Java 言語を用いた物理・化学・生物学的現象のシミュレーション

橋口 幸司

2003 年 2 月 22 日

電子情報工学教室

目次

第 **1** 章　序論 **1**

1.1　研究の背景 1

1.2　並列と並行の違い 1

1.3　Java 言語について 2

1.4　マルチスレッド 3

1.5　本研究の目的 4

第 **2** 章　シミュレーション **5**

2.1　イジング模型のシミュレーション 5

2.1.1　概要 5

2.1.2　プログラムの条件 6

2.1.3　プログラムの流れ 6

2.2　酵素と基質の反応のシミュレーション 8

2.2.1　概要 8

2.2.2　プログラムの条件 10

2.2.3　プログラムの流れ 10

2.3　金平糖の生成シミュレーション 12

2.3.1　概要 12

2.3.2　プログラムの条件 12

2.3.3　プログラムの流れ 13

第 **3** 章　結果及び検討 **14**

3.1　イジング模型のシミュレーション 14

3.1.1　実行結果 14

3.1.2　検証 15

3.2　酵素と基質の反応のシミュレーション 17

3.2.1 実行結果 17
3.2.2 検証 19
3.3 金平糖の突起生成のシミュレーション 22
3.3.1 実行結果 22
3.3.2 検証 22

第 4 章 結論と今後の課題 **23**
4.1 結論 23
4.2 今後の課題 23

参考文献 **25**

図目次

1.1 並列（parallel）の並行（concurrent）の違い 2
1.2 シングルスレッドとマルチスレッドの違い 3

2.1 イジング模型 5
2.2 イジング模型シミュレーションのプログラムの流れ 7
2.3 酵素と基質の反応のペトリネットによる記述 9
2.4 酵素と基質の反応のシミュレーションのプログラムの流れ . 11
2.5 金平糖の生成シミュレーションのプログラムの流れ 13

3.1 イジング模型のシミュレーション実行中の様子 14
3.2 イジング模型のシミュレーションのある部分の様子 15
3.3 イジング模型のシミュレーションに現れた3つのパターン . . 16
3.4 酵素と基質の反応のシミュレーション実行中の様子 17
3.5 酵素と基質の反応のシミュレーションの様子 18
3.6 酵素と基質の反応の微分方程式の解 20
3.7 酵素と基質の反応のシミュレーションにおけるそれぞれの要素数の変化 21

第 1 章 序論

1.1 研究の背景

実世界の現象の多くは時間とともに同時並行で状態が変化している。これらの現象をコンピュータ上でシミュレーションするということは昔から行われてきたが、そうしたシミュレーションの多くは逐次型 (シーケンシャル : sequencial) のプログラムで動作しており並行型 (コンカレント : concurrent) なシミュレーションではなかった。そして、ある現象に対してのみのモデル作り、及びシミュレーションが多く行われてきた。このことからシミュレーションと実世界の現象にはずれがあったり、また、設定された仮定でのみ成立するというような場合があった。そのようなシミュレーションでは実際問題との比較は難しいと思われる。また、シミュレートしている現象に対して、現象などの変更、追加、削除があった場合、その現象専用のシミュレーションでは、モジュール、プロセス等の変更に多くの時間がかかってしまう。

このことからより現実に近い形の自然な記述の並行処理を行うシミュレーションが必要とされている。そこで本研究では、汎用性の高い Java 言語上で動くシーケンシャルではないコンカレントなプログラムを用いてシミュレーションを行っていく。

1.2 並列と並行の違い

さて、まず初めに並列処理と並行処理の違いについて簡単に述べておく。並列処理 (parallel processing) とは「物理的に」多数のプロセッサによって実行される処理のこと指し、主に 1 つの大きな処理を分割して複数のプロセッサで計算させることが多い。並行処理 (concurrent processing) とは「論理的に」複数の処理が実行される。よって、プロセッサの数が一個でも複

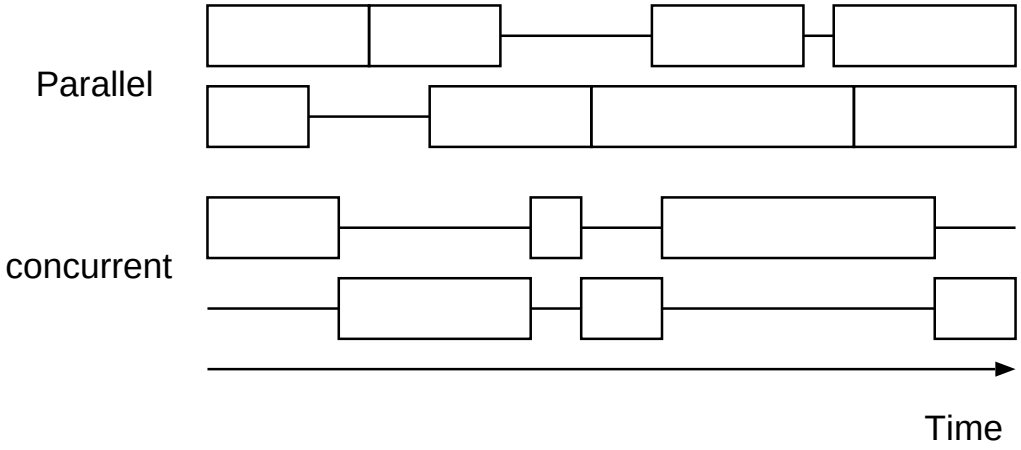

図 1.1 並列（parallel）の並行（concurrent）の違い

数でも構わない。並行処理は1.4 節でも述べるが、実時間に対して行われている処理は1つである。それに対して並列処理はプロセッサの数によって行われてる処理の数が変わってくる。図1.1 に並列処理と並行処理の模式図を示す。

白色の枠の部分がそれぞれ処理を表している。図1.1 の場合、並列処理の方はプロセッサが2個あると考えられる。並列処理は実時間に対して処理が重なっているが、並列処理の場合は実時間に対して行われている処理は1つである。

よく似た概念に分散処理 (distributed processing) がある。これはネットワークで結ばれた複数の計算機を用いて処理を行うことをいう。

1.3 Java 言語について

今回、並行なシミュレーションを行うに当たって用いるコンピュータ言語は Java である。Java 言語はアメリカの Sun Microsystems 社が開発したオブジェクト指向型プログラミング言語で、1995 年同社が発表した。Java 言語の最大の特徴はオペレーティングシステムに依存しないということである。Java 言語自身の中に仮想マシン (これを Java Virtual Machine という) を持っていて、この仮想マシンが理解できるようなコンパイラを含んでいる。Java Virtual Machine はさらに、実際の計算機側が理解できるようにインタープリタで翻訳をする。ユーザはソースをコンパイルした後、インタープリタを用いて実際の計算機で実行するような形をとる。このように

Java Virtual Machine が仲介することで異なったオペレーティングシステムでも、Java がインストールされた計算機であればプログラムを実行させることができる。

また Java は非常に多くのクラスを持っており、これを用いることでアプリケーションに応じた柔軟なプログラミングが可能となっている。

1.4 マルチスレッド

並行処理はマルチスレッドのプログラムによって実行される。ここではマルチスレッドについて説明する。スレッドとは本来「糸」を意味するもので、ここではプログラムを実行している主体のことを指す。従来のシーケンシャルなプログラムは時間に対して実行しているスレッドが 1 つしかないため、これをシングルスレッドのプログラムと呼ぶ。マルチスレッドとは時間に対して実行している 2 つ以上のスレッドを持ち、その処理を並行に行っていくことをいう。図 1.2 にシングルスレッドとマルチスレッドを比較した模式図を示す。白色部分と影色部分がプログラムの実行部分としている。シングルスレッドの場合、常にプログラムを実行している主体は 1 つしかないため、片方の処理が終了した後もう一方の処理に移る。マルチスレッドの場合ではスレッドの数に応じた数だけプログラムを実行している主体がある。つまり、図 1.2 の場合では 2 つの処理が並行に実行される。図 1.2 において、さらに詳しく見てみると、Thread1 がプログラムを実行している場合は, もう一方の Thread2 は休止状態にある。つまり、ある瞬間においてのプログラムの実行はシーケンシャルであるが、実行されている

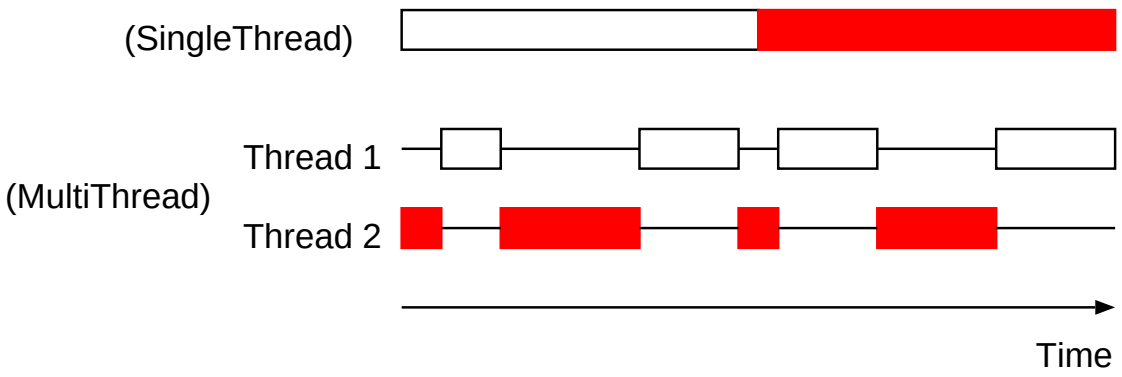

図 1.2 シングルスレッドとマルチスレッドの違い

スレッドは2つあり、全体的に見ていくとコンカレントである。この切り替わるタイミングは計算機環境、及びJavaの実行系に依存するが、排他制御が行われるためスレッドの実行が重なり合うことはない。

Javaはマルチスレッドを標準でサポートしており、したがってシミュレーションごとに異なった並行動作の定義をすることなくまた、そのシミュレーションに応じたスレッドの数、処理を行うことができる。

このマルチスレッドを用いることによって並行動作のシミュレーションが実現可能であると考えられる。

1.5 本研究の目的

本研究では逐次型のシーケンシャルなプログラムではなく、並行で動作するコンカレントなプログラムをマルチスレッドを用いてJava言語で作成することを目的とし、従来のシーケンシャルなシミュレーションとの比較を行う。その中でも物理分野の代表例としてイジング模型のシミュレーション、また生化学分野の代表例として酵素と基質の反応のシミュレーション、身近な現象例として金平糖の生成のシミュレーションを行う。

第 2 章 シミュレーション

2.1 イジング模型のシミュレーション

2.1.1 概要

物理学分野の例として、イジング模型のシミュレーションを行う。イジング模型とは磁気双極子の向きが上下 2 方向だけに限られている最も簡単な磁性体のモデルであり、ハイゼンベルグの弟子であったイジングによって 1925 年に導入された。イジング模型は、モデルの簡単さのおかげで数学的解析が可能となり、物理学のみならず、数学的にも広く研究されている。イジング模型は磁気モーメントモデルであり、上向きのスピン、下向きのスピン、2 つのスピンの状態のみが存在する。図 2.1 に簡単なイジング模型を示す。イジング模型においては、自転軸が z 軸と平行で自転の向きは上

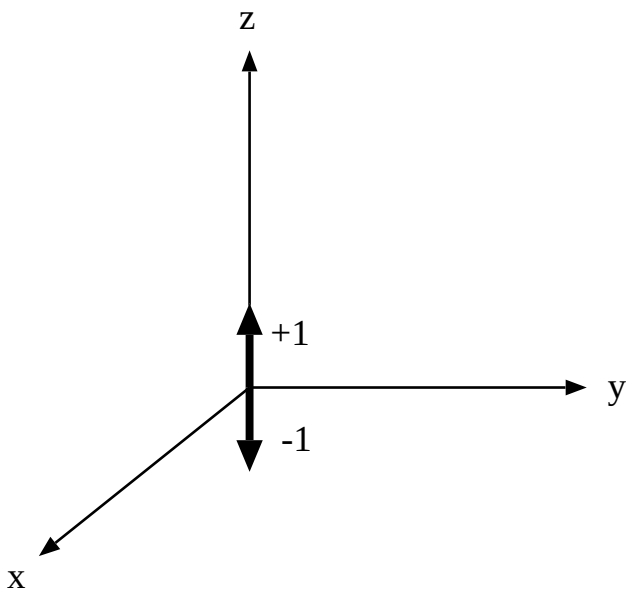

図 2.1 イジング模型

でも述べたように右回り、左回りの2通りである。磁極の向きが上向きの時のスピンの状態を +1 と呼び、下向きのスピンの状態を -1 と呼ぶ。

イジング模型は昔から多くの学者が解析を進めており、今となってはかなり論理が確立されている。このため、シミュレーションの比較を容易に行うことができると思われる。例えば、上田[5]はモンテカルロ法を用いて2次元イジング模型のシミュレーションを行い、スピン配位のパターンを的確に示している。

2.1.2 プログラムの条件

今回は簡略化するため、1次元の磁気モーメントモデルについてシミュレーションを行う。プログラムの条件として今回は、上向きスピンを1、下向きスピンを0と表現することにし、また、1つのスピンが1つのスレッドに対応し、それぞれ互いに素となることで、おのおののスピンがコンカレントな動作をするようにする。このようにすることでスピン同士1つ1つがある規則の元で変化していくことが可能である。変化させる規則を下に示す。

1. 同じ向きに挟まれているスピンは同じ向きに変化する
2. 1. に当てはまらない場合は少ない確率で変化させる
3. 周期的境界条件を導入することで、スピンの端と端の影響を考慮しない

以上の条件で、同時間内にスピンの向きがどのように変化していくかを検証していくことにする。

2.1.3 プログラムの流れ

2.1.2 節で述べたような条件を踏まえた上で、プログラムの流れを表したものを図2.2に示す。図中の矢印はスレッドを表していて、また、矢印の上の数字はスレッドの整理番号である。同じ番号のついたスレッドはそれぞれ同じスレッドを表している。また、上から下に時間が流れているものとする。

まず、スピンの向き、つまり0、1の値を持ったスレッドをスピンの向きを評価するメソッドへ登録する。このスレッドは全スピンの数だけ登録さ

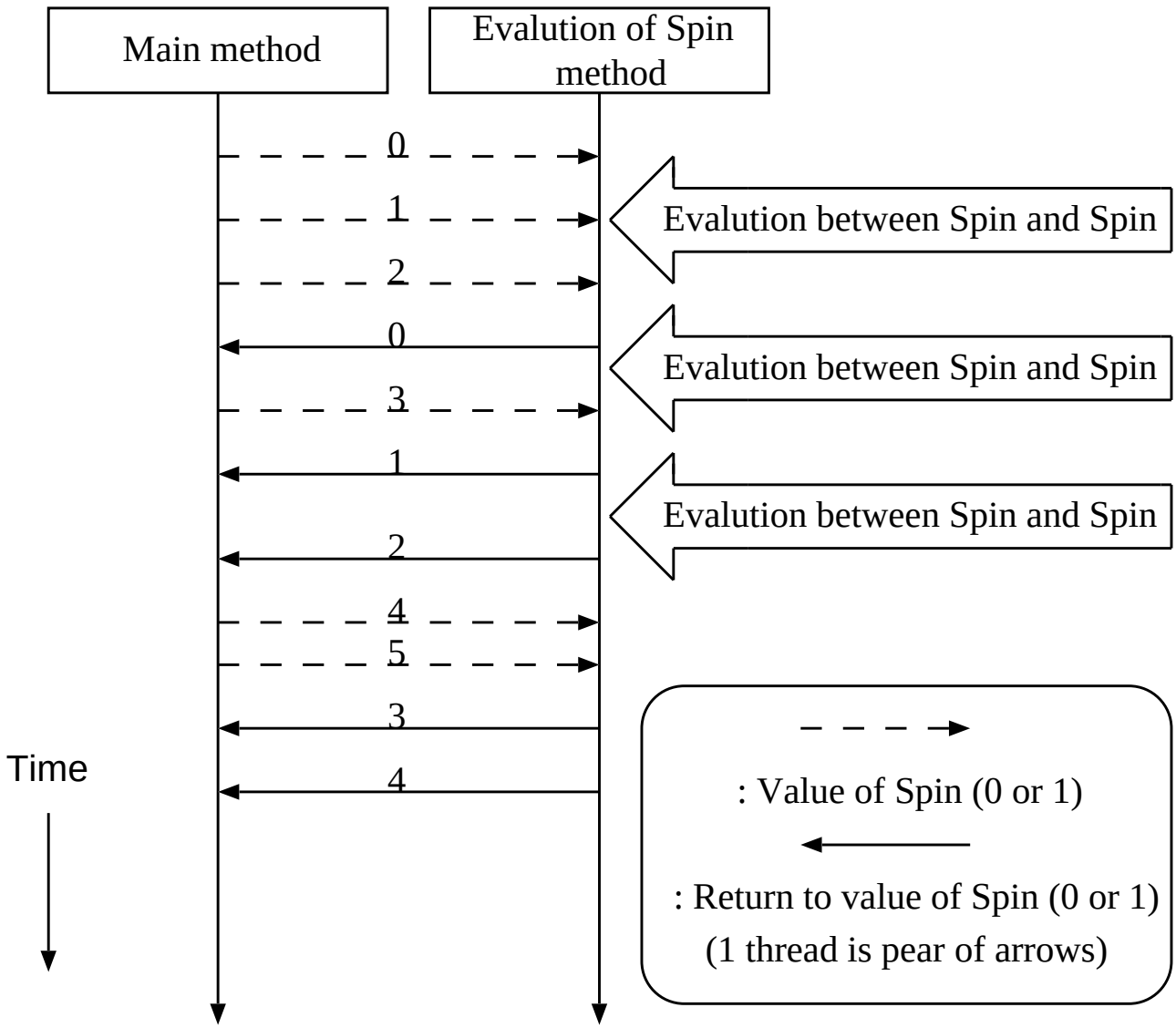

図 2.2 イジング模型シミュレーションのプログラムの流れ

れる。次に評価メソッドが登録されたスレッドが持つ値を 2.1.2 節の条件で評価して、スピンの向きを変更する。そして、その評価し終えたスレッドの持つスピンの向きを順次表示していく。表示を終えたスレッドは再び評価メソッドの方へ登録され、一連の動作を繰り返す。当然、これら 3 つの動作はコンカレントに行われる。

2.2 酵素と基質の反応のシミュレーション

2.2.1 概要

続いて、生化学分野の代表例として、酵素と基質の反応のシミュレーションについて説明する。今回、以下のような化学反応式を用いる。

$$E + S \underset{\alpha'}{\overset{\alpha}{\rightleftarrows}} ES \overset{\beta}{\rightarrow} P \tag{2.1}$$

E は酵素、S は基質、P は生成物をそれぞれ表し、α、α' および β は反応スピードを規定する確率を表す。式の左から流れを追ってみると、まず、酵素 E は基質 S と反応確率 α で別の物質 ES へと変化する。この物質 ES はさらに反応が進むと確率 β で生成物 P を生成する。しかし、反応が進まない場合は元の酵素 E と基質 S の状態になる。このモデルを反応速度論的な微分方程式で表現すると、以下のようになる。

$$\begin{cases} \dfrac{\mathrm{d}[P]}{\mathrm{d}t} &= \beta[ES] \\ \dfrac{\mathrm{d}[ES]}{\mathrm{d}t} &= \alpha[E][S] - (\alpha' + \beta)[ES] \\ \dfrac{\mathrm{d}[E]}{\mathrm{d}t} &= -\alpha[E][S] + (\beta + \alpha')[ES] \\ \dfrac{\mathrm{d}[ES]}{\mathrm{d}t} &= -\alpha[E][S] - \alpha'[ES] \end{cases} \tag{2.2}$$

同様な表現としてペトリネットと呼ばれる数学的なモデル化ツールがある。1962 年、ペトリの論文で発表され、以来、

- 並行的（concurrent）
- 非同期的（asynchronous）
- 分散的（distributed）
- 並列的（parallel）
- 非決定的（nondeterministic）

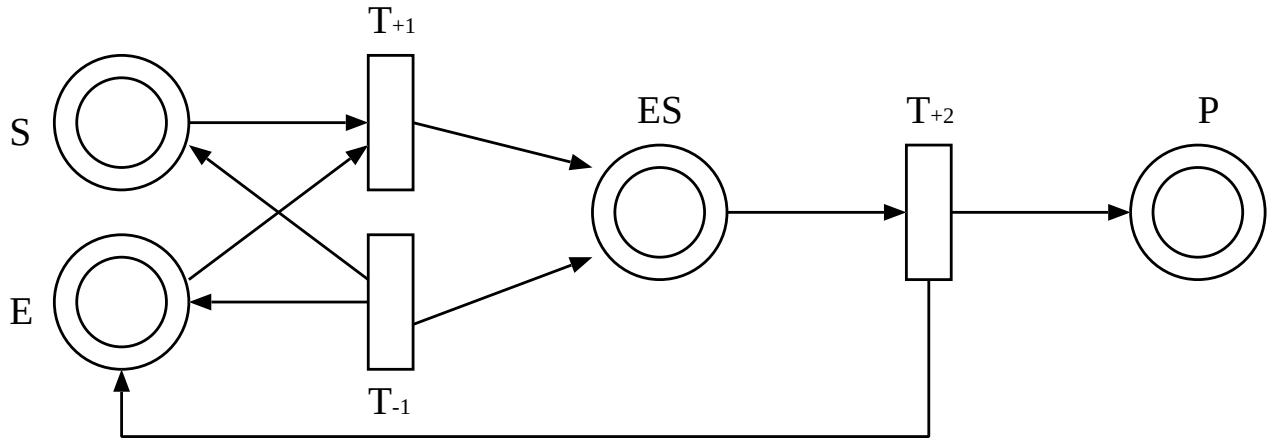

図 2.3 酵素と基質の反応のペトリネットによる記述

- 確率的（stochastic）

な動作を特徴とする情報システムを記述・研究するツールとして使用されている。ペトリネットはグラフィカルなツールとしてフローチャートやブロックダイアグラム・ネットワークと同じように。システムの構造の可視的な表現手段として使用できる。具体的には、プレース (place)、トランジション (transition) と呼ばれる二種類のノードを持つ二部有向グラフである。プレースは円または二重円で表されるノードであり、状態を表す。トランジションは箱で表されるノードであり、事象を表す。これらを結ぶアーク (arc) は状態、事象の間の関係を示す。これを踏まえて、今回の例をペトリネットで表したものを図 2.3 に示す。

これを数式で表すと以下のように表すことができる。

$$\begin{cases} v\left(T_{+1}\right) &= \alpha[E][S] \\ v\left(T_{-1}\right) &= \alpha'[ES] \\ v\left(T_{+2}\right) &= \beta[ES] \end{cases} \tag{2.3}$$

ここで v はトランジションの発火速度を表している。トランジションの発火とはすべての入力条件が満たされたとき、すべての出力条件が満たされるというものである。

この酵素と基質の反応をコンカレントなシミュレーションで表していく。

2.2.2　プログラムの条件

プログラムの条件として、まず

1. 酵素と基質はそれぞれ別のスレッドで動く
2. 反応確率によって反応が進んでいく
3. 1 度、別の物質 P を生成した酵素・基質は以後、反応を起こさない

があげられる。反応の確率はパラメータで与えられ、変化可能である。

2.2.3　プログラムの流れ

プログラムの流れを表したものを図 2.4 に示す。時間は図中、上から下へ流れていて、図中の数字は酵素の状態を表している。まず酵素 E は反応メソッドに登録される (図中 1 の状態)。登録された酵素 E は基質 S が登録されるのを待ち、基質 S が反応メソッドに登録されると、酵素と基質はペアを作る (図中 2 の状態)。酵素 E は与えられた確率 α によって ES へと変化する (図中 3 の状態)。反応し終えた ES はさらに別の確率 β で生成物 P を生成する (図中 4 の状態)、または確率 α' によって再び酵素 E と基質 S に戻る (図中 0 の状態)。この一連の流れはすべてコンカレントで行われる。

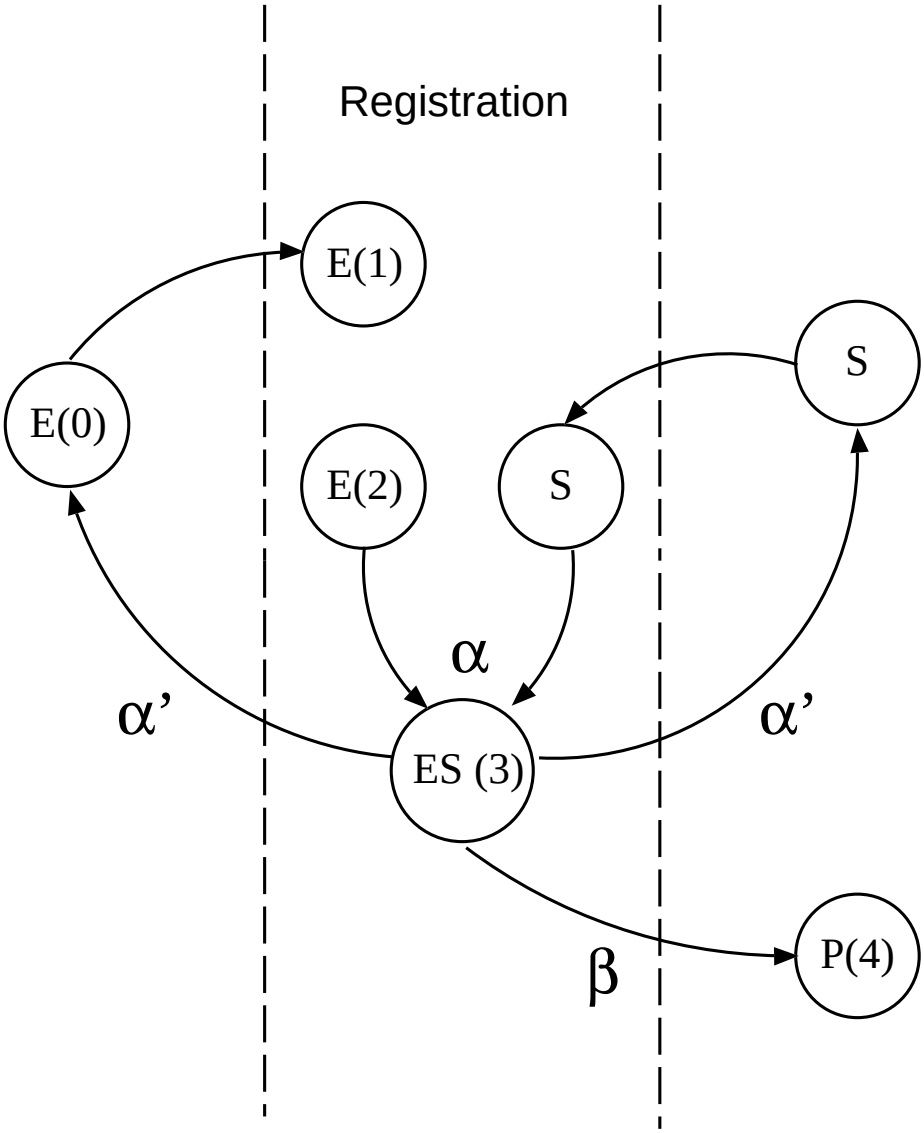

図 2.4 酵素と基質の反応のシミュレーションのプログラムの流れ

2.3 金平糖の生成シミュレーション

2.3.1 概要

金平糖は古くから日本人に親しまれてきたお菓子である。室町時代末期もしくは安土桃山時代にポルトガルの宣教師によって伝えられ、名前の由来はポルトガル語の「コンフェイト」である。金平糖の特徴は、その表面にいぼ状の突起をもっている事である。中田[7) 8)]の金平糖のシミュレーションでは、丸い核にランダムに砂糖を付着させ、突き出した部分ほど、砂糖が付着しやすくさせるというアルゴリズムを用いている。

まず、金平糖の核となる単位円 S を定める。その S を m 個の等間隔の弧 I_i に分割する。その1つの弧 I_i(i は1〜m) に対してモンテカルロ法を用いて砂糖粒を付着させる。各モンテカルロステップ毎に c 個の砂糖粒を S に付着させ、i 番目の弧 I_i に n 回のモンテカルロステップで蓄積した砂糖粒の個数を $S_i(n)$ とする。n 回のモンテカルロステップで砂糖粒が弧 I_i に付着する確率 $P_i(n)$ を

$$P_i(n) = \frac{0.5(1-w)S_{i-1}(n) + wS_i(n) + 0.5(1-w)S_{i+1}(n)}{N} \tag{2.4}$$

として与える。但し、w は重みであり、$1/3 \leq w \leq 1$ である。N は n 回のモンテカルロステップにおける全砂糖粒の数である。核の半径を b とすると、

$$N = \sum_{i=1}^{m} S_i(n) = nc + mb \tag{2.5}$$

で与えられる。これらを用いて、同時並行な金平糖の突起の成長をシミュレーションする。

2.3.2 プログラムの条件

プログラムの条件は2.3.1を踏まえ、

1. 砂糖粒はそれぞれ別々のスレッドで動き、確率 $P_i(n)$ で付着していく
2. 付着した数が多い個所ほど、さらに付着する

以上である。

2.3.3　プログラムの流れ

プログラムの流れを表したものを図 2.5 に示す。まず、砂糖の分割数分の配列を作成する。そこにランダムな確率で砂糖の粒を入れていく。プログラムの条件 2. にあるように、砂糖が入っているセルにはさらに砂糖が入りやすくなっている。この配列は図 2.5 の下部にあるように周期的境界条件により円形になっている。これをグラフィック表示させることで金平糖の様子がシミュレートされる。

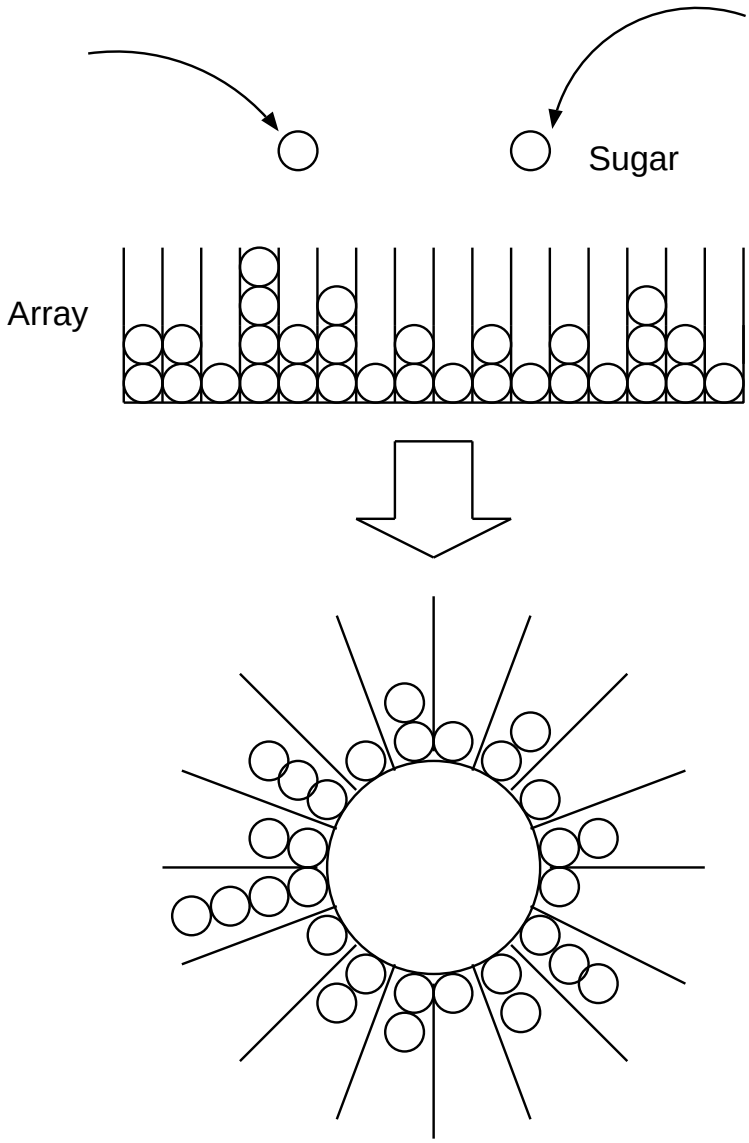

図 2.5 金平糖の生成シミュレーションのプログラムの流れ

第 3 章 結果及び検討

3.1 イジング模型のシミュレーション

3.1.1 実行結果

実行中のシミュレーションの様子を図 3.1 に示す。今回はスピンの数を 70 個としている。

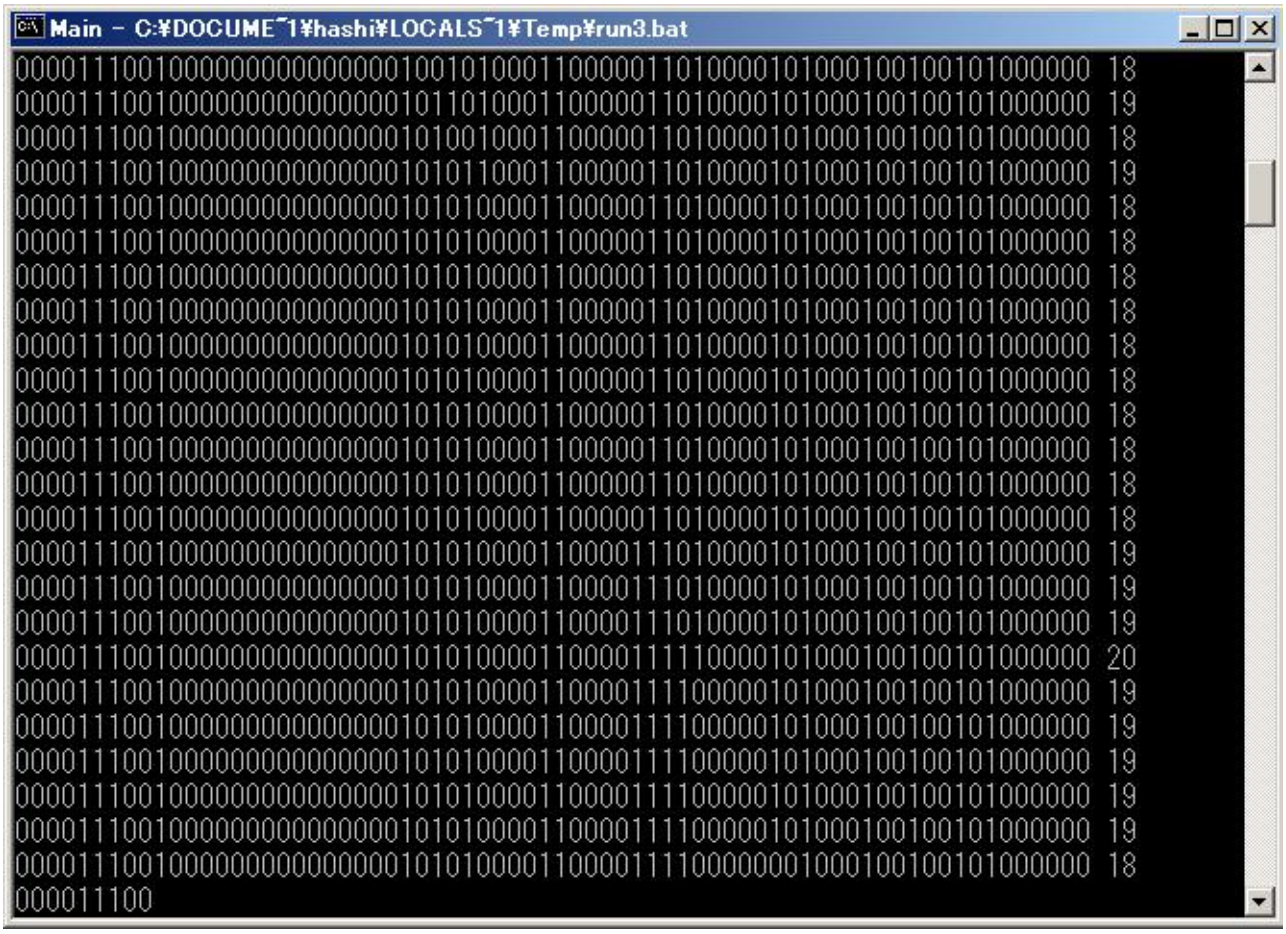

図 3.1 イジング模型のシミュレーション実行中の様子

```
--------------------------------------------------------------------------------
Result
--------------------------------------------------------------------------------
 1 | 0000001111111111111111100000000000001111100000001100001101110000010100 1 32
 2 | 0000001111111111111111100000000000001111100000001100001101110000010100 1 32
 3 | 0000001111111111111111100000100000001111100000001100001101110000010100 1 33
 4 | 0000001111111111111111100000100000001111100000001100001101110000010100 1 33
 5 | 0000001111111111111111100000100000001111100000001100001101110000010100 1 33
 6 | 0000001111111111111111100000100000001111100000001100001101110000010100 1 33
 7 | 0000001111111111111111100000100000001111100000001100001111110000010100 1 33
 8 | 0000001111111111111111100000100000001111100000001100001111110000010100 1 34
 9 | 0000001111111111111111100000100000001111100000001100001111110000010100 1 34
10 | 0000001111111111111111100000100000001111100000001100001111110000010100 1 34
11 | 0000001111111111111111100000100000001111100000001100001111110000010100 1 34
```

図 3.2 イジング模型のシミュレーションのある部分の様子

図 3.2 に実行中のある部分を書き出したものを示す。左端の数字はそれぞれ行数を示し、右端の数字は上向きスピンの数、つまり 1 の値をカウントしたものを示している。スピンの向きがある部分では上向き、下向きとはっきり分かれていることが分かる。また、7 行目においては下向きスピンが周りのスピンによって上向きスピンに変化している部分である。また、3 行目では周りがゼロで囲まれているにもかかわらず、少ない確率で上向きスピンへと変わっている部分である。

3.1.2 検証

このシミュレーションを何度も繰り返すことで、あるパターンが現れることがわかった。図 3.3 はそのパターンが顕著に出たシミュレーション結果をグラフに表したものである。横軸を回数、縦軸を上向きスピンの個数とし、初期値として 70 個のスピンのうち、その半分の 35 個を上向きスピンとしている。1 つ目のパターンとしては図 3.3 中 (a) のように、上向きスピンの数がゼロ近づいていき、最後はゼロに行き着くパターン、2 つ目のパターンとして、図 3.3 中 (b) のように向きスピンの数が増減して総スピンの数の半分の 35 個あたりを行き来するそして 3 つ目のパターンとして図 3.3 中 (c) のように、(a) とは逆にすべてのスピンが上向きに変わっていくパターンの 3 つに分類された。また、(b) のパターンは時間を増やしていけば、最終的に (a) または (c) のパターンに当てはまった。

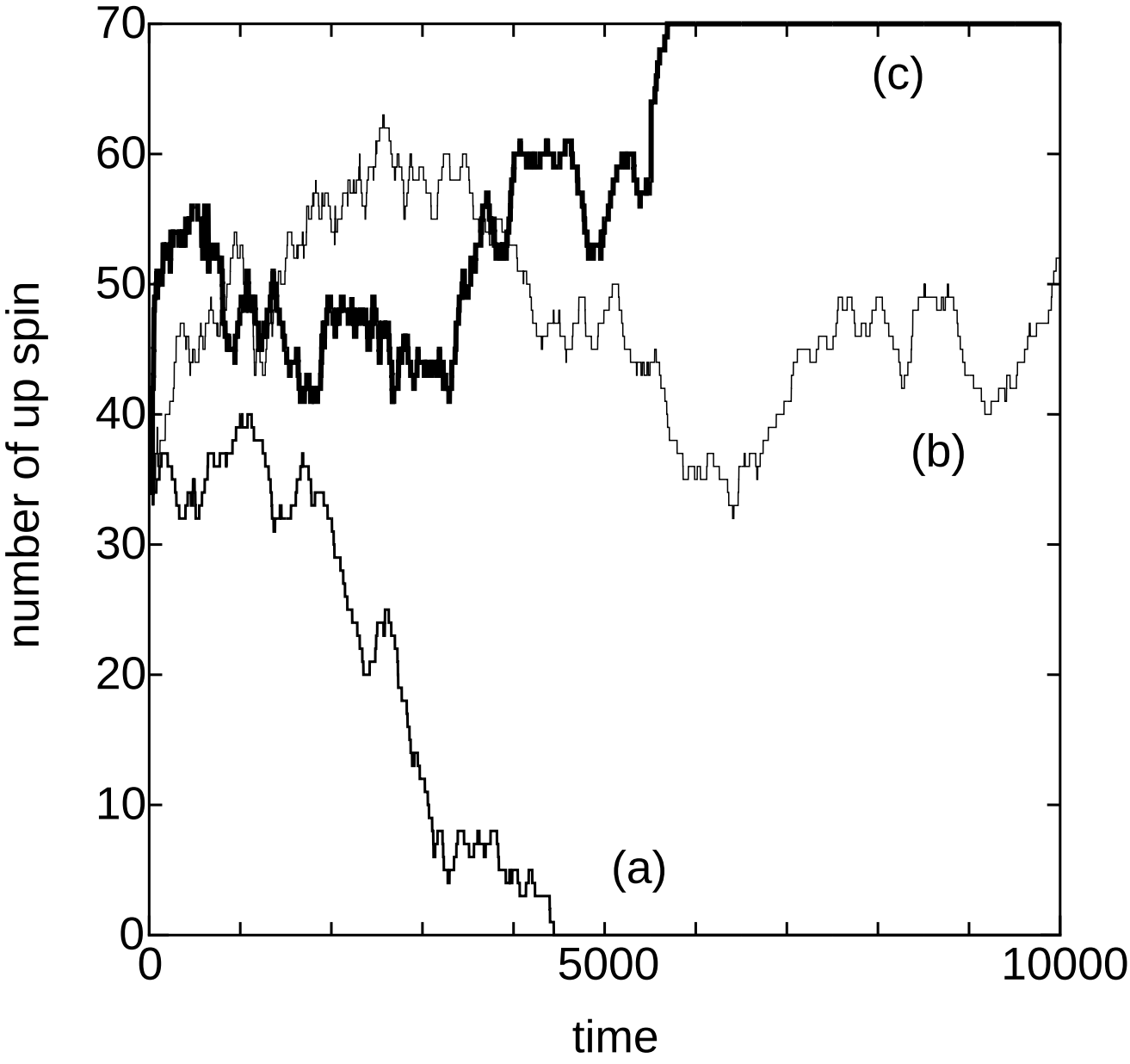

図 3.3 イジング模型のシミュレーションに現れた 3 つのパターン

このパターンはよく知られているイジング模型のパターンであり、このシミュレーションの妥当性を示している。また、上田 [5] が行ったシミュレーションとは次元が違うものの、基本をなすパターンは同じである。このようにコンカレントなシミュレーションでも逐次型と同じようなイジング模型を表すことができることが分かった。

3.2　酵素と基質の反応のシミュレーション

3.2.1　実行結果

実行中のシミュレーションの様子を図 3.4 に示す。今回は酵素の数を 4 つ、基質の数を 7 つとしている。図 3.5 に実行中のある部分を書き出したものを示す。

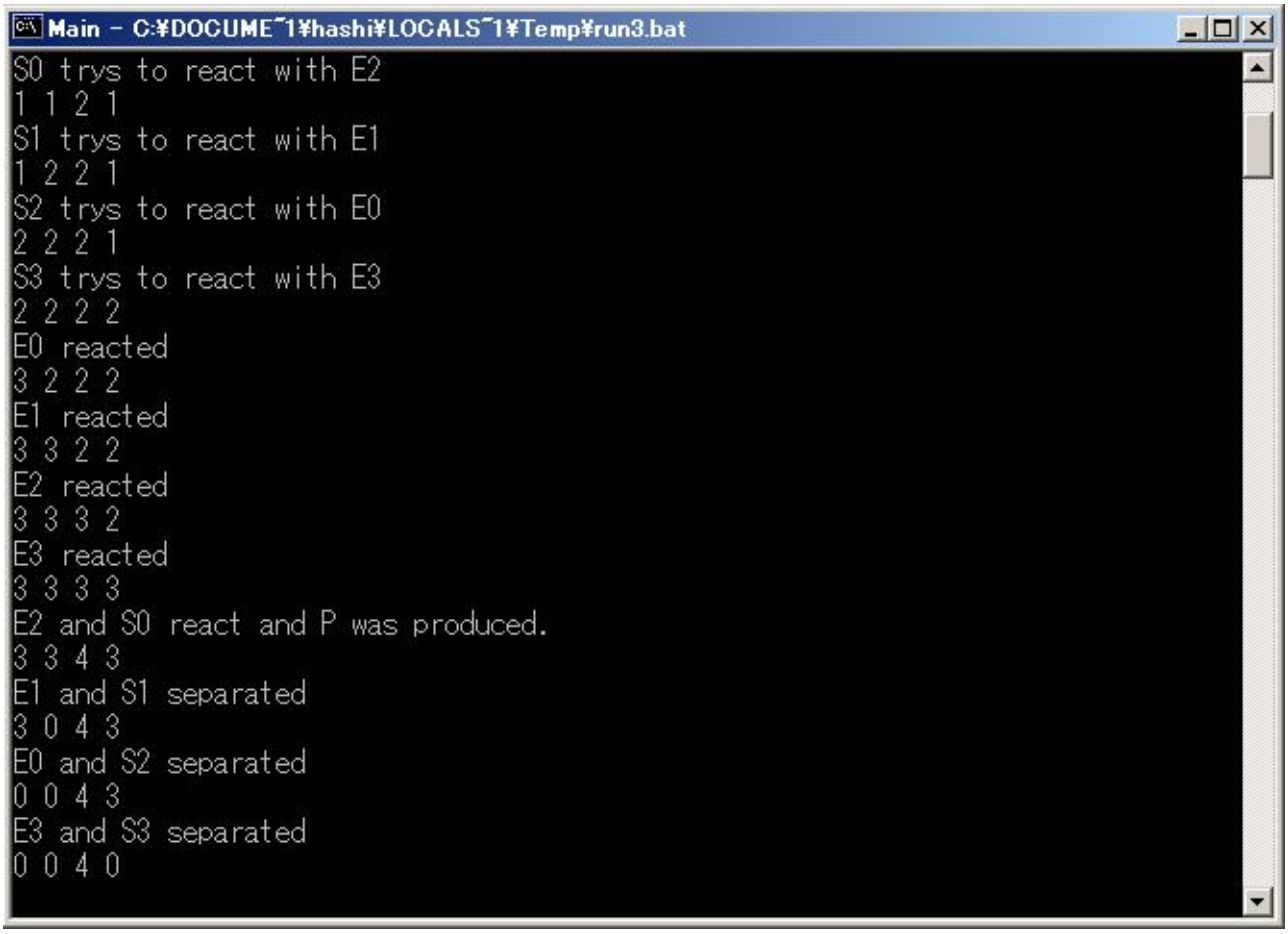

図 3.4 酵素と基質の反応のシミュレーション実行中の様子

```
----------------------------------------
Result
----------------------------------------
 1 | E0 registered
 2 | 1 0 0 0
 3 | E1 registered
 4 | 1 1 0 0
 5 | E2 registered
 6 | 1 1 1 0
 7 | E3 registered
 8 | 1 1 1 1
 9 | E2 trys to react with S0
10 | 1 1 2 1
11 | E3 trys to react with S1
12 | 1 1 2 2
13 | E0 trys to react with S2
14 | 2 1 2 2
15 | E1 trys to react with S3
16 | 2 2 2 2
17 | E0 turn out ES
18 | 3 2 2 2
19 | E1 turn out ES
20 | 3 3 2 2
21 | E2 turn out ES
22 | 3 3 3 2
23 | E3 turn out ES
24 | 3 3 3 3
25 | E3 & S1 separate
26 | 3 3 3 0
27 | E0 & S2 separate
28 | 0 3 3 0
29 | E1 & S3 separate
30 | 0 0 3 0
31 | E2 & S0 react, P was produced.
32 | 0 0 4 0
33 | E1 registered
34 | 0 1 4 0
35 | E3 registered
36 | 0 1 4 1
37 | E3 trys to react with S4
38 | 0 1 4 2
39 | E1 trys to react with S1
40 | 0 2 4 2
41 | E0 registered
42 | 1 2 4 2
43 | E1 turn out ES
44 | 1 3 4 2
45 | E3 turn out ES
46 | 1 3 4 3
47 | E3 & S4 react, P was produced.
48 | 1 3 4 4
49 | E0 trys to react with S5
50 | 2 3 4 4
51 | E1 & S1 separate
52 | 2 0 4 4
53 | E0 turn out ES
54 | 3 0 4 4
55 | E1 registered
56 | 3 1 4 4
57 | E1 trys to react with S4
58 | 3 2 4 4
59 | E0 & S5 separate
60 | 0 2 4 4
61 | E0 registered
62 | 1 2 4 4
63 | E0 trys to react with S1
64 | 2 2 4 4
65 | E1 turn out ES
66 | 2 3 4 4
67 | E1 & S4 react, P was produced.
68 | 2 4 4 4
69 | E0 turn out ES
70 | 3 4 4 4
71 | E0 & S1 react, P was produced.
72 | 4 4 4 4
```

図 3.5 酵素と基質の反応のシミュレーションの様子

左側の数字は行番号を示しており、それぞれ酵素の様子を英語と数字で表現している。偶数行の数字は、図2.4 中の数字に対応している。酵素が反応メソッドに登録されると1、酵素と基質がペアを作ると2、酵素と基質が反応すると3、生成物を作ると4、元に戻ると0と表示する。初め、反応メソッドの方へ酵素が順に登録されていき、最終的にすべての酵素が生成物 P を生成した状態で（つまり結果中の数字がすべて4になった状態）プログラムは終了となる。その中の酵素の1つ、$E1$ に注目して流れを追っていく。まず、酵素 $E1$ は反応メソッドに登録される（シミュレーション結果3行目。以下同様）。登録された $E1$ は登録されていく基質を待ち、そして、登録された基質とペアを作る（15行目）。そして与えられた確率で反応し、ES となる（19行目）。ここで、さらにある確率でまた酵素と基質に分かれる（29行目）。再び登録メソッドの方へ登録され（33行目）、この一連の動作を繰り返し、最終的に生成物を作っている（67行目）。

$E1$ だけを見てきたが途中、他の酵素は別の反応をしている。つまり、1つ1つの酵素が別なスレッドで動いており、コンカレントに動作していることが確認できると思う。

3.2.2 検証

まず、2.2.1節で述べた反応速度論的微分方程式は以下のようであった。

$$\begin{cases} \dfrac{\mathrm{d}[P]}{\mathrm{d}t} &= \beta[ES] \\ \dfrac{\mathrm{d}[ES]}{\mathrm{d}t} &= \alpha[E][S] - (\alpha' + \beta)[ES] \\ \dfrac{\mathrm{d}[E]}{\mathrm{d}t} &= -\alpha[E][S] + (\beta + \alpha')[ES] \\ \dfrac{\mathrm{d}[ES]}{\mathrm{d}t} &= -\alpha[E][S] - \alpha'[ES] \end{cases} \tag{3.1}$$

この微分方程式をJava言語上でプログラムし、今回のシミュレーションとの比較を行った。図3.6 は上の微分方程式を解いた結果を表したものである。縦軸はそれぞれの要素の濃度、横軸は時間である。

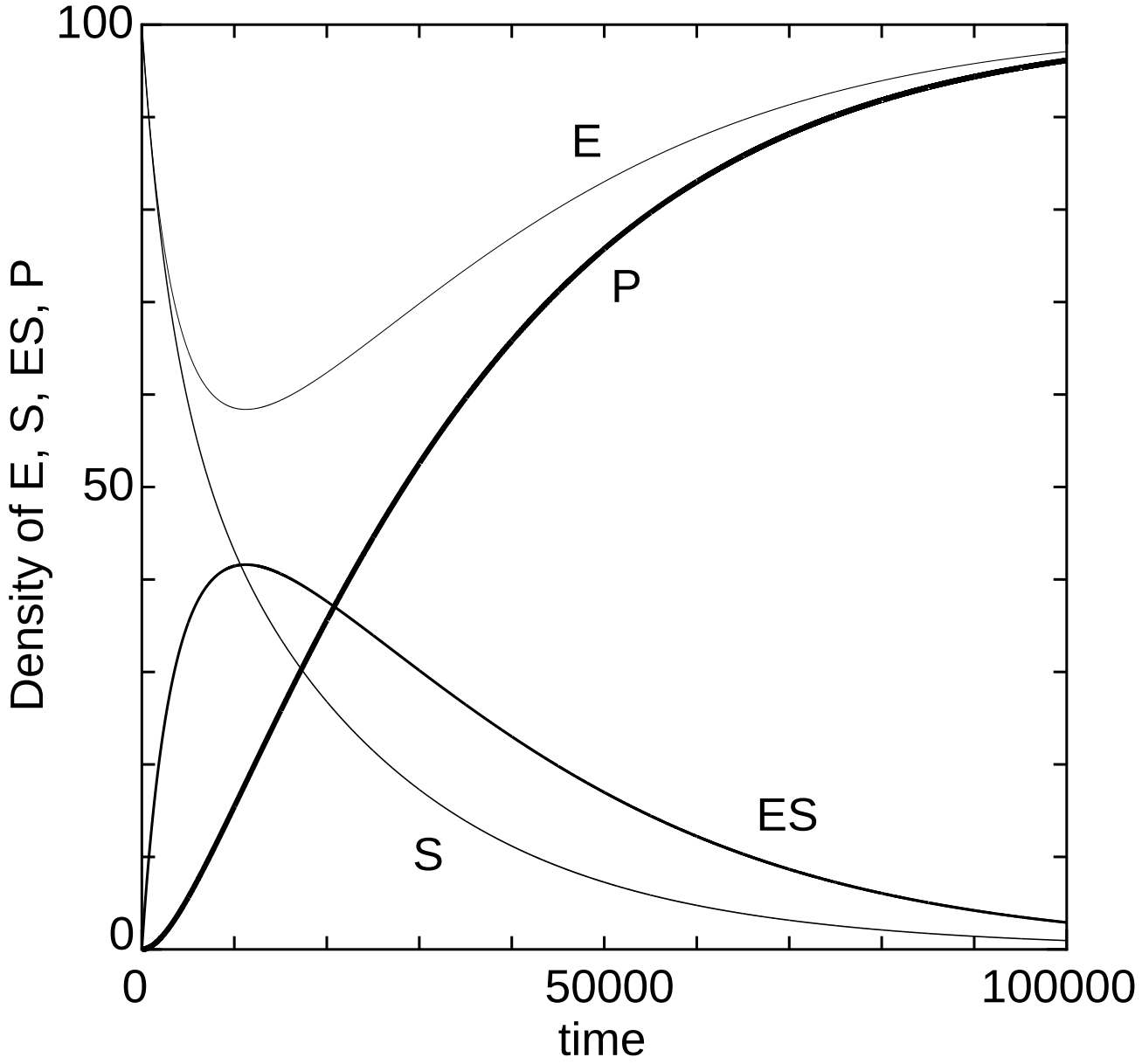

図 3.6 酵素と基質の反応の微分方程式の解

次に今回行ったシミュレーションでのそれぞれの要素の時間的変化を表したものを図 3.7 に示す。なお、3.2.1 節では酵素の数を 4、基質の数を 7 としたが、今回はシミュレーションの妥当性を検討するため、酵素と基質の数をそれぞれ 100 に増やしている。図 3.6 と図 3.7 を比較するとおおよそではあるが同じような変化をしているのが分かる。

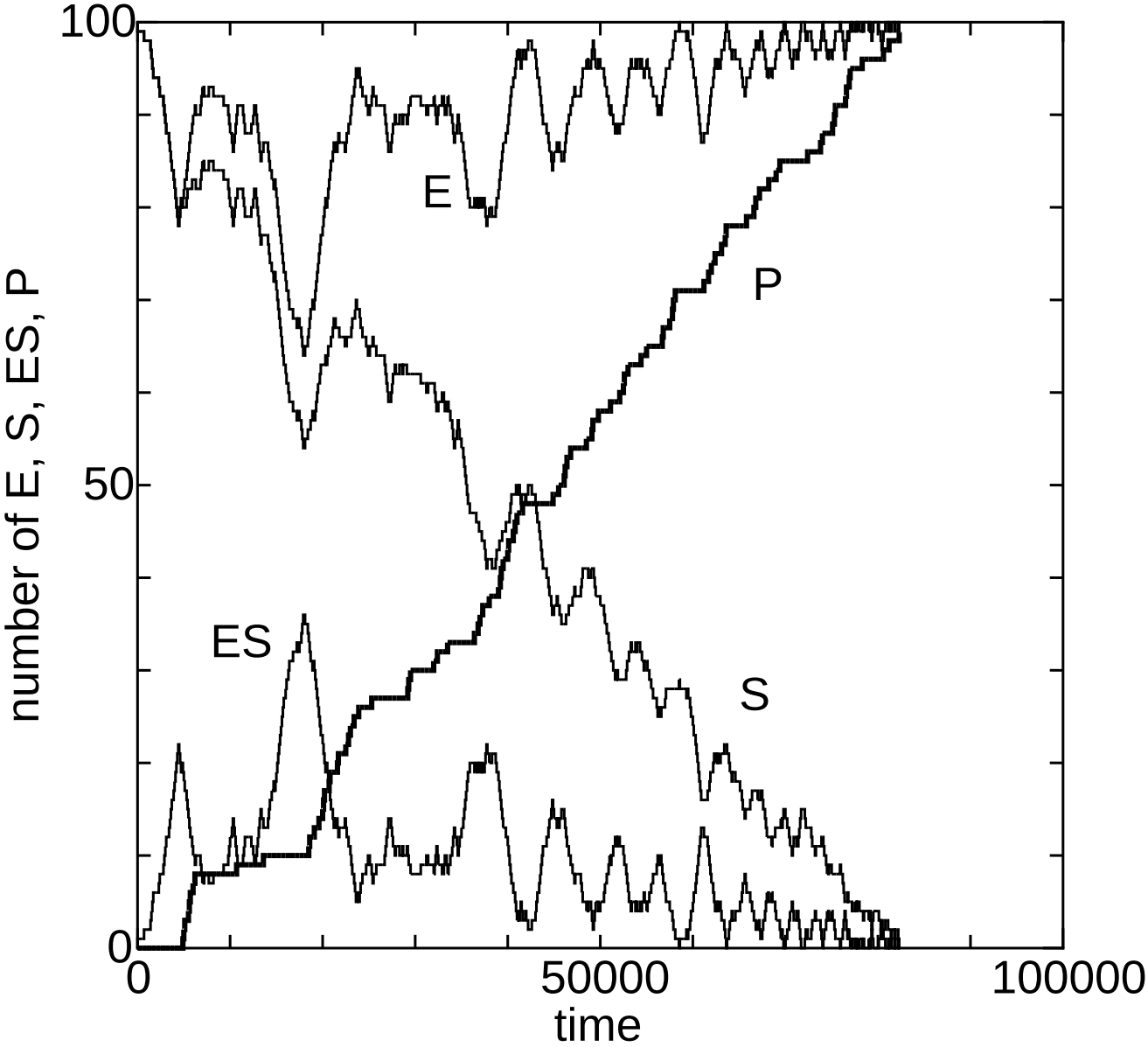

図 3.7 酵素と基質の反応のシミュレーションにおけるそれぞれの要素数の変化

3.3　金平糖の突起生成のシミュレーション

3.3.1　実行結果

3.3.2　検証

第 4 章 結論と今後の課題

4.1 結論

今回、シーケンシャルではない、コンカレントな動作を行うプログラムを Jva 言語を用いて作成し物理・化学・生物学的現象のシミュレーションを行った。その結果以下のことがわかった。

- Java 言語を用いることでコンカレントで動作可能なシミュレーションを実行できる。このことはマルチスレッドプログラミングを行うことで実証できた。
- 現象に応じた柔軟な、自然なプログラミングが可能である。つまり、実時間で刻一刻と変わっていく現象をシーケンシャルなシミュレーションで行うことに無理があり、これをコンカレントな動作で実行させることで、より現実に近いシミュレーションが可能である。
- コンカレントなシミュレーションとシーケンシャルなシミュレーションとでは、最終的に得られる結果は等しくなる。今回の 3 つの現象においてはすべて、最終的な結果は同じであった。

4.2 今後の課題

今回行ったシミュレーションはモデルが簡単なものであった。今後はさらに複雑な現象のシミュレーションを行い、理論値との整合性、およびシーケンシャルなシミュレーションとの検証を行っていく。また、Java 言語のグラフィック処理を利用することで、ユーザが現象を理解する際の手助けとなるようなグラフィカルなシミュレーションも行っていきたい。

謝辞

本研究を行うにあたり、多大な御指導、御助言を頂いた小田部荘司助教授に深く感謝します。また並行処理の基本、及び個々の現象のシミュレーションについての助言や指導をして頂いた有明高専電子情報工学科松野哲也助教授、金平糖のシミュレーションを提案して頂いた松下照男教授、微分方程式プログラムを提案してくださった小田部研究室修士山内浩太郎さんに深く感謝します。

最後に本研究を行うにあたり、色々と支援を頂いた小田部研究室、及び松下研究室の皆さんに深く感謝します。

参 考 文 献

1) 結城浩：Java 言語で学ぶデザインパターン入門［マルチスレッド編］

2) 高橋麻奈：やさしい Java （2001 年）

3) Steven Holzner 著、トップスタジオ訳：Java プログラミング Black Book 2nd Edition （2002 年）

4) 黒田耕嗣：秩序・無秩序の世界—イジング・モデルと相転移の話（昭和 62 年）

5) 上田顕：コンピュータシミュレーション—マクロな系の中の原子運動—（1990 年） p. 96–102

6) 日本物理学会：ランダム系の物理学（昭和 56 年）

7) Nobuyuki Murai、 Tomoichi Nakata：American Journal of Physics - May 1988 - volume 56, Issue 5, pp. 459–462 （1988 年）

8) 中田友一：おーい、コンペイトー（1990 年）