並列計算のための乱数列の高速初期化法
—5項GFSR法の場合—

2004MM013 波多野達　2004MM015 原田政季
指導教員: 伏見正則

1　はじめに

現在、社会において、多くの場面でシミュレーションが用いられている。このシミュレーションに使用するのが、乱数である。乱数とは、本来はサイコロを振って出る目の数のようなものを意味するこのことから乱数とは予測不能で、規則性があってはならない。しかし、シミュレーションにおいて何十万という乱数を必要とする場合、サイコロを振って乱数を発生させることは現実的ではない。シミュレーションではコンピュータのプログラムによって発生させた乱数を用いる。プログラムによって、乱数を発生させているのだから、次にでる数はプログラムによって決まってしまう。これでは完全に規則性がないとは言えないが、後に示す乱数の条件さえ満たせば、実験で用いてもさしつかえない。シミュレーションにおいて、使用する乱数の量が膨大になれば、その乱数を発生させるのにも多くの時間を要する。このとき、計算能力の高いコンピュータを使って高速に乱数を発生させれば、短時間で大量の乱数を発生させることができるが、本研究では、それほど処理速度の速くないコンピュータを複数用意し、それぞれを接続し、大きな乱数列を生成する方法を用いる。この方法を「乱数の並列計算」という。この方法で発生させた乱数において、後で述べる乱数の条件を満たしているかを調べ、また、乱数として用いるのにふさわしいかどうかの検定も行い、今回行う乱数の並列計算が正しいものであるかを検証する。

2　乱数の条件

- 長い周期であること
- 再現性があること
- 統計的検定に耐えうること

3　乱数の発生方法

乱数の発生方法として以下のような方法が知られている。

- 平方採中法
- 線形合同法
- 3項GFSR法
- 5項GFSR法
- メルセンヌツイスター法
- セルオートマトン法

本研究では、この中から、線形合同法を一部用いながら、5項GFSR 法を主として乱数を発生させ研究を進めた。

3.1　線形合同法

一様乱数の生成法として最も広く使われてきたのは、レーマー(Lehmer) が1948 年頃に発表した線形合同法である。線形合同法は、漸化式

$$X_n = aX_{n-1} + c \qquad (\mathrm{mod}\ \ M) \tag{1}$$

を用いて非負数列$<X_n>$を生成するものである。区間[0,1) 上の実数型乱数が必要な場合には、$x_n = X_n/M$を使う。$c = 0$の場合を乗算型合同法、$c \neq 0$の場合を混合型合同法と呼ぶ。本研究では、乗算型合同法を用いている。線形合同法においては、最大周期は法M を超えることができない。法Mは32ビットの計算機の場合、最大で$M = 2^{32}$をとることが可能であるが、これは現在では十分な大きさとは言えない。

3.2　線形合同法の問題点

線形合同法によって乱数列を生成する漸化式(1) の次数は1である。すなわち、X_nはX_{n-1}によって定まってしまう。したがって、この数列の一周期の間には同一の数が現れることはない。この特徴は、乱数のランダムネスに影響を及ぼす。これは図1を見ればわかる。図1は$M = 16, a = 5, c = 1, X_0 = 1$として、漸化式(1) によって発生させた乱数をプロットしたものである。この図からわかるように、示された点は直線的な構造を成し、疎になる部分が出てくる。このような性質を「多次元疎結晶構造」と呼ぶ。これは高次元になればなるほど点列の密度が疎になる。

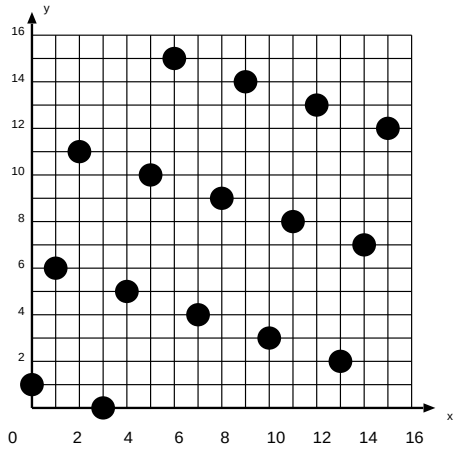

図 1: 線形合同法乱数列の2次元疎結晶構造$M = 16, a = 5, c = 1, X_0 = 1$

4　M系列

線形合同法で発生させた乱数列が多次元疎結晶構造を成すのは、乱数列を生成するのが1次の漸化式であることによる。線形合同法の漸化式が1次であることに対して、高次の漸化式によって乱数を生成する方法として理論的

性質が比較的よくわかっているのが、M系列を用いて乱数列を構成する方法である。演算としては、線形合同法が、乗算であるのに対して、排他的論理和(以下EORと略す)をとる論理演算を用いる。EORをとる演算は、原理的には乗算よりも速くできるので、項数が少ない漸化式を用いれば、線形合同法よりも速く乱数を発生することも可能である。

4.1 M系列法

pを自然数、$c_1, c_2, \cdots, c_{p-1}$を0または1として、漸化式

$$x_{n+p} = c_{p-1}x_{n+p-1} + c_{p-2}x_{n+p-2} + \cdots + c_1 x_{n+1} + x_n \pmod 2 \quad (n = 1, 2, 3, \cdots) \quad (2)$$

によって生成される系列$x_1, x_2, \cdots$の周期が$2^p - 1$ であるとき、この系列のことをM系列という。多項式

$$f(z) = z^p + c_{p-1}z^{p-1} + \cdots + c_1 z + 1 \quad (3)$$

を漸化式(2)の特性多項式と言う。M系列が得られるための条件は、$f(z)$が原始多項式で、かつ、2の初期値$x_1, x_2, \cdots, x_p$の中の少なくともひとつは1となっていることである。特性多項式が3項とか5項とか言うのは、特性多項式の0でない項の個数を表している。例えば、多項式が$z^{89}+z^{38}+1$の場合は3項の特性多項式である。

M系列は0と1からなる系列、すなわち1ビットの系列である。しかし乱数列として使われるのは、例えば合同法乱数のように、もっとけた数の大きい系列である。そこで、M系列を基にして、$l(l >= 2)$ビットの2進数の系列$< x_t >$を次のようにして構成する。

$$X_t = 0.x_{t+\tau_1}x_{t+\tau_2}\cdots x_{t+\tau_l} \quad \text{(2進表現)} \quad (4)$$

ここに、$\tau_1, \tau_2, \cdots, \tau_l$は互いに異なり、$t$ に無関係な定数である。$\tau_1 = 0$としても一般性を失わないので、以後そうすることにする。$< X_t >$は$< x_t >$の位相を適当にずらしたものを各ビットに配置して構成される。つまり、$< X_t >$の各ビット位置に現れる数列は、同一の漸化式によって生成される。したがって、次の漸化式を用いることによって、$< X_t >$を高速に生成することができる。

$$X_t = c_1 X_{t-1} \oplus c_2 X_{t-2} \oplus \cdots \oplus X_{t-p} \quad (5)$$

ただし、$\oplus$はビットごとの繰り上がりなしの足し算を表し、排他的論理和の演算を行うことにより、高速に処理できる。

4.2 3項GFSR法

3項GFSR法は漸化式

$$X_{n+p} = X_{n+q} \oplus X_n \quad (n = 1, 2, 3, \cdots) \quad p > q \quad (6)$$

によって、2進整数を生成し、乱数として用いるものである。

この方法は特性多項式の項数が3項であるため非常に高速で、周期も長くできる。3項特性多項式を利用すると、周期は、例えば$2^{9689} - 1$くらいまで大きくできる。この3項GFSR法は、すでに2005年度の高井の卒業論文[2] において研究されているため、本研究では5項GFSR法について研究する。

4.3 5項GFSR法

これは5項の特性多項式を使用するものであり、漸化式

$$X_{n+p} = X_{n+q_1} \oplus X_{n+q_2} \oplus X_{n+q_3} \oplus X_n \quad (n = 1, 2, 3, \cdots) \quad (7)$$

でwビットの2進整数の列を生成するものである。

この5項GFSR法と3項GFSR法の違いは、3項GFSR法に比べ、p個より長い部分列の分布の1,0の偏りが小さくなっていることである。また、周期は$2^p - 1$である。

5 並列計算の方法

図2のような乱数列の周期があるとする。並列計算をするとき、aとbのコンピュータが図のような範囲の乱数を計算した場合、aとbは互いに独立していないので、並列計算の意味を成さない。

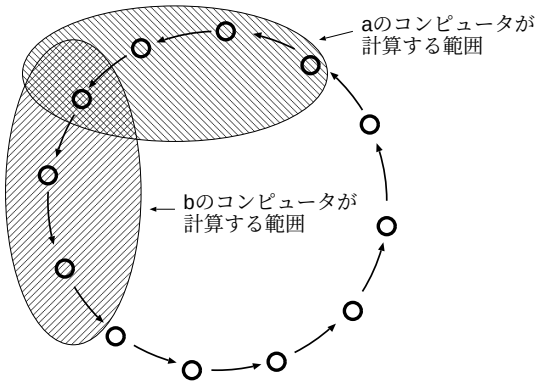

図 2: 周期上の重複がある場合

次に図3のような範囲をそれぞれのコンピュータが計算するのであれば並列計算の意味がある。ただし、互いの乱数列の間に相関があってはならない。なので、できあがった乱数列の相関を調べる必要がある。十分に位相をずらした時の初期値の求め方は次の章で述べる。

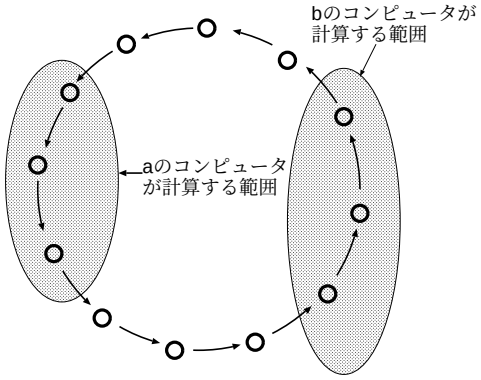

図 3: 十分に位相差がある場合

6 初期値の与え方

本研究では並列計算によって乱数を求めるので、複数のコンピュータに、十分に位相をずらした初期値を与える必要がある。これは、各コンピュータが発生させる乱数の周期上の同一部分を重複して発生させないようにす

るためである。10台のコンピュータを使用するときは図4のようになる。十分に位相差のある乱数を各コンピュータに割り当てるためには、各コンピュータで使う乱数の初期値を求める必要がある。1台目ののコンピュータに長時間計算させれば2台目のコンピュータの初期値は求めることができる。しかしこれにはかなりの時間が必要であり、高速化を目的とする本研究にそぐわない。よって、計算によって、十分位相差のある値を求める。

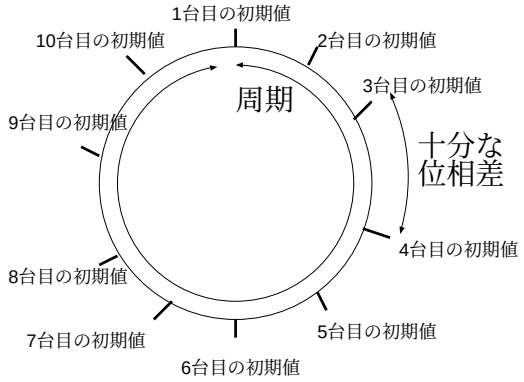

図 4: 10台のコンピュータを使用する場合

6.1 位相をずらして初期値を与える。

乱数列の同一周期上で、先の項を求める方法として、参考文献[1] より抜粋した例を述べることにより説明する。

例として$f(z) = 1 + z^{489} + z^{521}$の場合に、$a_{52100}$を$a_0, a_1, \cdots, a_{520}$の1次結合で表す。(ここでは、(mod 2)という添え書きを省略する。)

$$\begin{aligned} a_t &= a_{t-489} \oplus a_{t-521} \\ &= a_{(t-489)-489} \oplus a_{(t-489)-521} \\ &\oplus a_{(t-521)-489} \oplus a_{(t-521)-521} \\ &= a_{t-2*489} \oplus a_{t-2*521} \end{aligned} \tag{8}$$

この作業を繰り返すと、

$$\begin{gathered} a_{t-2*489} \oplus a_{t-2*521} \\ \downarrow \\ a_{t-4*489} \oplus a_{t-4*521} \\ \downarrow \\ a_{t-8*489} \oplus a_{t-8*521} \end{gathered} \tag{9}$$

となり、添字を減らす数値が2倍ずつ増えていく。

a_{52100}を求めたいので、$a_{t-n*489} \oplus a_{t-n*521}$ の$n*521$が52100を越えてはいけない。

$$n \leq 52100/521 = 100$$

ここでnは２のべき乗になっているので、100以下で最大の２のべき乗は64であるから

$$\begin{aligned} a_{52100} &= a_{52100-64*489} \oplus a_{52100-64*521} \\ &= a_{20804} \oplus a_{18756} \end{aligned} \tag{10}$$

次にa_{20804}の添字を減らす場合、20804/521=39.93なので、

先ほどと同様に39以下で最大の2のべき乗である32を使って、

$$\begin{aligned} a_{20804} &= a_{20804-32*489} \oplus a_{20804-32*521} \\ &= a_{5156} \oplus a_{4132} \end{aligned} \tag{11}$$

また18756/521=36、36以下で最大の2のべき乗は32なので、

$$\begin{aligned} a_{18756} &= a_{18756-32*489} \oplus a_{18756-4*521} \\ &= a_{3108} \oplus a_{2084} \end{aligned} \tag{12}$$

以上のことを繰り返すと、次式が得られる。

$$\begin{aligned} a_{52100} &= a_0 \oplus a_{14} \oplus a_{28} \oplus a_{46} \oplus a_{110} \\ &\oplus a_{128} \oplus a_{174} \oplus a_{202} \oplus a_{220} \oplus a_{266} \\ &\oplus a_{467} \oplus a_{485} \oplus a_{499} \oplus a_{503} \oplus a_{517} \end{aligned} \tag{13}$$

この表現は、再帰を使うプログラムを書くことによって簡単に求めることができる。説明は3項GFSR法で行ったが、本研究では5項GFSR法を用いた方法のプログラムを作成した。作成したプログラムを実行すると、求める項の番号(添字)が100万くらいになると、その値を求めるのに時間がかかり過ぎる。実際に計算をさせたところ、一日コンピュータによって計算をさせても計算が終わらなかった。3項GFSR法を用いて同じようなプログラムによって計算させた場合、１億ほどの番号(添字)でも数十秒ほどで解が求まる。原因としては5項GFSR法のプログラムは3項GFSR法のプログラムに比べ、コンピュータのメモリとのやりとりが多過ぎるためだと考えられる。この問題の改善のために、別案を考えた。

7 行列の積によって位相をずらした初期値を計算する

この方法は、乱数列を行列を用いて表示し、その行列の積を利用することによって位相差のある初期値を計算する方法である。付録にプログラムを載せるが、この方法は、6.1の方法のプログラムよりも、再帰関数の呼出し回数を少なくすることができるため、プログラムの実行時間も短縮できると考えられる。

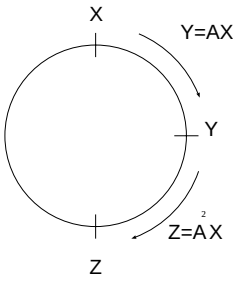

図 5: 行列の積によって先の項を求める

7.1 理論

この方法は、乱数列の周期の一部を計算し、計算した乱数を行列表示し、その行列の2乗を繰り返すことによって、先の項を速く求めようとするものである。わかりやすく図で説明すると、図5のようなイメージである。

7.2 証明

この理論を簡単な例によって証明する。
漸化式、

$$a_t = a_{t-3} \oplus a_{t-5} \tag{14}$$

を例とすると、6.1より、

$$\begin{aligned} a_{40} &= a_{40-8*3} \oplus a_{40-8*5} \\ &\cdot \\ &\cdot \\ &= a_1 \oplus a_3 \oplus a_4 \end{aligned}$$

となる。この漸化式ではa_0からa_4の5つの初期値を必要とし、求める項は、5つの初期値のうち、使うものを1、使わないものを0と表示するようにする。つまり、a_{40}は、[01011]と表示される。5つの初期値を使い、行列の2乗により、先の項を求めるのだから、計算するためには5×5の行列が必要となる。a_{40}は計算したので、$a_{41}, a_{42}, a_{43}, a_{44}$は、漸化式を計算して求めれば良い。こうして次の表現が得られる。

$$\begin{pmatrix} a_{40} \\ a_{41} \\ a_{42} \\ a_{43} \\ a_{44} \end{pmatrix} = A \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} \tag{15}$$

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \tag{16}$$

行列(16)を2乗することによって、先の項を求めるのが本研究の目的である。行列(16)を２乗すれば、a_{80}が求められるはずである。しかし、本研究における演算は繰り上がりなしの排他的論理和を用いた論理演算であるので、行列の２乗を求める場合でも排他的論理和を用いる。よって行列(16)の２乗は行列(17)のようになる。

$$A^2 = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \end{pmatrix} \tag{17}$$

行列(17)より、a_{80}は[11000]と表示することができる。このことから、a_{80}は、

$$a_{80} = a_0 \oplus a_1 \tag{18}$$

のように表すことができる。実際に6.1によりa_{80} を計算すると、

$$\begin{aligned} a_{80} &= a_{80-16*3} \oplus a_{80-16*5} \\ &\cdot \\ &\cdot \\ &= a_0 \oplus a_1 \end{aligned}$$

となり、たしかに式(18)と同じになる。作成したプログラムによって小さい乱数を発生させ、確認したところ、たしかに同じ乱数列になることが確認された。理論とプログラムの両方で正しく乱数を発生させられることが確認できたので、この方法の信頼性が保証されたといっていいだろう。

7.3 乱数と乱数の間の相関

乱数と乱数の間の相関を調べるが、これは共に5項GFSR法によって生成された乱数列の中から、十分に位相をずらした2つの乱数の相関を調べる。計算方法は式(19)に乱数列を代入して関数の値を調べる。この関数はある乱数列$< x_n >$と、その位相をτだけずらした乱数列$< x_{n+\tau} >$の相関を調べるものである。関数中のx_nはn番目の乱数、N は相関を調べる乱数の個数、$\bar{x}$は乱数列の一周期の平均値である。

$$P_{xx}(\tau) = \frac{1}{N} \sum_{n=0}^{N-1} (x_n - \bar{x})(x_{n+\tau} - \bar{x}) \tag{19}$$

自己相関の大きさを測る指標として関数$P_{xx}(\tau)$を考えたが、この値では、はっきりと相関の大きさを比べることはできないので$P_{xx}(\tau)$ を$P_{xx}(0)$で割った値、つまり、式(20)で得られる$r(\tau)$の値を考える。τを横軸にとり、$r_{xx}(\tau)$を縦軸にとり、少しずつτを大きくしていきτが大きくなると$r_{xx}(\tau)$ (相関係数)がどのような値をとるのかをグラフに点をプロットして調べる。

$$r(\tau) = \frac{P_{xx}(\tau)}{P_{xx}(0)} \tag{20}$$

8 おわりに

本研究において発生させた乱数において、相関が強い箇所は見られなかった。このことから本研究の発生方法は信頼性が保証されたといっていいだろう。本研究では32ビットに限った乱数発生法であったが、より性能のよいコンピュータを使用し、64ビットで計算できれば、さらに長い乱数列を得ることが可能である。

参考文献

[1] 伏見正則：乱数，東京大学出版会, (1989).

[2] 高井 宏和：並列計算のための乱数生成法，南山大学卒業論文,(2005).

[3] 日本工業規格: 乱数発生及びランダム化の手順，Z 9031,pp.1-5(2001).