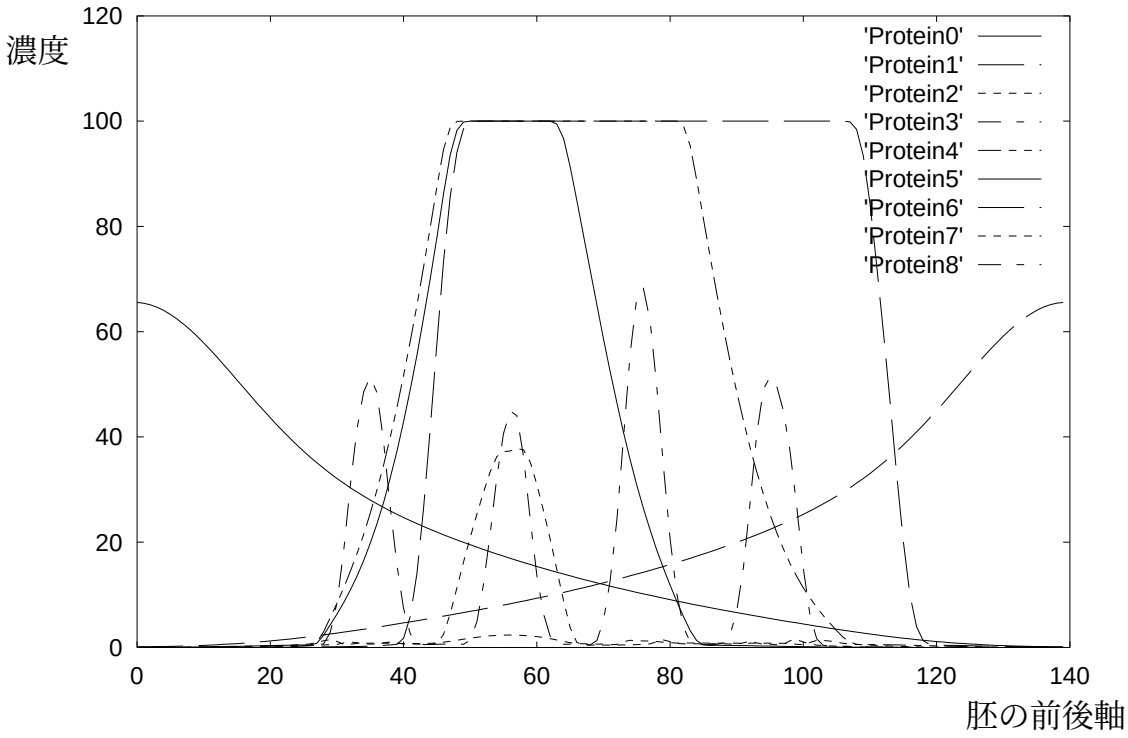
ステップ240 時の各遺伝子調節タンパク質の分布



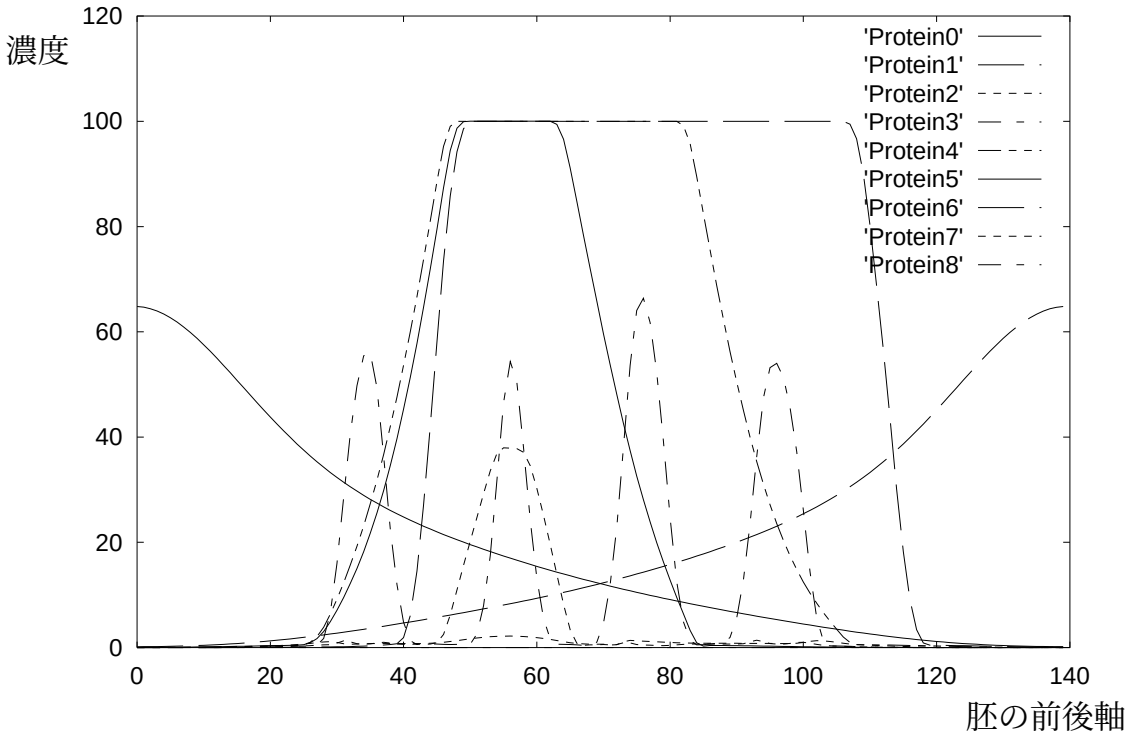
ステップ260 時の各遺伝子調節タンパク質の分布



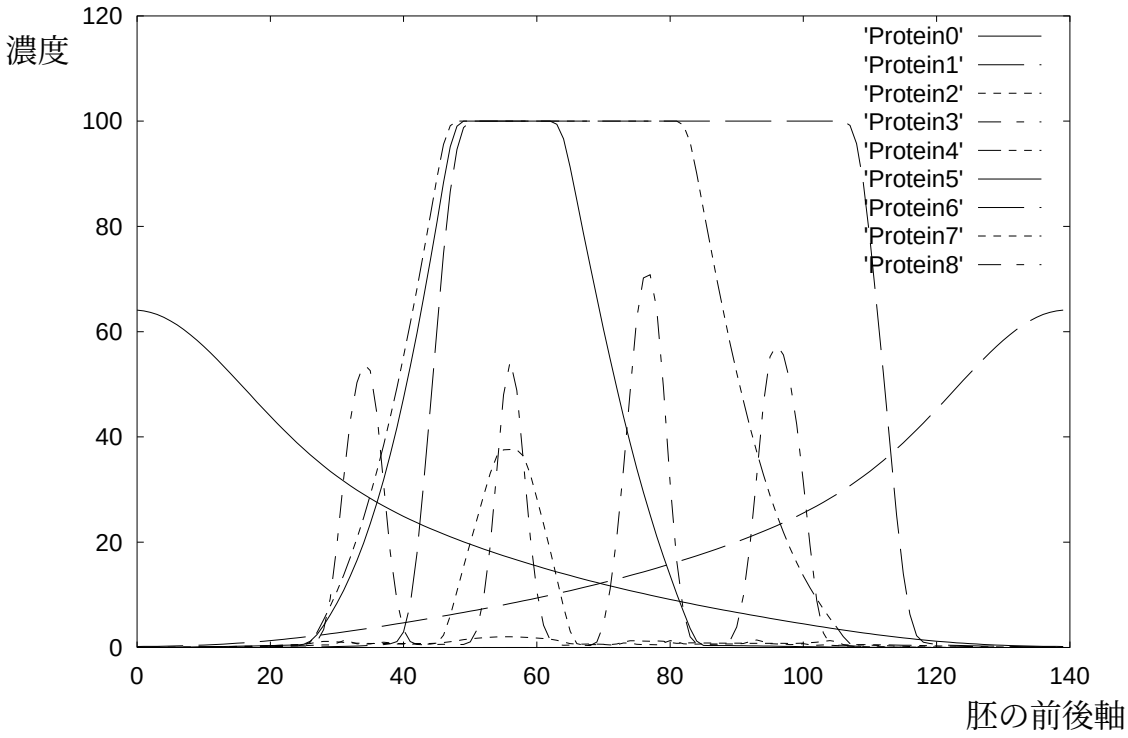
ステップ280 時の各遺伝子調節タンパク質の分布



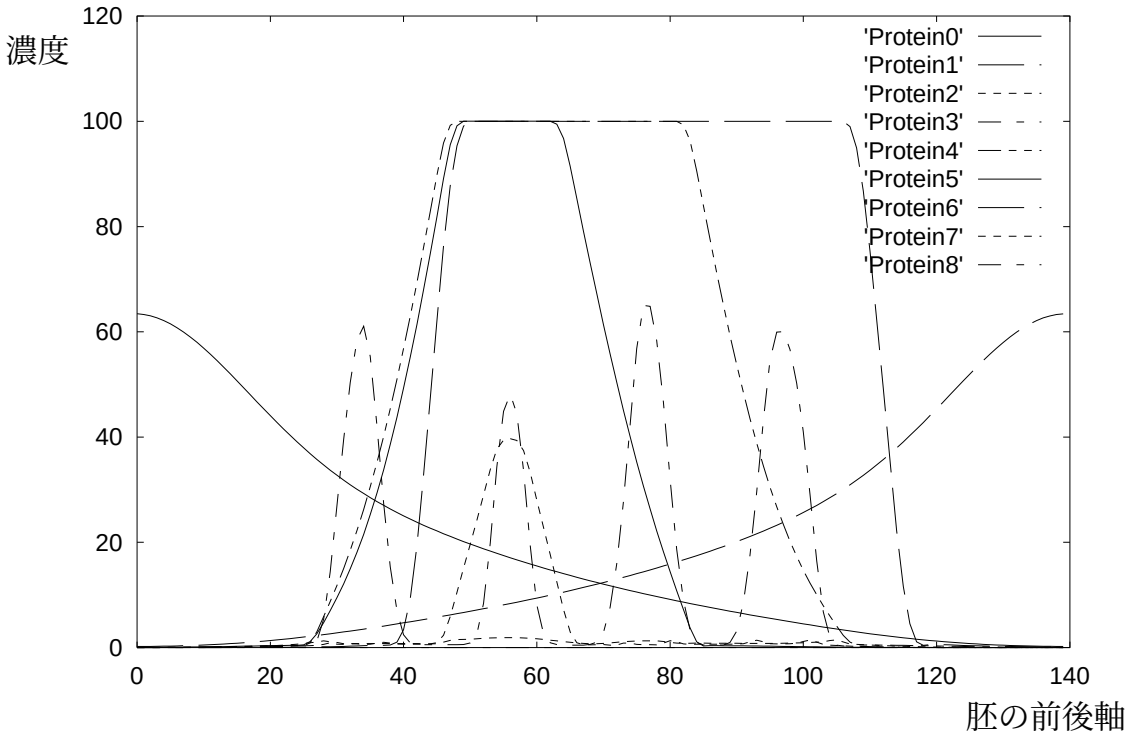
ステップ300 時の各遺伝子調節タンパク質の分布



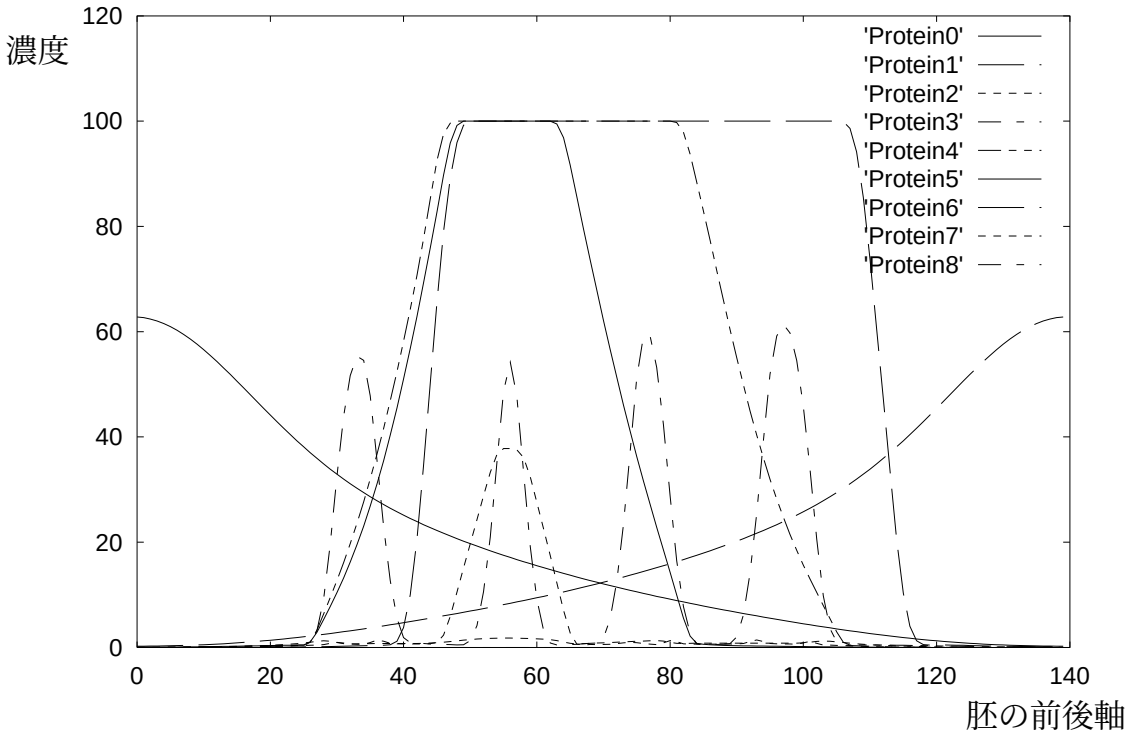
ステップ320 時の各遺伝子調節タンパク質の分布



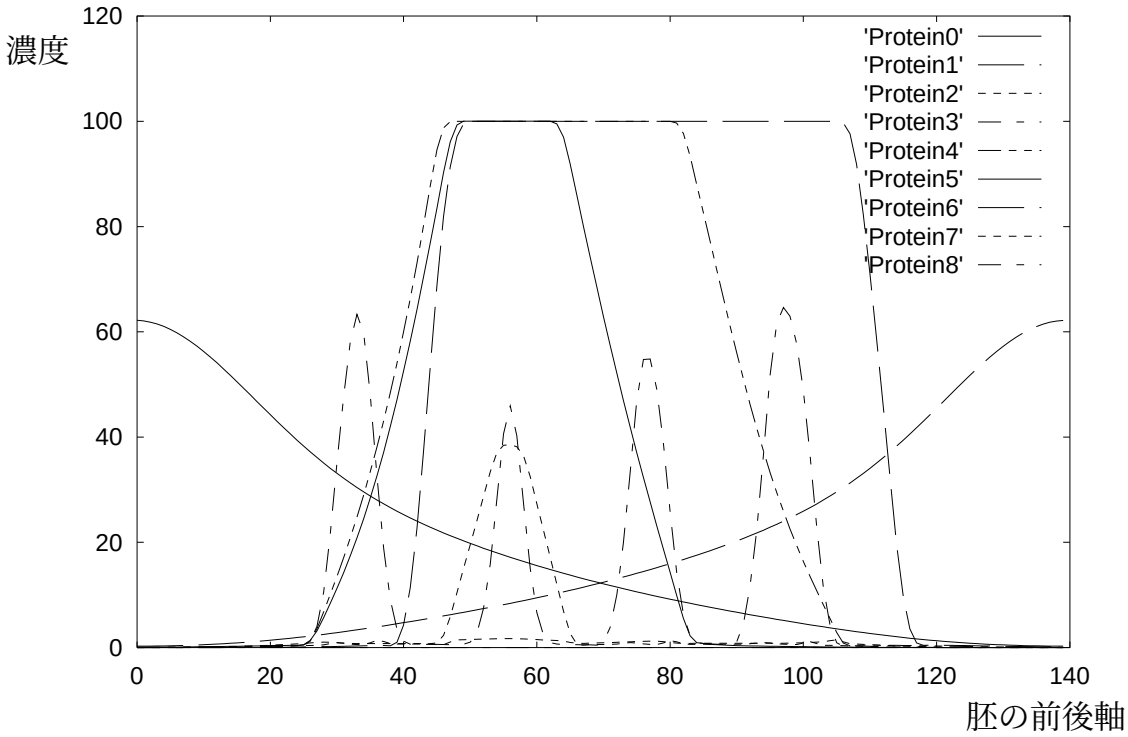
ステップ340 時の各遺伝子調節タンパク質の分布



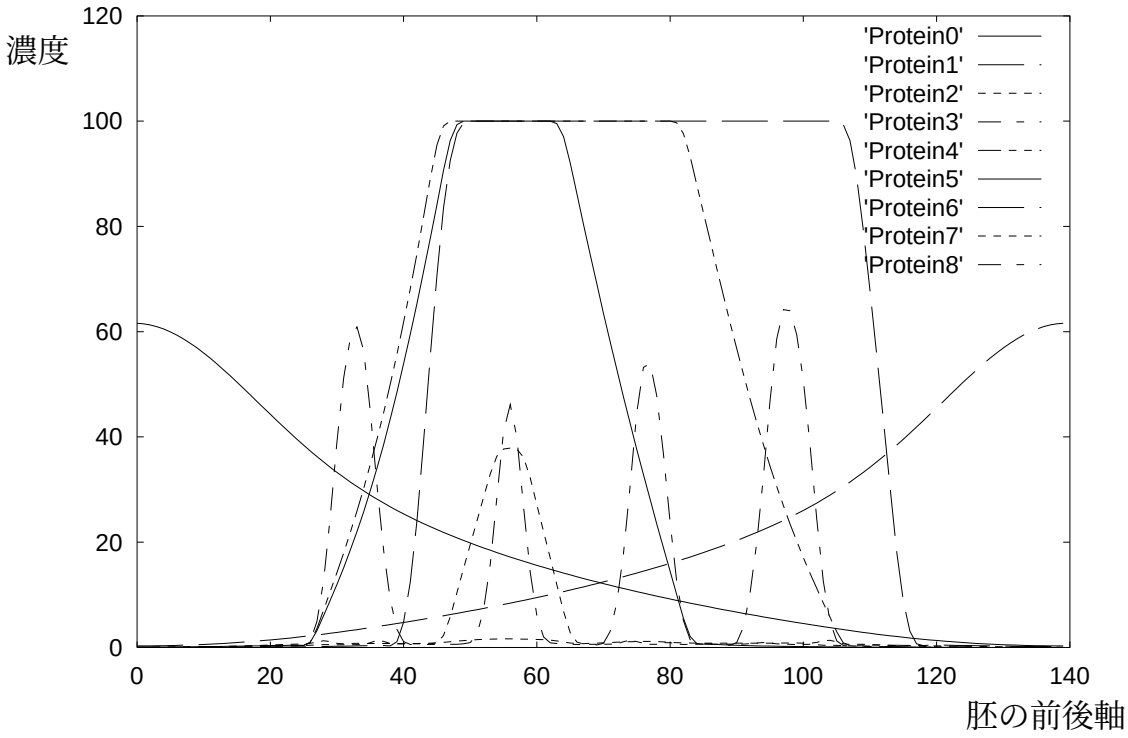
ステップ360 時の各遺伝子調節タンパク質の分布



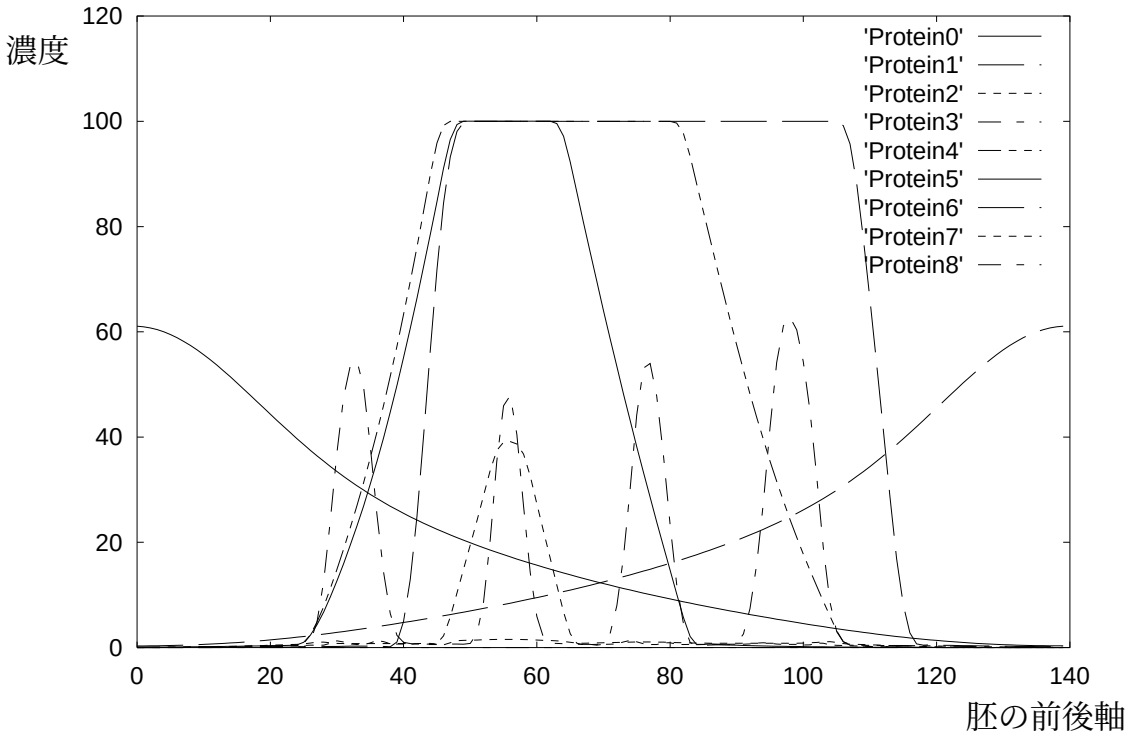
ステップ380 時の各遺伝子調節タンパク質の分布



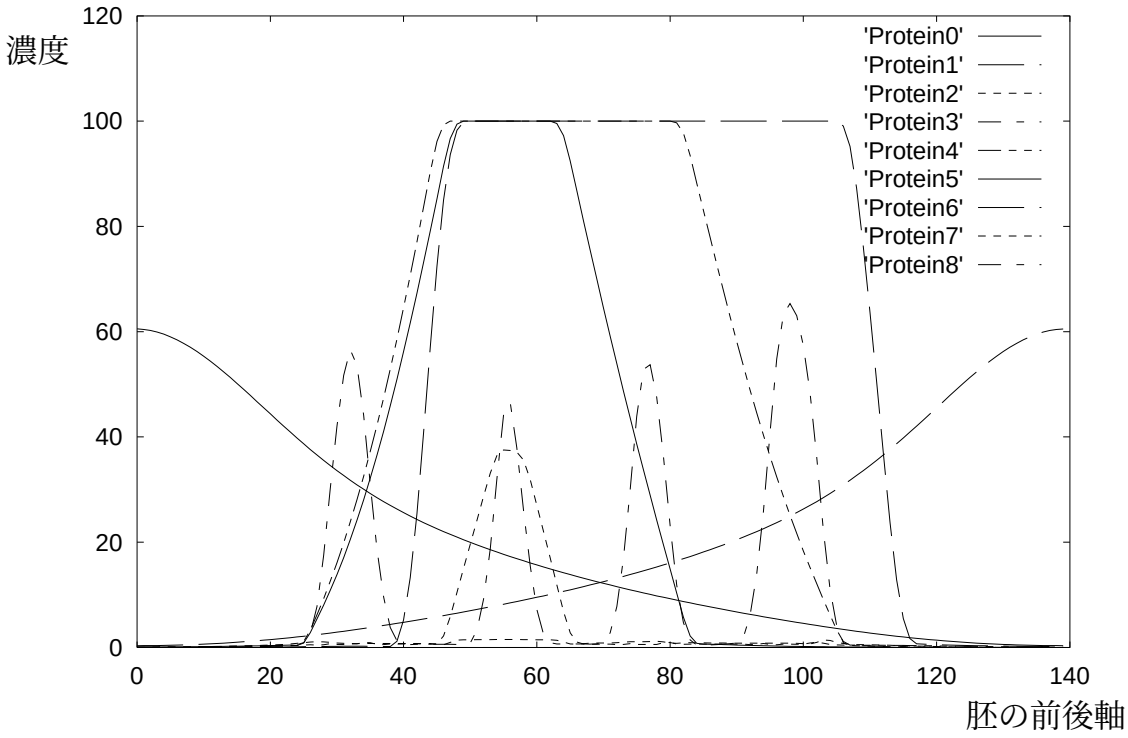
ステップ400 時の各遺伝子調節タンパク質の分布



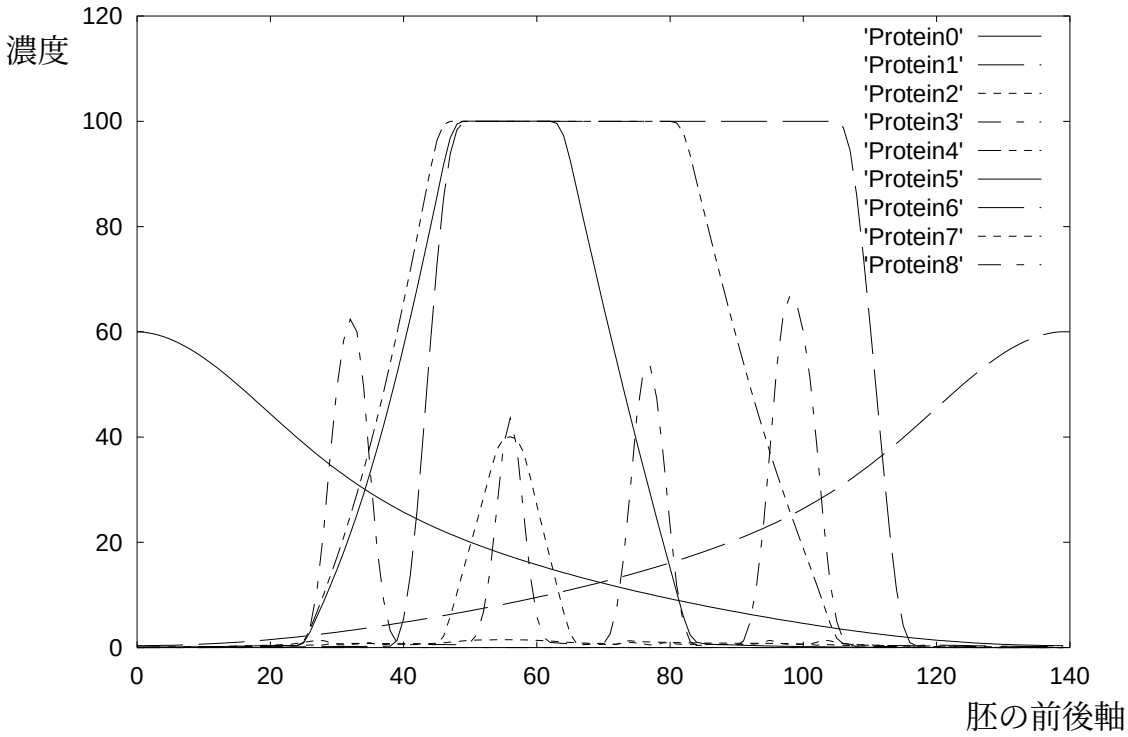
ステップ420 時の各遺伝子調節タンパク質の分布



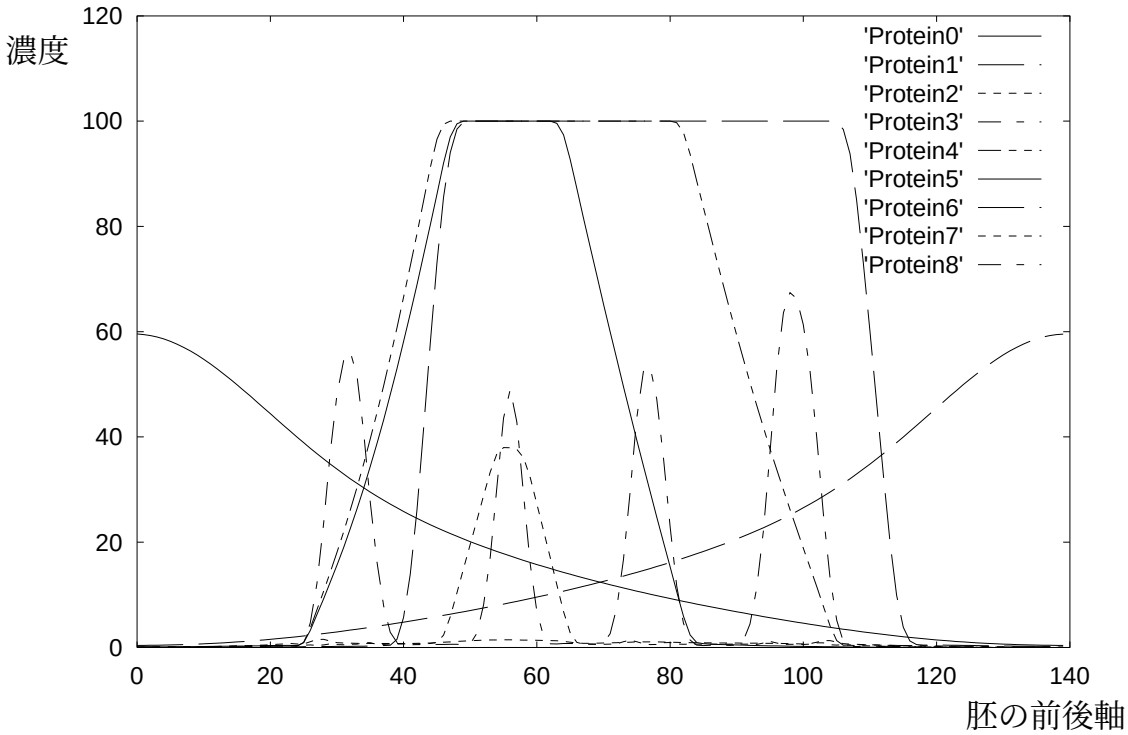
ステップ440 時の各遺伝子調節タンパク質の分布



ステップ460 時の各遺伝子調節タンパク質の分布



ステップ480 時の各遺伝子調節タンパク質の分布



ステップ500 時の各遺伝子調節タンパク質の分布