2. GERAÇÃO DE NÚMEROS PSEUDO-RANDÔMICOS

2.1. INTRODUÇÃO

Para a utilização de seqüências de números randômicos precisamos dispor de meios eficientes de geração destas seqüências, e de um método para acesso rápido aos números. Vimos, na descrição sobre a história do assunto feita acima, que diversos meios foram tentados, e os números eram estocados em tabelas para uso posterior. Com o advento da era da informática é natural procurarmos utilizar os recursos computacionais para a geração de seqüências de números randômicos.

Chamamos de *seqüências de números pseudo-aleatórios* ou *pseudo-randômicos* às seqüências de números gerados através de programas computacionais. O termo *pseudo* é aqui introduzido porque, para construir tais programas, é necessário impor regras, e quando se impõe regras não podemos mais falar em aleatoriedade. Em uma seqüência de números pseudo-aleatórios sabemos que número segue outro, de acordo com as regras que foram implementadas no programa. Apesar disso tais processos têm sido muito úteis nas aplicações.

Existem vários métodos para a geração de números pseudo-aleatórios através de sistemas computacionais. As seqüências obtidas com esses métodos são submetidas a vários tipos de testes, que examinam as características da seqüência sob o ponto de vista de sua distribuição em um domínio. Os testes são escolhidos de acordo com a finalidade a que se destina a seqüência, pois um gerador pode gerar seqüências mais aleatórias em alguns aspectos do que em outros.

Um mecanismo para geração computacional de números pseudo-aleatórios foi proposto por Von Neumann e M. Metropolis em 1946. O método consistia em elevar ao quadrado um número previamente dado e extrair os dígitos centrais para obter o próximo número. Por exemplo, dado o número 6139, consideramos seu quadrado 37687321, e dele tomamos o número 6873. Repetindo-se sucessivamente esse processo, se obtêm uma seqüência de números pseudo-randômicos.

O método dos quadrados-médios de Von Neumann é bom para experimentos não muito precisos, mas para experimentos exigentes apresenta problemas. Dependendo da semente considerada, isto é, do primeiro número tomado para a seqüência, a seqüência gerada pode ter um ciclo muito pequeno. Por exemplo, se a semente for zero, a seqüência se degenera em zero. Outro exemplo: se tomarmos a semente como 4100, teremos a seqüência 8100, 6100, 2100, 4100, 8100, 6100, 2100, ... que tem ciclo muito pequeno, fazendo com que a

seqüência fique muito previsível. É possível obter seqüências longas por esse método que passam por testes de randomicidade, mas é arriscado usar esse método sem uma análise bem acurada e também é bem trabalhoso.

Da mesma forma que o método dos quadrados-médios de Von Neumann também foi surgerido o método dos produtos médios, que consistia em tomarmos dois inteiros, x_1 e x_2 como sementes, e múltiplicá-los, obtendo um valor u, e, então os dígitos médios de u constituirão x_3. Da mesma forma obtemos x_n com x_{n-1} e x_{n-2} com $3 \le n$ inteiro.

Estudaremos a seguir diversos métodos de geração computacional de seqüências pseudo-aleatórias. Começaremos com o *método de congruência linear,* que é o mais simples e utilizado .

2.2. O MÉTODO DE CONGRUÊNCIA LINEAR

2.2.1. DEFINIÇÃO

O *método de congruência linear* foi introduzido por D. H. Lehmer em 1948 e é um dos mais simples e eficientes métodos de geração computacional de seqüências de números pseudo-aleatórios.

O método consiste em tomarmos três constantes m, a e c, escolhidas previamente, com as condições $0 \le a$, $a < m$, $0 \le c$, $c < m$. As constantes m, a e c são chamadas respectivamente *módulo*, *multiplicador* e *incremento*. Também é escolhido um valor x_0, chamado *semente*, que é o primeiro número da seqüência. x_0 deve ser tal que $0 \le x_0$ e $x_0 < m$.

A seqüência de números pseudo-aleatórios é definida recursivamente pela equação de congruência dada abaixo:

$$x_{n+1} = (a\,x_n + c) \bmod m \qquad 0 \le x_{n+1} \quad \text{e} \quad x_{n+1} < m \quad \text{para } n = 0, 1, 2, 3, \ldots$$

Os exemplos abaixo ilustram o método de congruência linear.

Exemplo 1 : Com m = 12, a = 3, c = 4 e $x_0 = 5$, obtemos:

```
> m:=12:   a:=3:   c:=4:
```
Introduzindo as constantes

```
> x[0]:=5:
```
Definindo a semente

```
> x[1]:=(a*x[0] + c) mod (m);
```
Gerando o primeiro elemento

$x_1 := 7$

```
> x[2]:=(a*x[1] + c) mod (m);
```
Gerando o segundo elemento

$x_2 := 1$

```
> x[3]:=(a*x[2] + c) mod (m);
```
Gerando o terceiro elemento

$x_3 := 7$

```
> x[4]:=(a*x[3] + c) mod (m);
```
Gerando o quarto elemento

$x_4 := 1$

```
>
```

E assim obtemos uma seqüência com ciclo de comprimento dois:

5, 7, 1, 7, 1, 7, 1, ...

Exemplo 2 : Procedendo da mesma forma com m = 12, a = 1, c = 5 e $x_0 = 5$ obtemos a seqüência

5, 10, 3, 8, 1, 6, 11, 4, 9, 2, 7, 0, 5, ...

Observamos nesses exemplos que *existem ciclos* que se repetem infinitamente. Isto vai ocorrer em todas as seqüências geradas por esse método, pois existem no máximo *m* números distintos entre si. O comprimento do ciclo é chamado ***período***.

A seqüência do primeiro exemplo tem período 2, já a seqüência do segundo exemplo tem período 12. Escolhido um módulo *m*, desejamos obter seqüências que tenham o maior período possível. Não é para quaisquer *a*, *c*, e *m* que teremos um bom gerador. Veremos como escolher o multiplicador e o incremento para que a seqüência seja de período máximo.

2.2.2. A ESCOLHA DO MÓDULO

Para escolhermos o módulo, queremos que o valor de *m* seja suficientemente grande para que o período também seja suficientemente grande. O valor de *m* depende assim da aplicação que se esteja fazendo da seqüência. De modo geral se pode afirmar que quanto maior *m*, menos previsível é a seqüência.

Assim, uma pessoa que quer gerar uma seqüência de *zeros* e *uns* provavelmente não escolherá $m = 2$, pois o ciclo teria comprimento no máximo 2. Uma seqüência de *zeros* e *uns* com $m = 2$ teria uma das quatro formas abaixo, muito previsíveis:

0, 0, 0, 0, 0, 0, 0, 0, ... ;

0, 1, 0, 1, 0, 1, 0, 1, ... ;
1, 0, 1, 0, 1, 0, 1, 0, ... ;
1, 1, 1, 1, 1, 1, 1, 1, ... ;

Quando se deseja uma seqüência aleatória de *zeros* e *uns* pode-se começar gerando uma seqüência pseudo-aleatória com m suficientemente grande, e depois se modifica a seqüência para que ela atinja a forma desejada. Existem eficientes métodos de modificação de seqüências uniformes que serão estudados mais abaixo.

2.2.3. A ESCOLHA DO MULTIPLICADOR E DO INCREMENTO

Escolhido o valor do módulo m, desejamos que a seqüência de números pseudo-aleatórios tenha período de comprimento máximo. Para isso é preciso escolher convenientemente os valores do multiplicador e do incremento.

O método para a escolha das constantes a (multiplicador) e c (incremento) para que a seqüência tenha comprimento máximo é dada pelo teorema abaixo [*Seminumerical Algorithms*, Knuth 15-18]:

***Teorema* 1:** Uma seqüência de congruência linear tem período máximo m se, e somente se, estiverem satisfeitas as seguintes condições:

i) c e m são relativamente primos;
ii) a é congruente a 1 módulo p para todo primo p que divide m;
iii) a é congruente a 1 módulo 4 se 4 divide m.

Para se demostrar o teorema será necessário utilizar alguns resultados adicionais que são vistos nos lemas abaixo:

Lema 1: Sejam p um número primo e e um inteiro positivo tal que $2 < p^e$. Se

$$x = 1 \bmod p^e \quad \text{e} \quad x \neq 1 \bmod p^{(e+1)}$$

então

$$x^p = 1 \bmod p^{(e+1)} \quad \text{e} \quad x^p \neq 1 \bmod p^{(e+2)}$$

Demonstração: Temos $x = 1 + q\,p^e$ para algum inteiro q que não é um múltiplo de p. Pela fórmula binomial temos:

$$x^p = (1 + q\,p^e)^q =$$

$$1 + \text{binomial}(p, 1)\,q\,p^e + \ldots + \text{binomial}(p, p-1)\,q^{(p-1)}\,p^{((p-1)e)} + q^p\,p^{(pe)} =$$

$$= 1 + q\,p^{(e+1)}\left(1 + \frac{\text{binomial}(p, 1)\,q\,p^e}{p} + \frac{\text{binomial}(p, 3)\,q^2\,p^{(2e)}}{p} + \ldots + \right.$$

$$+ \frac{\text{binomial}(p, p)\, q^{(p-1)}\, p^{((p-1)e)}}{p})$$

A quantidade entre parênteses é um inteiro, e de fato, todo termo entre os parênteses é um múltiplo de p exceto o primeiro termo . Para $1 < k$ e $k < p$, o binomial(p, k) é divisível por p, então $\frac{\text{binomial}(p, k)\, q^{(k-1)}\, p^{((k-1)e)}}{p}$ é divisível por $p^{((k-1)e)}$. Por outro lado, o último termo também é divisível por p, pois ele é igual a $q^{(p-1)}\, p^{((p-1)e-1)}$. Como $2 < p^e$ segue que $1 < (p-1)\, e$ e p divide o último termo. Portanto, existe um número inteiro n tal que $x^p = 1 + q\, p^{(e+1)}\, (1 + n\, p)$. Pondo $q_1 = q\, (1 + n\, p)$, temos $x^p = 1 + q_1\, p^{(e+1)}$, onde q_1 é um inteiro não divisível por p, e isto completa a demonstração deste lema.

Lema 2: Seja $m = {p_1}^{e_1} \ldots {p_t}^{e_t}$ a decomposição de m em fatores primos. O comprimento λ do período da seqüência de congruência linear determinada por (x_0, a, c, m) é o mínimo multiplo comum dos comprimentos λ_j dos períodos das seqüências de congruência linear $(x_0 \bmod {p_j}^{e_j}, a \bmod {p_j}^{e_j}, c \bmod {p_j}^{e_j}, {p_j}^{e_j})$, $1 \le j$ e $j \le t$.

Demonstração: Por indução sobre t, é suficiente provar que se r e s são relativamente primos, o comprimento λ da seqüência de congruência linear determinada por $(x_0, a, c, r\, s)$ é o mínimo múltiplo comum (mmc) dos comprimentos λ_1, λ_2 dos períodos das seqüências $(x_0 \bmod r, a \bmod r, c \bmod r, r)$ e $(x_0 \bmod s, a \bmod s, c \bmod s, s)$. Observamos que se os elementos daquelas três seqüências são denotadas por x_n, y_n e z_n, respectivamente, teremos:

$$y_n = x_n \bmod r \qquad \text{e} \qquad z_n = x_n \bmod s \text{ para todo } 0 \le n.$$

Segue que $x_n = x_k$ se, e somente se, $y_n = y_k$ e $z_n = z_k$. (*)

Seja λ_0=mmc(λ_1, λ_2). Queremos provar que $\lambda = \lambda_0$. Como $x_0 = x_\lambda$, vem $x_0 \bmod r = x_\lambda \bmod r$, ou $y_0 = y_\lambda$. Daí λ é um múltiplo de λ_1. Do mesmo modo, λ é um múltiplo de λ_2 . Temos então $\lambda_0 \le \lambda$.

Por outro lado, $y_0 = y_{\lambda_0}$ e $z_0 = z_{\lambda_0}$. Usando (*) temos $x_0 = x_{\lambda_0}$. Daí λ_0 é um múltiplo de λ. Segue que $\lambda \le \lambda_0$. Provamos que $\lambda_0 = \lambda$. Isto termina a demonstração do Lema.

Agora estamos preparados para provar o teorema. Em virtude do lema 2, é suficiente provar o teorema quando m é uma potência de um número primo. De fato,

${p_1}^{e_1} \ldots {p_t}^{e_t} = \lambda = \text{mmc}(\lambda_1, \ldots, \lambda_t) <= \lambda_1 \ldots \lambda_t <= {p_1}^{e_1} \ldots {p_t}^{e_t}$

é verdadeiro se, e somente se, $\lambda_j = {p_j}^{e_j}$ para $1 \le j$ e $j \le t$.

Desta forma suponhamos que $m = p^e$, onde p é primo e e é um inteiro positivo.

O teorema é obviamente verdadeiro quando $a = 1$, então podemos tomar $1 < a$. Observemos que o período não depende da semente x_0. De fato, o período é m se, e somente se, cada inteiro x tal que $0 \le x$ e $x < m$ aparece na seqüência exatamente uma vez. Vamos então assumir, sem perda de generalidade, que $x_0 = 0$. Notemos que

$$x_n = a^n x_0 + (a^{(n-1)} + a^{(n-2)} + ... + a + 1)c$$

donde

$$x_n = \frac{(a^n - 1)\, c}{a - 1} \bmod m \qquad (1)$$

Se c não é relativamente primo com m, x_n nunca pode ser igual a 1, então a condição (i) do Teorema 1 é necessária para que o período seja máximo. O período é m se e somente se o menor valor positivo de n para o qual $x_n = x_0 = 0$ é $n = m$. Por (1) e pela condição (i), nosso teorema reduz-se à demonstração do seguinte fato:

Lema 3: Assume-se que $1 < a$ e $a < p^e$, onde p é primo. Se λ é o menor inteiro positivo para o qual $\frac{a^\lambda - 1}{a - 1} = 0 \bmod p^e$, então

$\lambda = p^e$ se, e somente se, $a = 1 \bmod p$ quando $2 < p$ e $a = 1 \bmod 4$ quando $p = 2$.

Demonstação: Suponhamos $\lambda = p^e$. Se $a \neq 1 \bmod p$, então $\frac{a^n - 1}{a - 1} = 0 \bmod p^e$ se e somente se $a^n - 1 = 0 \bmod p^e$. A condição $a^{(p^e)} - 1 = 0 \bmod p^e$ implica $a^{(p^e)} = 1 \bmod p$. Em virtude do Teorema de Fermat, $a^{(p^e)} = a \bmod p$. Portanto, $a \neq 1 \bmod p$ conduz a uma contradição. Por outro lado, suponhamos que $p = 2$ e $a = 3 \bmod 4$. Temos $\frac{a^{(2^{(e-1)})} - 1}{a - 1} = 0 \bmod 2^e$ (confira observação abaixo). Isto é uma contradição. Estes argumentos mostram que é necessário ter $a = 1 + q\, p^f$, quando $2 < p^f$ e q não é um múltiplo de p.

Vamos demonstrar a recíproca. Aplicando repetidamente o Lema 1, temos

$$a^{(p^g)} = 1 \bmod p^{(f+g)} \quad \text{e} \quad a^{(p^g)} \neq 1 \bmod p^{(f+g+1)}$$

e dessa forma

$$\frac{a^{(p^g)} - 1}{a - 1} = 0 \bmod p^g \quad \text{e} \quad \frac{a^{(p^g)} - 1}{a - 1} \neq 0 \bmod p^{(g+1)} \qquad (2)$$

Em particular, $\frac{a^{(p^e)}-1}{a-1} = 0 \bmod p^e$, donde $\frac{(a^{(p^e)}-1)\,c}{a-1} = 0 \bmod p^e$. Isto significa que p^e é um múltiplo do período λ. Então $\lambda = p^g$ para algum g. Mas (2) mostra que $g = e$. Logo $\lambda = p^e$, terminando a demonstração do Teorema. ■

Com as regras do teorema acima podemos gerar seqüências de números pseudo-aleatórios com ciclos de comprimento máximo.

Corolário 1: Sejam $p_1, p_2, ..., p_n$ os primos que dividem m e $M = p_1\, p_2 ... p_n$. O multiplicador é da forma $a = k\,M + 1$ onde $k = 0, 1, 2, ...$ se m não for relativamente primo com 4 ou $a = k\,2\,M + 1$, onde $k = 0, 1, 2, ...$ se m for relativamente primo com 4.

Demonstração:

Seja m o módulo e a o multiplicador. Se $m = p_1^{\alpha_1}\, p_2^{\alpha_2} ... p_s^{\alpha_s}$, com $p_1 < p_2 < ... < p_s$. Se 4 não divide m, então (se $p_1 = 2$ então $\alpha_1 = 1$) definimos $M = p_1\, p_2 ... p_s$.

As seguintes condições são equivalentes:

(j) $a - 1$ é multiplo de p para todo primo p que divide m
(jj) $a = k\,M + 1$ onde $k = 0, 1, 2, ...$

Suponhamos que vale (j), então p_i divide $a - 1$ para $1 \le i$ e $i \le s$ logo M divide $a - 1$, então $a - 1 = k\,M$, então $a = k\,M + 1$, onde $k = 0, 1, 2, ...$ Conclui-se que vale (jj).

Suponhamos que vale (jj). Então $a - 1 = k\,M$, e como p_i divide M , temos p_i divide $a - 1$, com $1 \le i$ e $i \le s$. Conclui-se então que vale (j).

Agora se 4 divide m, então definimos $m = 4\, p_2\, p_3 ... p_s$ e então as seguintes condições são equivalentes:
(j) $a - 1$ é multiplo de p para todo primo p que divide m e $a - 1$ é multiplo de 4.
(jj) $a = k\,M + 1$, onde $k = 0, 1, 2, 3, ...$

Suponhamos que vale (j). Então $2\, p_2\, p_3 ... p_s$ divide $a - 1$, (onde $2 < p_i$ se $1 < i$), pois p divide $a - 1$ para todo primo p que divide m. Como 4 divide $a - 1$, tem-se que $4\, p_2\, p_3 ... p_s$ divide $a - 1$, ou M divide $a - 1$, daí $a - 1 = k\,M$ ou $a = k\,M + 1$, onde $k = 0, 1, 2, ...$ e portanto vale (jj).

Suponhamos que vale (jj). Então $a - 1 = k\,M$, e 4 divide $a - 1$ e p divide $a - 1$ para todo primo p que divide m. Logo vale (j).■

Corolário 2: Sejam $p_1, p_2, ..., p_n$ os primos que dividem m e $M = p_1\, p_2 ... p_n$. O incremento é da forma $c = k\,M + r_i$ onde $k = 0, 1, 2, ...$ e o conjunto r_i é o *Sistema Reduzido de*

Resíduos módulo M. Obs: para cada k associa-se todos os r_i.

Demonstração:

Seja $m = p_1^{\alpha_1} p_2^{\alpha_2} \ldots p_s^{\alpha_s}$ a decomposição canônica de m. Definimos $M = p_1 p_2 \ldots p_s$

Então são equivalentes as seguintes condições:

(j) $1 \le c$ é relativamente primo com m.

(jj) $c = k M + r$, onde $k = 0, 1, 2, 3, \ldots$ e r é um inteiro de $[\ 1, M\ )$ relativamente primo com M.

Suponhamos que vale (jj), e seja $c = k M + r$ nas condições acima. Seja p um primo divisor comum de c e m. Se p divide m então p divide M, então p divide $c - k M$ e daí p divide r. Logo p divide r e p divide M o que é absurdo, logo c e m são relativamente primos entre sí e vale (j).

Suponhamos que vale (j), dado c relativamente primo com m, divida c por M, então $c = k M + r$, onde $0 \le r$ e $r < M$. Note que $0 \le k$. Se p divide r e p divide M entao p divide c e p divide m, o que não ocorre se p for primo. Logo mdc$(r, m) = 1$.■

Utilizando os recursos de programação do Maple construímos procedimentos para implementar os corolários acima. Escolhido o valor do módulo, nossos procedimentos calculam listas de multiplicadores e incrementos que permitem obter seqüências pseudo-aleatórias com a propriedade de período máximo. No *Anexo* 1 descrevemos esses procedimentos, que estão também disponíveis em disquete.

Exemplo 3: Vamos gerar uma seqüência de números pseudo-aleatórios constituída com os números inteiros não negativos com quatro ou menos casas. Para isso escolhemos $m = 10000$. Observe que

```
> ifactor(10000);
```

$$(2)^4 (5)^4$$

Os números primos p que dividem m são 2 e 5, e 4 divide m. Em vista disso $a - 1$ deve ser divisível por 2, 4 e 5. Tomamos como exemplo $a = 21$.

Como c e m devem ser relativamente primos, tomamos, por exemplo, $c = 11$. Escolhemos ainda $x_1 = 3247$ como o valor da semente.

Definimos a seqüência x_i de números pseudo-aleatórios executando os comandos:

```
> x[1]:=3247: m:=10000: a:=21: c:=11:
```
definindo as constantes

```
> for i to 10001 do
> x[i+1]:=(a*x[i]+c)mod(m);
> od:
```
gerando 1001 números

O procedimento acima estocou os números gerados em um vetor x_i , $1 \le i$, $i \le 10001$. Vejamos os vinte primeiros números dessa seqüência:

```
> for i to 10 do
> print(x[i],`     `,x[i+10]);
> od:
```

Obs: Os numeros estão ordenados em colunas.

3247, , 6657
8198, , 9808
2169, , 5979
5560, , 5570
6771, , 6981
2202, , 6612
6253, , 8863
1324, , 6134
7815, , 8825
4126, , 5336

Para ver o sexagésimo terceiro número da seqüência, pedimos:

```
> x[63];
```

5029

O décimo-milionésimo primeiro é:

```
> x[10001];
```

3247

Para calcular o período dessa seqüência podemos utilizar o seguinte procedimento:

```
> p:=1:
> i:=2:
> while x[1]<>x[i] do
> p:=p+1;
> i:=i+1;
> od:
> print(`período = `.p);
```

período = 10000

```
>
```

Assim, o comprimento do ciclo da seqüência calculada é 10000. Como vemos, o período do gerador é igual ao valor do módulo, o que já era previsto, pois escolhemos o valor do multiplicador e o valor do incremento de acordo com as regras propostas pelo Teorema 1.

2.2.4. O CASO PURO MULTIPLICATIVO

O gerador de congruência linear puramente multiplicativo é um caso particular do método de congruência linear, no qual o incremento é zero. Nesse caso a seqüência obedece à equação de congruência linear abaixo:

$$x_{n+1} = a\, x_n \bmod m \qquad 0 < a \text{ e } a < m \quad 0 < x_n \text{ e } x_n < m$$

Observe que se aparecer o número zero na seqüência ela se degenera em zeros. Portanto o maior período que se pode conseguir com esse método é $m - 1$.

Usando este método, para conseguirmos a seqüência com comprimento de período máximo usamos a seguinte regra:

i) x_0(semente) é relativamente primo com m;
ii) a é um elemento primitivo módulo m.

Nós obtemos o período de comprimento $m - 1$ se m for primo.

Teorema 2: a é um elemento primitivo módulo p^f se, e somente se,

i) $p^f = 2$, a é ímpar; ou $p^f = 4$, $a \bmod 4 = 3$; ou $p^f = 8$, $a \bmod 8 = 3, 5, 7$; ou $p = 2$, $4 \le f$, $a \bmod 8 = 3$ ou 5; ou

ii) p é ímpar, $f = 1$, $a \neq 0 \bmod p$, e $a^{\left(\frac{p-1}{q}\right)} \neq 1 \bmod p$ para algum divisor primo q de $p - 1$; ou

iii) p é ímpar, $1 < f$, a satisfaz (ii), e $a^{(p-1)} \neq 1 \bmod p^2$.

Para muitas aplicações o gerador puro multiplicativo é usado com o módulo m igual ao Primo de Mersene $M_{31} = 2^{31} - 1$.

O método puro multiplicativo é um método muito satisfatório, pois as seqüências de números gerados por ele passam em quase todos os testes de randomicidade. O gerador do Maple V utiliza este método para gerar os números pseudo-randômicos. Ele também usa $m = 999999999989$ e $a = 427419669081$. Portanto o período máximo do gerador do Maple V é 999999999988, gerando números de [1 .. 999999999989].

2.3. O MÉTODO DE MacLAREN E MARSAGLIA

Para se conseguir uma seqüência de números pseudo-randômicos podemos combinar duas seqüências de números geradas por geradores distintos. Usando essa idéia, MacLaren e Marsaglia desenvolveram o seguinte método de geração de números pseudo-randômicos:

1º passo) geramos duas seqüências X_n e Y_n de números aleatórios;

2º passo) construímos índices j ⇦ floor$\left(\frac{k\,Y_n}{m}\right)$, onde m é o módulo do gerador usado para gerar a seqüência Y_n e k é uma constante (inteiro positiva) e j está entre [0 , k);

3º passo) construímos a seqüência V_j, onde V_j ⇦ X_n .

F. Gebhardt [*Math. Comp.* 21, 708 - 709] mostrou em 1967 que seqüências satisfatoriamente aleatórias são produzidas a partir deste método, mesmo quando as seqüências X e Y não são tão satisfatórias (como, por exemplo, as seqüências geradas pelo método de Fibonacci, que estudaremos mais abaixo) .

Exemplo 4:

Construímos um procedimento para gerar a seqüência V descrita acima.

Sintaxe:

MacLaren (v , k) :
v é o nome da seqüência a ser gerada
k é a quantidade de números da seqüência

```
>
> MacLaren:=proc(v::string,k::nonnegint)
> local i,X,Y:
> global rand01,rand02:
>
> rand01:=proc()                    Gerador para a seqüência X
> global semt1:
> if not assigned(semt1) then semt1:=40 fi;
> semt1:=modp(semt1*21+7,1000):
> RETURN(semt1):
> end;
>
> rand02:=proc()                    Gerador para a seqüência Y
> global semt2:
> if not assigned(semt2) then semt2:=467 fi;
```

```
> semt2:=(semt2*81+11)mod(1000):
> RETURN(semt2):
> end:
>
> for i from 0 to k-1 do
> X[i]:=rand01():
> Y[i]:=floor(k*rand02()/1000)+1:
> od:
> v:=array(1..k):
> for i to k do
> v[i]:=X[i-1]:
> od:
>
> for i from 0 to k-1 do
> v[Y[i]]:=X[i]:
> od:
> RETURN(evalm(v)):
> end:
```

```
> semt1:=40:
```
Inicializa-se a semente do primeiro gerador

```
> semt2:=467:
```
Inicializa-se a semente do segundo gerador

A seqüência V é obtida executando o procedimento abaixo:

```
> MacLaren(K,170);
```

[197, 11, 653, 658, 266, 45, 593, 529, 893, 533, 967, 240, 473, 628, 195, 102, 413, 935, 737, 130, 737, 674, 745, 130, 565, 872, 976, 872, 940, 825, 507, 986, 628, 568, 635, 437, 319, 504, 189, 508, 830, 824, 448, 538, 572, 412, 525, 846, 686, 328, 47, 441, 268, 19, 207, 277, 208, 416, 415, 610, 817, 164, 451, 342, 610, 412, 979, 164, 849, 980, 305, 334, 977, 151, 47, 406, 169, 556, 161, 803, 416, 310, 112, 679, 864, 492, 683, 5, 653, 623, 127, 747, 200, 817, 155, 262, 503, 116, 895, 443, 882, 844, 126, 358, 90, 881, 679, 136, 593, 460, 517, 587, 301, 654, 339, 524, 980, 836, 563, 652, 437, 722, 581, 298, 836, 478, 19, 591, 870, 418, 354, 354, 358, 680, 565, 342, 298, 976, 503, 375, 760, 524, 178, 238, 314, 570, 184, 546, 473, 301, 197, 265, 556, 208, 311, 882, 529, 460, 262, 893, 829, 14, 699, 155, 745, 773, 699, 897, 413, 350]

```
>
```

2.4. OUTROS MÉTODOS DE GERAÇÃO DE PSEUDO-RANDÔMICOS

2.4.1. O MÉTODO DE FIBONACCI

Outro método para a geração de seqüências de números pseudo-randômicos é o de Fibonacci que estabelecendo *m*, inteiro positivo, x_0 e x_1 sementes menores que *m* . A seqüência de Fibonacci é dada recursivamento pela congruência:

$$x_{n+1} = (x_n + x_{n-1}) \bmod m \qquad 0 \le x_{n+1} \text{ e } x_{n+1} < m$$

Observação: As *sementes* **nunca** poderão ser ambas nulas, caso contrário a seqüência se degenera a zeros.

O período do gerador produzido a partir desse método pode ser maior que o módulo *m* do gerador.

Testes mostraram que números obtidos da recursão do gerador de Fibonacci não são sempre satisfatoriamente aleatórios.

Podemos a partir do método de Fibonacci considerar a sequinte equação, que constitui um novo método: $x_{n+1} = (x_n + x_{n-k}) \bmod m$, onde$k + 1$ é a quantidade de sementes. Se as sementes $x_0, x_1 ... x_k$ são escolhidas apropriadamente então a seqüência criada com este método provavelmente gerará bons números pseudo-randômicos. Este método foi introduzido por Green, Smith e Klem em 1959.

Quando $m = p$ é primo e $a_1, \ldots, a_k$ são multiplicadores, a seqüência definida por $x_n = a_1 x_{n-1} + \ldots + a_k x_{n-k} \bmod p$ terá período máximo $p^k - 1$, com os multiplicadores escolhidos adequadamente e os valores das sementes $x_0, \ldots, x_{k-1}$ escolhidos arbritariamente, mas não todos nulos.

As constantes $a_1, \ldots, a_k$ têm a propriedade desejada (seqüência gerada com período máximo) se, e somente se, o polinômio $f(x) = x^k - a_1 x^{(k-1)} - \ldots - a_k$ é um polinômio primitivo módulo *p*.

Para determinar se $f(x)$ é um polinômio primitivo módulo *p* o seguinte critério pode ser utilizado:

i) $-1^{(k+1)} a_k$ deve ser uma raíz primitiva módulo *p*.

ii) O polinômio x^r deve ser congruente a $(-1^{(k+1)} a_k) \bmod f(x)$ e *p*.

iii) O grau de $x^{\left(\frac{r}{q}\right)} \bmod f(x)$ usando aritmética polinomial módulo *p*, deve ser positivo, para cada divisor primo *q* de *r*.

Aqui $r = \dfrac{p^k - 1}{p - 1}$

Exemplo 5:

O gerador criado a partir do método de Fibonacci tem as sementes x_1 e x_2 e módulo *m*como as constantes e calcula *n* números e imprimindo-os.

Sintaxe:

aleafibonacci(comprimento da seqüência, semente1, semente2, módulo);

comprimento da seqüência deve ser um número inteiro e positivo;
módulo é a única constante.
semente1 e semente2 devem ser do tipo não-negativo e devem pertencer ao intervalo [0,módulo[.

```
> aleafibonacci:=proc(_n::posint,_sem1::nonnegint,
  _sem2::nonnegint,_m:posint)
> local _i,_x;
>   _x:=array(1.._n);
>   _x[1]:=_sem1;
>   _x[2]:=_sem2;
>    if (_sem1<_m) and (_sem2<_m) then
>     for _i from 2 to _n-1 do
>      _x[_i+1]:=(_x[_i]+_x[_i-1])mod(_m):
>     od:
>     RETURN(eval(_x));
>    else
>     ERROR(`semente1 e/ou semente2 maiores que o módulo`);
>    fi:
> end:
```

```
> aleafibonacci(63,3,5,10);
```

Imprime 63 números pseudo-randômicos

[3, 5, 8, 3, 1, 4, 5, 9, 4, 3, 7, 0, 7, 7, 4, 1, 5, 6, 1, 7, 8, 5, 3, 8, 1, 9, 0, 9, 9, 8, 7, 5, 2, 7,
9, 6, 5, 1, 6, 7, 3, 0, 3, 3, 6, 9, 5, 4, 9, 3, 2, 5, 7, 2, 9, 1, 0, 1, 1, 2, 3, 5, 8]

Observe que na seqüência gerada acima o período é 60, mas o módulo do gerador é 10.

```
>
```

2.4.2. O MÉTODO DE CONGRUÊNCIA QUADRÁTICA

Este método apresenta uma similaridade com o método de congruência linear pois o método é baseado a partir de uma ncia quadrática da Teoria dos Números.

A seqüência é gerada da seguinte forma:

$$x_{n+1} = (d\,x_n^{\,2} + a\,x_n + c) \bmod m$$

Teorema 3 : A seqüência de congruência quadrática terá período de comprimento máximo se, e somente se, as seguintes condições forem satisfeitas:

i) c e m são relativamente primos;
ii) d e $a-1$ são ambos múltiplos de p, para todo primo ímpar p que divide m.
iii) d é par, e $d = (a-1) \bmod 4$, se m é um múltiplo de 4; , se m é $d = (a-1) \bmod 2$ um múltiplo de 2;
iv) também $d = 0 \bmod 9$, ou $a = 1 \bmod 9$ e $c\,d = 6 \bmod 9$, se m é um multiplo de 9.

Exemplo 6:

Vamos criar o gerador para o metodo acima:

Sintaxe:

random2 (comprimento da seqüência, multiplicador1, multiplicador2, incremento, módulo);

comprimento da seqüência deve ser do tipo positivo inteiro;
multiplicador1, multiplicador2, incremento e módulo são as constantes;

A semente pode ser escolhida com o comando

```
_sement := valor;
```

Este programa imprime uma seqüência com n números aleatórios.

```
> random2:=proc(_g::posint,_a::posint,_b::posint,_c::posint,
  _m::posint)
>  local   d,k,v:
>  global  _sement:
>  d:=array(1.._g);
>  if not assigned(_sement) then _sement:=0 fi:
>  if _sement<_m then
>   for k from 0 to _g-1 do
>    v[k]:=_sement;
>    _sement:= modp(_a*_sement^2 +_b*_sement + _c, _m);
>   od:
>   for k from 0 to _g-1 do
```

```
>     d[k+1]:=v[k];
>    od:
>    RETURN(eval(d));
>   else
>    ERROR(`semente maior ou igual ao mdulo`);
>   fi:
> end:

> _sement:=31:
> random2(98,5,13,7,100);
```

Imprime 98 números pseudo-randômicos

[31, 15, 27, 3, 91, 95, 67, 23, 51, 75, 7, 43, 11, 55, 47, 63, 71, 35, 87, 83, 31, 15, 27, 3, 91, 95, 67, 23, 51, 75, 7, 43, 11, 55, 47, 63, 71, 35, 87, 83, 31, 15, 27, 3, 91, 95, 67, 23, 51, 75, 7, 43, 11, 55, 47, 63, 71, 35, 87, 83, 31, 15, 27, 3, 91, 95, 67, 23, 51, 75, 7, 43, 11, 55, 47, 63, 71, 35, 87, 83, 31, 15, 27, 3, 91, 95, 67, 23, 51, 75, 7, 43, 11, 55, 47, 63, 71, 35]

OBS: O exemplo não satisfaz as condições acima.

```
>
```