2
Método de Monte Carlo

2.1
Introdução

O método de Monte Carlo é um método estatístico utilizado em simulações estocásticas com diversas aplicações na ciência, indústria e comércio. O método foi criado na metade da década de 40 por Stanislaw Ulam [1] e John Von Neumann [2] para resolver problemas de difusão de nêutrons em Los Alamos [30] e, rapidamente se difundiu na engenharia, física e matemática.

Como exemplo, ele é utilizado no estudo do movimento de partículas microscópicas, na geração e circulação de informações em redes e no controle das chegadas e partidas de embarcações em portos movimentados. Ele está no coração de algoritmos utilizados para fazer previsões sobre processos estocásticos e, representa uma poderosa ferramenta para obter-se aproximações numéricas de funções complicadas.

O método é largamente utilizado no estudo de sistemas com um grande número de graus de liberdade, de sistemas que envolvem fluidos e que estão associados à incertezas. Muitas aplicações também podem ser encontradas na análise de risco financeiro, em computação numérica e simulação estocástica. Um clássico exemplo é o uso do método para a aproximação do cálculo de integrais multidimensionais em regiões com contornos complicados. Muitas vezes, essas integrais não possuem solução analítica conhecida ou, simplesmente, são expressões muito complicadas, cujo cálculo exigiria um alto custo computacional.

A capacidade do método de Monte Carlo em obter aproximações numéricas para problemas complexos o tornou uma importante ferramenta matemática. A maneira como é feita a implementação do método de Monte Carlo está intrinsecamente correlacionada com o problema no qual ele será aplicado. Porém, todas essas diferentes abordagens do método tendem a seguir um determinado padrão. Geralmente, deseja-se gerar realizações de uma variável aleatória (sujeita a alguma distribuição de probabilidade) e posteriormente utilizar as amostras obtidas para aproximar alguma função de interesse. Assim, o método faz uso de amostragem aleatória como ferramenta para produzir observações

[1] Stanislaw Ulam (1909 – 1984). Renomado matemático americano.
[2] John Von Neumann (1903 – 1957). Renomado matemático húngaro de etnia judaica.

PUC-Rio - Certificação Digital Nº 0921514/CA

sobre as quais se realizam inferências estatísticas para extrair informações sobre as observações.

As etapas do método de Monte Carlo são tipicamente:

1. Gerar realizações de uma variável ou vetor aleatório com uma determinada distribuição de probabilidade.

2. Realizar cálculos determinísticos usando cada uma das amostras geradas.

3. Agregar os resultados dos cálculos determinísticos no resultado final desejado (estatísticas ou aproximações da distribuição de probabilidade da resposta).

4. Fazer uma análise da convergência da resposta, ou seja, assegurar-se que os resultados são representativos e que alterando-se o número de realizações da variável ou vetor aleatório, as estatísticas da resposta permanecem dentro de uma margem de erro prescrita.

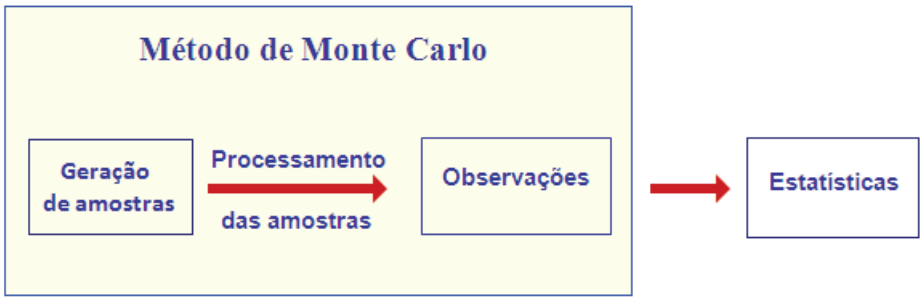

Figura 2.1: Método de Monte Carlo.

Como o método tipicamente envolve a geração de observações de alguma distribuição de probabilidade, torna-se indispensável estudar-se geração de variáveis e vetores aleatórios para poder-se trabalhar com Monte Carlo.

Este capítulo é dedicado ao estudo de geradores de amostras de variáveis aleatórias reais. Inicialmente são tratados os casos de distribuição de probabilidade uniforme (geradores baseados em congruência linear) e posteriormente os casos de outras distribuições (método da Transformada Inversa). Alguns algoritmos utilizados para a geração são descritos ao longo do capítulo e, na terceira seção é feita uma análise da influência dos valores dos parâmetros desses algoritmos. Resultados de geradores implementados em MATLAB e dois métodos de testes para a validação dos geradores são apresentados no final do capítulo.

PUC-Rio - Certificação Digital Nº 0921514/CA

2.2 Gerador de Amostras de Variáveis Aleatórias

Nesta seção da dissertação são mostrados os princípios básicos associados à geração de amostras de variáveis aleatórias reais.

Uma variável aleatória real é definida como uma função real X que associa a cada ponto amostra w, pertencente ao espaço de amostras Ω, um número $x(\mathrm{w}) \in \mathbb{R}$ [27] [16].

$$\begin{array}{rccc} X: & \Omega & \longmapsto & \mathbb{R} \\ & \mathrm{w} & \longmapsto & x(\mathrm{w}). \end{array} \tag{2-1}$$

A notação utilizada para representar variáveis aleatórias são letras maiúsculas. O mapa representado em (2-1) é esquematizado na figura (2.2).

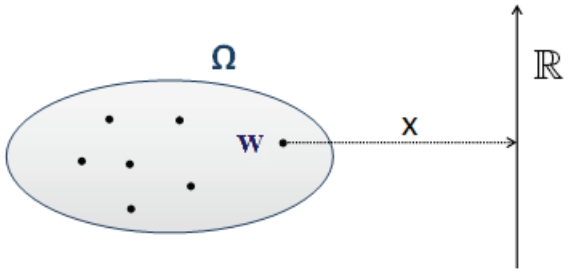

Figura 2.2: Variável aleatória (mapa de Ω para $\mathbb{R}$).

A distribuição de probabilidade, P, de uma variável aleatória, X, é definida pela função:

$$\begin{array}{rccc} P: & \mathbb{R} & \longmapsto & [0,1] \\ & x & \longmapsto & P(x) = Pr(\{\mathrm{w} \in \Omega \ : \ X(\mathrm{w}) \leq x\}) \end{array} \tag{2-2}$$

onde $Pr(\{\mathrm{w} \in \Omega \ : \ X(\mathrm{w}) \leq x\})$ indica a probabilidade de X assumir um valor menor ou igual a x. Por esse motivo, é usual utilizar a notação simplificada:

$$P(x) = Pr(X \leq x). \tag{2-3}$$

A função densidade de probabilidade, p, de uma variável aleatória real e contínua X é definida com a derivada da sua função distribuição de probabilidade, ou seja:

$$p(x) = \frac{d}{dX} P(x). \tag{2-4}$$

PUC-Rio - Certificação Digital Nº 0921514/CA

Quando deseja-se gerar amostrar de uma variável aleatória real, X, com uma densidade de probabilidade p, busca-se obter uma sequência de valores $\{x_i\}_{i\in\mathbb{N}}$, com $x_i \in \mathbb{R}, \forall i$ que representem realizações de X.

Muitos livros e artigos chamam os geradores de amostras de variáveis aleatórias frequentemente de geradores de números aleatórios (RNG-*Random Number Generator*). Abaixo segue uma lista com os principais requisitos de um RNG, espera-se que ele:

1. seja rápido, pois em suas aplicações podem ser necessário gerar milhões de amostras.

2. seja repetível, para tornar possível uma depuração do processo.

3. seja passível de análise, para poder-se estudar as propriedades da sua distribuição de probabilidade.

4. tenha um longo período.

5. seja aparentemente aleatório para a aplicação a qual é destinado.

No início do desenvolvimento de geradores, muitas técnicas híbridas de sistemas analógicos e digitais foram experimentadas. Como exemplo, tem-se os circuitos eletrônicos geradores de ruído branco. No entanto, estes métodos apresentavam muitas desvantagens: eram muito lentos, não repetíveis, tendiam a tornar-se tendenciosos e, exigiam equipamentos muito especializados. Por isso, essas técnicas foram sendo substituídas com o passar dos anos. Hoje, praticamente todos os geradores são baseados em algoritmos.

Como o computador é um dispositivo determinístico, pode parecer impossível que ele seja utilizado como um gerador. De fato, amostras geradas por computadores são obtidas através de algoritmos determinísticos. No entanto, elas parecem ser aleatórias e, para confirmar esse comportamento, deveriam ser submetidas a rigorosos testes. Muitas vezes, essas amostras são chamadas de *'pseudo-aleatórias'* [30].

Em [12], o autor Knuth descreve uma tentativa inicial de sua parte na construção de um gerador de amostras de variáveis aleatórias. Ele escreveu um algoritmo que trabalhava com números decimais de 10 dígitos, gerando uma sequência de valores $\{x_i\}_{i\in\mathbb{N}}$ a partir de um valor inicial.

O algoritmo de Knuth, é descrito a seguir. Dado o valor x_i, x_{i+1} é calculado através das seguintes etapas:

– [**K1**] Calcule: $n = \frac{x_i}{10^9}$, o dígito mais significativo de x_i. (A notação $a = \lfloor b \rfloor$ significa que a é o maior inteiro com $a \leq b$.) Os passos **K2**−**K13** são repetidos $n + 1$ vezes.

PUC-Rio - Certificação Digital Nº 0921514/CA

- [**K2**] Calcule o segundo dígito mais significativo de x_i ($m = \frac{x_i}{10^9}$), e vá para a etapa **K3+m**.
- [**K3**] Caso $x_i < 5.000.000.000$, faça $x_i \leftarrow x_i + 5.000.000.000$.
- [**K4**] Substitua o valor de x_i por $\left\lfloor \frac{x_i^2}{10^5} \right\rfloor \bmod 10^{10}$. Na operação $z = y \bmod m$, diz-se que z é congruente a y módulo m, se $z - y$ for divisível por m.
- [**K5**] Substitua o valor de x_i por $(1001001001\ x_i) \bmod 10^{10}$.
- [**K6**] Caso $x_i < 100.000.000$, faça $x_i \leftarrow x_i + 9.814.055.677$. Caso contrário, faça $x_i \leftarrow 10^{10} - x_i$.
- [**K7**] Faça $x_i \leftarrow 10^5\ (x_i \bmod 10^5) + \left\lfloor \frac{x_i}{10^5} \right\rfloor$.
- [**K8**] Substitua o valor de x_i por $(1001001001\ x_i) \bmod 10^{10}$.
- [**K9**] Diminua cada dígito diferente de zero da representação decimal de x_i de uma unidade.
- [**K10**] Caso $x_i < 10^5$, faça $x_i \leftarrow x_i^2 + 99999$. Caso contrário, faça $x_i \leftarrow x_i - 99999$.
- [**K11**] Caso $x_i > 0$ e $x_i < 10^9$, faça $x_i \leftarrow 10x_i$ e repita esse passo.
- [**K12**] Substitua o valor de x_i por $\left\lfloor \frac{x_i(x_i - l)}{10^5} \right\rfloor \bmod 10^{10}$.
- [**K13**] Caso $n > 0$, faça $n \leftarrow n - 1$ e volte ao passo **K2**. Caso $n = 0$, o algoritmo termina com o valor atual de x_i sendo o próximo valor da sequência, ou seja $x_{i+1} = x_i$.

Apesar de extenso, Knuth verificou que seu algoritmo podia convergir rapidamente para um ponto fixo. Além disso, testando-o, ele percebeu que, mesmo utilizando diferentes valores iniciais, a sequência de números começava rapidamente a se repetir.

Dessa forma, o algoritmo de Knuth é um excelente exemplo de que a uma grande complexidade em um algoritmo não garante que ele será um bom gerador de amostras de variáveis aleatórias. Muitas vezes, a complexidade esconde o comportamento de repetição dos números gerados. E assim, conclui-se que: *amostras de variáveis aleatórias não devem ser geradas com um método escolhido ao acaso* [12].

Na sequência de amostras, $x_1, x_2, \cdots, x_i, \cdots$ com $x_i \in \mathbb{R}, \forall i \in \mathbb{N}$, produzida por geradores baseados em algoritmos, o termo x_n é definido através de uma função f dependente de algumas variáveis e parâmetros:

$$x_i = f(\text{variáveis}, \text{parâmetros}) \tag{2-5}$$

Cada vez que uma amostra é requisitada ao gerador, o algoritmo calcula a função f prescrita utilizando os parâmetros e valores atuais de suas variáveis.

PUC-Rio - Certificação Digital Nº 0921514/CA

Geralmente, os parâmetros da função são valores fixos e, o que muda de amostra para amostra são as variáveis.

As variáveis podem ser quantidades externas ao algoritmo, ou valores armazenados internamente que mudam a cada iteração (nesse caso, as variáveis recebem o nome de sementes). Quando o valor da variável é atribuída por um processo externo, dificulta-se muito que a geração de uma sequência de números seja repetível. Por esta razão, a variável deve ser gerada internamente à função f. Uma exceção a essa regra é a atribuição do primeiro valor x_0.

Muitos algoritmos simplificam o cálculo dos números aleatórios, utilizando como semente para a geração o valor x_i o próprio valor de x_{i-1}.

Nas próximas seções deste capítulo, faz-se um estudo de alguns algoritmos geradores de números aleatórios posteriores ao Algoritmo Knuth apresentado. Esses geradores são chamados *Geradores de Amostras de Variáveis Aleatórias por Congruência linear* - LCRNG e garantem que as sequências por eles geradas possuem certas características desejáveis.

2.3 Geradores de Amostras de Variáveis Aleatórias Baseados em Congruência linear

Nesta seção da dissertação, analisa-se métodos para gerar uma sequência de amostras de variáveis aleatórias com uma distribuição de probabilidade uniforme. Dentre essa classe de geradores, os de congruência linear formam a base dos algoritmos mais utilizados na atualidade.

Este gerador envolve três parâmetros fixos a, c e m e também um valor inicial x_0 (a semente). A sequência de números geradas pode ser definida como:

$$x_{i+1} = (a\, x_i + c) \bmod m \qquad \forall i \in \mathbb{N} \tag{2-6}$$

Como já definido anteriormente, na operação $z = y \bmod m$, diz-se que z é congruente a y módulo m, se $z-y$ for divisível por m. Por exemplo, tomando-se: $a = 13$, $c = 0$, $m = 31$ e $x_0 = 1$, os primeiros termos da sequência definida em (2-6) valem:

$$1,\ 13,\ 14,\ 27,\ 10,\ 6,\ 16,\ 22,\ 7,\ 29,\ 5,\ 3,\ \cdots$$

Nesse exemplo, pode-se observar que os primeiros 30 termos da sequência são uma permutação dos inteiros de 1 a 30 e, a partir do trigésimo primeiro termo, a sequência começa a se repetir. Dessa forma, ela possui um período igual a $m - 1$.

Para obter-se uma sequência de valores distribuídos uniformemente no intervalo $[0, 1]$, basta dividir os valores obtidos por m. No exemplo acima, a sequência passa a ser:

PUC-Rio - Certificação Digital Nº 0921514/CA

$$0.032\,0.419,\ 0.451,\ 0.871,\ 0.322,\ 0.193,\ 0.516,\ \cdots$$

Supondo que [2]:

$$a \geq 2 \tag{2-7}$$

obtém-se uma generalização para (2-6):

$$x_{i+k} = \left(a^k x_i + \frac{(a^k - 1)c}{a - 1}\right) \bmod m \tag{2-8}$$

2.4 Parâmetros do Gerador Baseado em Congruência linear

Nesta seção da dissertação, são apresentadas algumas importantes considerações na escolha dos valores dos parâmetros a, c e m do gerador de amostras de variáveis aleatórias baseado em congruência linear.

O maior número possível de diferentes valores gerados por um LCRNG é igual a m, o módulo. Por este motivo, em qualquer aplicação real, deseja-se que m seja o maior possível, por exemplo, 2^{32}. Aliás, uma escolha para m da forma 2^k, onde k é o tamanho da palavra do computador (*wordlength* - tamanho do registrador do processador), pode ser implementada muito eficientemente [30].

Essa eficiência está associada a forma como o computador, que é um dispositivo binário, calcula a divisão na operação do módulo. O cálculo do resto de uma divisão feita em notação binária (cujo divisor seja uma potência de 2) é muito simples. Basta alinhar pela direita o divisor e o dividendo representados em notação binária e, o resultado do resto da divisão é dado pelos algarismos do dividendo que correspondem aos zeros do divisor. Na tabela (2.1) é mostrado um exemplo.

	Notação Decimal	Notação Binária
Dividendo	13	1101
Divisor	4	100
Resto		01

Tabela 2.1: Cálculo do resto de uma divisão em representação binária para divisor sendo uma potência de 2.

Vale observar que as potências de 2 em representação binária são compostas apenas por um dígito 1 seguidos de zeros, como mostrado na tabela (2.2).

Dessa forma, observa-se que quando o módulo da expressão (2-6) é uma potência de 2, a conta do resto é bastante simplificada. Nesse caso, as operações necessárias ao cálculo são:

PUC-Rio - Certificação Digital Nº 0921514/CA

Notação Decimal	Notação Binária
2^0	1
2^1	10
2^2	100
2^3	1000
2^4	10000
$\vdots$	$\vdots$

Tabela 2.2: Potências de 2 em representação decimal binária.

1. copiar o dividendo $(a\ x_i + c)$ da memória para o processador;
2. copiar o divisor (módulo m) da memória para o processador;
3. contar quantos zeros possui o divisor do bit menos significativo até encontrar o primeiro algarismo 1;
4. deslocar o dividendo para a direita de acordo com o tamanho do dividendo menos esse número de zeros e;
5. deslocar o dividendo de volta a posição original.

Este procedimento engloba cinco operações, sendo que esse número pode ainda crescer dependendo da arquitetura do processador do computador utilizado como gerador. Além disso, a cópia dos dados da memória para o processador é uma operação custosa, dado que o processador trabalha em frequências muito diferentes da frequência do barramento ao qual se conectam as memórias (limitações físicas das memórias RAM). Dessa forma, a economia da cópia de dados da memória para o processador reduziria significativamente o custo computacional do cálculo do resto de uma divisão.

Uma outra característica a ser observada é que operações no processador consomem ciclos de processamento e, portanto, tempo. Assim, quanto menos operações forem feitas com os dados no processador, em menos tempo um gerador da amostras de variáveis aleatórias será capaz se produzir uma sequência de valores.

A opção encontrada, mais simples e eficiente para minimizar a operação de cópia dos dados da memória para o processador, é utilizar como módulo uma potência de 2^k. Onde k é o tamanho da palavra do computador.

Tipicamente, as informações são armazenadas na memória do computador divididos em palavras de tamanho k. Dessa forma, por exemplo, um número de 8 bits 10110101 armazenado em uma memória com tamanho de palavra igual a 4 estaria na forma: 1011 0101. Assim, o resto da divisão desse

PUC-Rio - Certificação Digital Nº 0921514/CA

número por 2^k, onde $k = 4$, é igual a palavra menos significativa de 1011 0101, ou seja 0101 (em representação binária).

Utilizando potências de 2^k para o módulo, a operação do cálculo do resto exige muito pouco custo computacional. Neste caso, o LCRNG precisa apenas o cálculo de uma multiplicação e uma adição. Vale observar se for desejado que os números gerados pelo LCRNG estejam uniformemente distribuídos no intervalo $[0, 1]$, deverá ser realizada uma divisão por m para cada valor calculado.

Uma outra opção de escolha dos parâmetros do LCRNG é $c = 0$. Dessa forma, evita-se a operação de soma, e os valores da sequência são calculados por: $x_{i+1} = (a\ x_i) \bmod m$. Vale observar que nesse caso, o período da sequência não será m dado que o primeiro termo x_0 deve ser diferente de zero. A escolha de $x_0 = 0$ faria com que todos os demais termos da sequência também valessem zero. Fazendo-se $c = 0$, o período $m-1$ é obtido se o módulo for definido como um número primo.

A seguir será mostrado como boas escolhas de a, c e x_0 podem fazer com que a sequência de números tenha um período máximo m. Para isso, são enunciados três teoremas que garantem um período m de um `LCRNG`. A prova destes teoremas podem ser encontradas em [12].

Teorema A:

O gerador de números aleatórios por congruência linear $x_{i+1} = (ax_i + c) \bmod m$ possui período m se e somente se:

1. c é um número primo em relação a m (ou seja, m e c não possuem divisor comum diferente de 1);
2. $a - 1$ é um múltiplo de todos os números primos c que dividem m;
3. $a - 1$ é um múltiplo 4, se m for múltiplo de 4.

Como exemplo, pode-se escolher um módulo $m = 2^{32}$, $c = 1$ e $a = 4l + 1$, onde l é qualquer inteiro positivo.

Teorema B:

Decompondo o módulo m:

$$m = p_1^{e_1} \dots p_t^{e_t} \tag{2-9}$$

PUC-Rio - Certificação Digital Nº 0921514/CA

sendo $p_1^{e_1} \ldots p_t^{e_t}$ números primos, o máximo valor do período de um `LCRNG` (quando $c = 0$) pode ser calculado pela função λ do módulo m, definida em (2-10):

$$\begin{aligned} \lambda(p^e) &= p^{e-1}(p-1) \\ \lambda(p_1^{e_1} \cdots p_t^{e_t}) &= \text{mmc } (\lambda(p_1^{e_1}) \cdots \lambda(p_t^{e_t})) \end{aligned} \tag{2-10}$$

onde mmc $(a_1 \cdots a_n)$ indica o menor múltiplo comum entre $a_1 \cdots a_n$. Este período é alcançado se:

1. x_0 é primo em relação a m;
2. a é um elemento primitivo módulo m.

Observe que quando $c = 0$, obtém-se um período de comprimento $m - 1$ tomando-se m um número primo. Dessa forma, o período será somente uma unidade menor do que o período máximo. No teorema C, mostra-se as condições necessárias para ter-se um número a sendo um elemento primitivo módulo m.

Teorema C:

O número a é um elemento primitivo módulo p^e se e somente se ocorrer um dos seguintes casos:

1. $p = 2$, $e = 1$, e a é ímpar;
2. $p = 2$, $e = 2$, e $a \bmod 4 = 3$;
3. $p = 2$, $e = 3$, e $a \bmod 8 = 3$, 5 ou 7;
4. $p = 2$, $e \geq 4$, e $a \bmod 8 = 3$ ou 5;
5. p é impar, $a \not\equiv 0$ (modulo p), e $a^{\frac{(p-1)}{q}} \not\equiv 1$ (modulo p), para qualquer divisor primo q de $p - 1$;
6. p é impar, $e > 1$, a atende as condições do item (5) e $a^{p-1} \not\equiv 1$ (modulo p^2).

PUC-Rio - Certificação Digital Nº 0921514/CA

2.5 Exemplos de LCRNG

Na década de 60, a IBM incluiu em suas máquinas um gerador de amostras chamado `RND` ou `RANDU` [17]. Este gerador utilizava congruência linear com os seguintes valores de parâmetros: $a = 65539$, $c = 0$ e $m = 2^{31}$.

O fato de $a = 2^{16} + 3$ tornava a operação de multiplicação muito simples, dado que muitos computadores tinham um comprimento de palavra de 32 bits. Porém, esse valor de a dava a sequência gerada uma propriedade muito ruim, como é mostrado em (2-11). A partir da relação (2-8), pode-se definir o termo x_{i+2} da sequência gerada pelo algoritmo `RND` como:

$$\begin{aligned} x_{i+2} &= [(2^{16}+3)^2 x_i] \bmod 2^{31} \\ &= [(6(2^{16}+3) - 9 + 2^{32})x_i] \bmod 2^{31} \\ &= [(6(2^{16}+3)x_i) \bmod 2^{31} - (9x_i) \bmod 2^{31} + (2^{32}x_i) \bmod 2^{31}] \bmod 2^{31} \\ &= [6x_{i+1} - 9x_i] \bmod 2^{31} \end{aligned} \tag{2-11}$$

Assim, fica mostrado que existia uma forte relação entre três termos consecutivos da sequência. A implementação desse gerador `RND` foi feita em MATLAB e foi utilizada para calcular um valor aproximado para π através do Método de Monte Carlo. As rotinas desenvolvidas para esse exemplo chamam-se: AproximacaoPI e Rnd_IBM.

O cálculo feito foi baseado na relação de volume existente entre um cubo uma esfera inscrita nesse cubo:

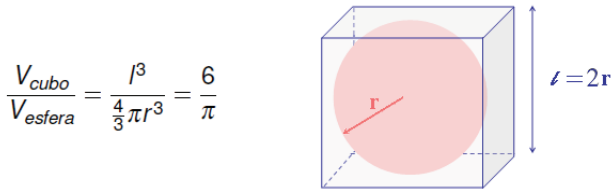

Figura 2.3: Relação de volume entre cubo e esfera.

Etapas do Método de Monte Carlo para aproximar π:

1. Gerar n amostras distribuídas uniformemente dentro de um cubo de lado $l = 2r$.

PUC-Rio - Certificação Digital Nº 0921514/CA

2. Contar quantas amostras estão dentro da esfera de raio r, ou seja quantas amostras atendem a condição: $x_1 + x_2 + x_3 < r$.

3. Calcular a razão R entre o número de amostras dentro da esfera e o número de amostras total n.

4. Aproximação: $\pi = 6R$.

O resultado obtido é mostrado no gráfico da figura (2.4).

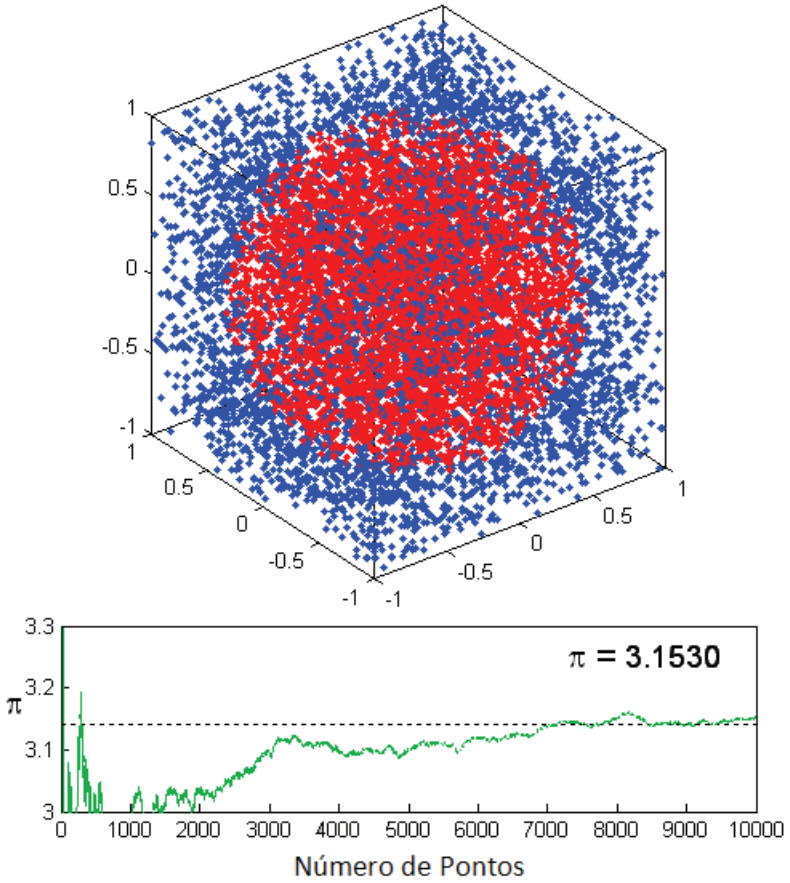

Figura 2.4: Cálculo de uma aproximação para π através do gerador `RND` da IBM utilizando 10000 amostras.

Durante muitos anos, o gerador do MATLAB (comando `rand`) também era baseado em congruência linear. Os valores dos parâmetros utilizados por esse gerador foram recomendados no artigo de Park e Miller [13] e são:

$$a = 7^5 = 16807 \qquad c = 0 \qquad m = 2^{31} - 1 = 2147483647. \qquad (2\text{-}12)$$

O algoritmo do MATLAB é capaz de gerar números reais da forma $\frac{k}{m}$ para $k = 1, \cdots m-1$. O menor e maior são respectivamente 0.00000000046566 e $0,99999999953434$. A sequência se repete após $m - 1$ valores, ou seja, em torno de 2 bilhões de números.

PUC-Rio - Certificação Digital Nº 0921514/CA

Esse antigo gerador do MATLAB também foi utilizado no Método de Monte Carlo para calcular uma aproximação para π, assim como feito para o gerador da IBM da década de 60. As rotinas desenvolvidas neste exemplo chamam-se AproximacaoPI e Rnd_MATLAB. O resultado obtido é mostrado no gráfico da figura (2.5).

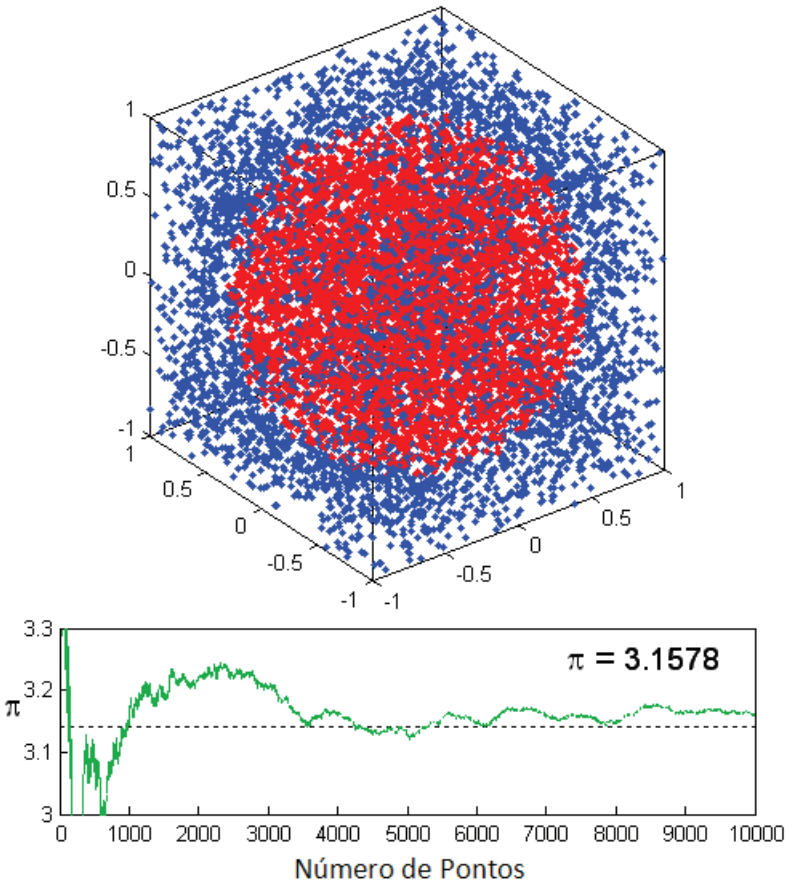

Figura 2.5: Cálculo de uma aproximação para π através do gerador antigo do MATLAB utilizando 10000 amostras.

Para poder-se comparar os resultados fornecidos pelos algoritmos da IBM e do antigo MATLAB, foram traçados histogramas com os resultados de 2000 simulações para o cálculo aproximado de π. Em cada simulação foram utilizados 10000 pontos, e os resultados são mostrados nas figuras (2.6) e (2.7).

Foram calculados também a variância e desvio padrão das simulações. Para o algoritmo da IBM foi obtida uma variância de 0.0021 e desvio padrão de 0.0463. Para o algoritmo do antigo MATLAB foi obtida uma variância de 0.0021 e desvio padrão de 0.0454. Esses resultados mostram que ambos os geradores foram eficazes na tarefa de aproximar π e obtiveram uma precisão de resultados parecida.

Em 1995, a versão 5 do MATLAB introduziu um tipo completamente diferente de gerador de números aleatórios. O algoritmo é baseado no trabalho

PUC-Rio - Certificação Digital Nº 0921514/CA

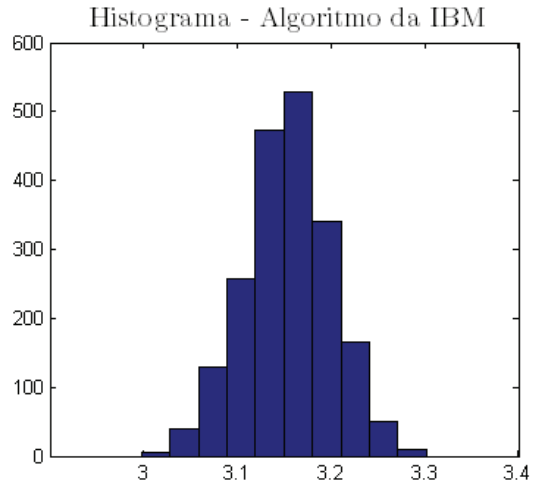

Figura 2.6: Histograma - gerador da IBM.

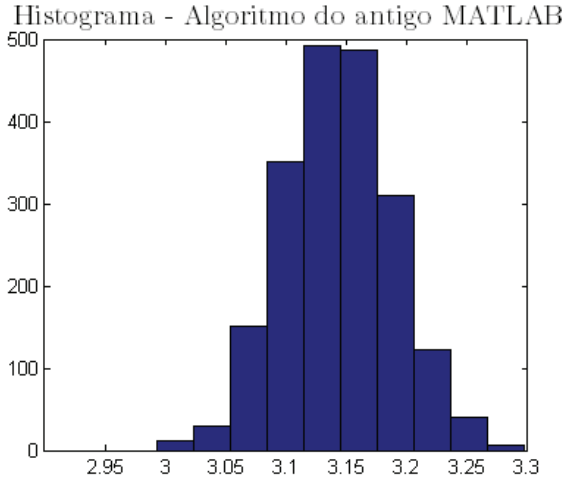

Figura 2.7: Histograma - gerador antigo do MATLAB.

de George Marsaglia, um professor da *Florida State University* [4].

O gerador de Marsaglia [4] não utiliza congruência em seu algoritmo. No código também não são realizadas nem multiplicações nem divisões e, ao invés de ter uma única semente, o novo gerador tem 35 palavras de memória interna ou o estado. Trinta e duas dessas palavras são números entre 0 e 1.

2.6 Interpretação Geométrica do Gerador LCRNG

Nesta seção do trabalho, faz-se uma interpretação geométrica do algoritmo do gerador por congruência linear. A figura (2.8) exemplifica um gerador $x_{i+1} = (5\ x_i + 1) \bmod 32$. Pode-se observar que os elementos da sequência $x_0, x_1, \ldots$ estão localizados sobre as retas em vermelho ($f(x) = (5\ x + 1) \bmod 32$). A função aumenta linearmente a partir de $f(0) = 1$ até atingir o valor 31. Nesse ponto, a operação do módulo faz com que o valor de

PUC-Rio - Certificação Digital Nº 0921514/CA

f seja reduzido a zero, criando uma descontinuidade.

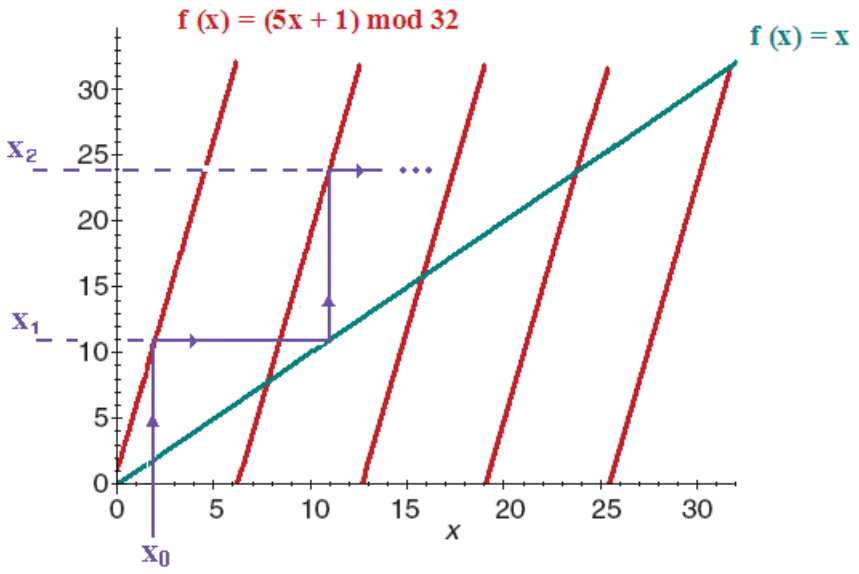

Figura 2.8: Interpretação Geométrica do Gerador LCRNG.

2.7 Outros Geradores de Amostras de Variáveis Aleatórias

Outros algoritmos, diferentes dos geradores de congruência linear, foram propostos nos últimos anos. Como exemplo, tem-se o algoritmo, criado na década de 50, baseado na sequência de *Fibonacci*. Nessa classe de geradores, o valor de um elemento x_{i+1} depende não somente de x_i e sim de mais de um valor anterior da sequência, como mostrado em (2-13):

$$x_{i+1} = (x_i + x_{i-1}) \bmod m. \tag{2-13}$$

Esse gerador, apesar de ter um período maior do que m, produz uma sequência de valores com uma forte relação entre termos consecutivos, ou seja, apresenta um comportamento semelhante ao gerador utilizado na década de 60 pela IBM (2-11).

Geradores baseados na sequência de *Fibonacci* também foram desenvolvidos por Green Smith e Klem [12] no final da década de 50. A ideia era utilizar no cálculo do valor um elemento x_n elementos anteriores, ou seja, x_{i-k}, sendo que k deve ser maior ou igual a 16.

$$x_{i+1} = (x_i + x_{i-k}) \bmod m \qquad k \geq 16 \tag{2-14}$$

Outros geradores de números foram propostos, como o algoritmo de G. J. Mitchell e D. P. Moore [12] de 1958 mostrado na equação (2-15):

$$x_i = (x_{i-24} + x_{i-55}) \bmod m \qquad i \geq 55 \tag{2-15}$$

PUC-Rio - Certificação Digital Nº 0921514/CA

onde m é par, e $x_0 \cdots, x_{54}$ são inteiros não todos pares. Mais um exemplo interessante é um gerador não linear que utiliza como semente para o termo $i+1$, o inverso do termo x_{i-1}.

$$x_{i+1} = (a\, x_{i-1}^{-1} + c) \bmod p \qquad \text{com } p \text{ primo.} \tag{2-16}$$

Knuth sugeriu em 1998 um outro tipo de gerador baseado em congruência não linear [15] no qual os termos anteriores ao elemento x_i formam um polinômio de grau 2:

$$x_{i+1} = (d\, x_{i-2}^2 + a\, x_{i-1} + c) \bmod m. \tag{2-17}$$

Geradores com polinômios de grau superior a 2 também foram propostos.

2.8 Geração de Amostras de Variáveis Aleatórias com Distribuição de Probabilidade diferente da Uniforme: Método da Transformada Inversa

Até a presente seção, somente foram apresentados métodos para se gerar amostras de variáveis aleatórias com uma distribuição de probabilidade uniforme. Porém, muitas vezes ao aplicar o método de Monte Carlo, é necessário simular o comportamento de variáveis aleatórias com outras distribuições de probabilidade, como por exemplo, a distribuição Gaussiana, Exponencial, Beta, etc [15]. Por isso, nesta seção do trabalho é mostrado como obter amostras dessas distribuições diferentes da uniforme através do Método da Transformada Inversa.

Segundo esse método, uma amostra x_i de uma variável aleatória X com distribuição de probabilidade P contínua e estritamente monotônica, pode ser gerada através das seguintes etapas:

1. gere uma amostra u_i da distribuição uniforme no intervalo unitário $[0, 1]$ (pode-se utilizar por exemplo algum gerador de congruência linear apresentado anteriormente);

2. faça $x_i = P^{-1}(u_i)$.

A demonstração de que os valores gerados, $\{x_i\}_{i\in\mathbb{N}}$, são de fato amostras de X é mostrada a seguir.

Como já definido anteriormente, uma variável aleatória real é uma função real X que associa a cada ponto amostra w, pertencente ao espaço de amostras Ω, um número $x(\mathrm{w}) \in \mathbb{R}$ [27] [16]. Pode-se escrever:

$$\begin{array}{rccc} X: & \Omega & \longmapsto & \mathbb{R} \\ & \mathrm{w} & \longmapsto & x(\mathrm{w}). \end{array} \tag{2-18}$$

PUC-Rio - Certificação Digital Nº 0921514/CA

A distribuição de probabilidade P de uma variável aleatória arbitrária X é definida para todo $x \in \mathbb{R}$ pela relação:

$$P(x) = Pr(X \leq x) \tag{2-19}$$

onde $Pr(X \leq x)$ é a probabilidade da variável aleatória X ser menor ou igual a x. Essa função P é não decrescente, mas não necessariamente estritamente monotônica e, não necessariamente contínua. Observe que P pode ter períodos de estabilidade ou descontinuidades. Contudo, sabe-se que:

$$\lim_{x \to -\infty} P(x) = 0 \qquad \text{e} \qquad \lim_{x \to \infty} P(x) = 1 \tag{2-20}$$

Supondo que a função P é contínua e estritamente monotônica, tem-se que a função inversa P^{-1} existe e é contínua. Ela é definida como:

$$P^{-1}(y) = \inf \{x | P(x) \geq y \ , \ 0 < y < 1\}. \tag{2-21}$$

Assim, para $-\infty < x < \infty$ e $0 < y < 1$:

$$P^{-1}(P(x)) = x \qquad \text{e} \qquad P(P^{-1}(y)) = y. \tag{2-22}$$

Sendo U uma variável aleatória uniforme no intervalo unitário $[0, 1]$, tem-se que:

$$\begin{aligned} Pr(P^{-1}(U) \leq x) &= Pr(P(P^{-1}(U)) \leq P(x)) \\ &= Pr(U \leq P(x)). \end{aligned} \tag{2-23}$$

Como U é distribuída uniformemente no intervalo $[0, 1]$, a probabilidade $Pr(U \leq P(x))$ é igual ao tamanho do intervalo $[0, P(x)]$. Dessa forma:

$$Pr(P^{-1}(U) \leq x) = P(x) \tag{2-24}$$

o que justifica o método da Transformada Inversa.

A tabela (2.3) mostra algumas distribuições em que o método da Transformada Inversa pode ser aplicado.

A função densidade de probabilidade Gaussiana não possui uma expressão analítica fechada de sua inversa. Por isso, uma opção para gerar amostras de variáveis aleatórias com essa distribuição é utilizar o método de Box-Muller proposto 1958. Ele é explicado na próxima seção da dissertação.

Exemplo 2.8.1 (Método da Transformada Inversa)**.** Suponha que deseja-se gerar amostras de uma variável aleatória X com uma função densidade de probabilidade p dada por:

PUC-Rio - Certificação Digital Nº 0921514/CA

Distribuição	pdf $p(x)$	Inversa $P^{-1}(u)$
Uniforme $\mathcal{U}(a,b)$ $a \leq x \leq b$	$\frac{1}{b-a}$	$a+(b-a)u$
Exponencial $\mathcal{E}(\lambda)$ $\lambda > 0;\ x \geq 0$	$\lambda\, e^{-\lambda x}$	$-\frac{1}{\lambda}\ln(1-u)$
Beta $\mathcal{B}(\alpha,1)$ $\alpha > 0;\ 0 \leq x \leq 1$	$\alpha\, x^{\alpha-1}$	$u^{\frac{1}{\alpha}}$
Beta $\mathcal{B}(1,\beta)$ $\beta > 0;\ 0 \leq x \leq 1$	$\beta\,(1-x)^{\beta-1}$	$1-(1-u)^{\frac{1}{\beta}}$
Logística $\mathcal{L}(\alpha,\beta)$ $\beta > 0;\ -\infty < x;\ \alpha < \infty$	$\frac{e^{-(x-\alpha)/\beta}}{\beta[1+e^{-(x-\alpha)/\beta}]^2}$	$\alpha+\beta\ln[u/(1-u)]$
Weibull $\mathcal{W}(\alpha,\beta)$ $\alpha,\beta > 0;\ x \geq 0$	$\frac{\alpha}{\beta}\left(\frac{x}{\beta}\right)^{\alpha-1} e^{-(x/\beta)^{\alpha}}$	$\beta[-\ln(1-u)]^{1/\alpha}$
Cauchy $\mathcal{C}(\alpha,\beta)$ $\beta > 0;\ -\infty < x;\ \alpha < \infty$	$\frac{1}{\pi}\frac{\beta}{(x-\alpha)^2+\beta^2}$	$\alpha+\beta\tan\pi[u-(1/2)]$

Tabela 2.3: Método da Transformada Inversa aplicado a algumas distribuições.

$$p(x) = \begin{cases} 2x \quad , \quad 0 \leq x \leq 1 \\ \\ 0 \quad , \quad x < 0 \text{ ou } x > 1 \end{cases} \tag{2-25}$$

A função distribuição de probabilidade P será:

$$P(x) = \begin{cases} 0 \quad , \quad x < 0 \\ \\ \int_0^x 2y\, dy = x^2 \quad , \quad 0 \leq x \leq 1 \\ \\ 1 \quad , \quad x > 1 \end{cases} \tag{2-26}$$

Aplicando o Método da Transformada Inversa, cada amostra x_i de X é obtida por:

$$x_i = P^{-1}(u_i) = \sqrt{u_i} \qquad \forall i \in \mathbb{N} \tag{2-27}$$

onde cada u_i é uma amostra de uma variável aleatória uniforme, U, no intervalo

PUC-Rio - Certificação Digital Nº 0921514/CA

unitário $[0, 1]$.

Para verificar a eficiência do método apresentado, foi implementada uma rotina MATLAB chamada TransformadaInversa que compara o histograma das amostras de X (obtidas com o Método da Transformada Inversa) com a função densidade de probabilidade f. Nessa rotina, as amostras de U foram geradas através do comando `rand` do MATLAB. A figura (2.9) mostra os gráficos obtidos.

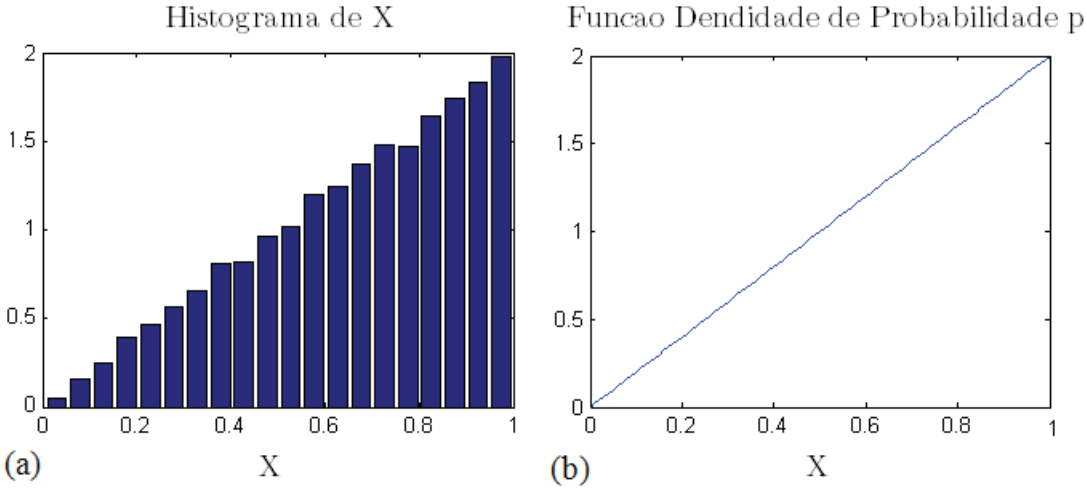

Figura 2.9: (a) Histograma de 10000 amostras de X. (b) Gráfico da função densidade de probabilidade f (2-25).

Na tabela (2.4) pode-se comparar as estatísticas (média, variância e desvio padrão) calculadas com diferentes números de amostras de X. Sendo a média de X igual a $0,6667$, a variância 0.0556 e, o desvio padrão 0.2357, verifica-se que o Método da Transformada Inversa obtém boas aproximações para essas estatísticas.

Número de amostras	1.000	5.000	10.000
Média	0.6727	0.6634	0.6658
Variância	0.0565	0.0557	0.0554
Desvio padrão	0.2377	0.2360	0.2353

Tabela 2.4: Estatísticas das amostras de X obtidas pelo método da Transformada Inversa.

2.9 Geração de Amostras de Variáveis Aleatórias Gaussianas

Amostras de variáveis aleatórias Gaussianas podem ser geradas através do Método de Box-Muller proposto em 1958 [32].

PUC-Rio - Certificação Digital Nº 0921514/CA

Nesse método, amostras de duas variáveis aleatórias Gaussianas, X_1 e X_2, com média nula e variância um são obtidas através de duas variáveis aleatórias uniformes, U_1 e U_2, no intervalo unitário $[0, 1]$.

O método é descrito a seguir. Sejam X_1 e X_2 duas variáveis aleatórias gaussianas independentes de média nula e variância um. A função densidade de probabilidade conjunta de X_1 e X_2 é o produto:

$$p(x_1, x_2) = \frac{1}{\sqrt{2\pi}}\, e^{-x_1^2/2} \,.\, \frac{1}{\sqrt{2\pi}}\, e^{-x_2^2/2} = \frac{1}{\sqrt{2\pi}}\, e^{-(x_1^2+x_2^2)/2}. \tag{2-28}$$

Como esta densidade conjunta é radialmente simétrica, X_1 e X_2 podem ser escritos em termos de duas variáveis aleatórias que representem as coordenadas polares: R e Θ, com $0 < \Theta < 2\pi$. Assim:

$$X_1 = R\cos(\Theta) \qquad \text{e} \qquad X_2 = R\sin(\Theta). \tag{2-29}$$

Sendo Θ uniformemente distribuído no intervalo $[0, 2\pi]$, amostras $\theta^{(i)}$,$i \in \mathbb{N}$, são obtidas por:

$$\theta^{(i)} = 2\pi u_1^{(i)}. \tag{2-30}$$

sendo $u_1^{(i)}$ amostras de uma variável aleatória uniforme, U_1, no intervalo unitário $[0, 1]$. A função distribuição de probabilidade de R pode ser escrito como:

$$Pr(R < r) = \int_{\Theta=0}^{2\pi}\int_{r'=0}^{r} \frac{1}{2\pi} e^{-r'^2/2} r' dr'\, d\theta = \int_{r'=0}^{r} e^{-r'^2/2} r' dr'. \tag{2-31}$$

Fazendo uma mudança de variáveis: $r'^2/2 = s$, $r'dr' = ds$, calcula-se:

$$Pr(R < r) = \int_{s=0}^{r^2/2} e^{-s} ds = 1 - e^{-r^2/2}. \tag{2-32}$$

A solução da distribuição de probabilidade (2-32) é assumir que: $R = \sqrt{-2\ln(U_2)}$, sendo U_2 uma variável aleatória uniforme no intervalo unitário $[0, 1]$.

Dessa forma, para gerar amostras $x_1^{(i)}$ e $x_2^{(i)}$, com $i \in \mathbb{N}$, de X_1 e X_2:

1. gera-se uma amostra $u_1^{(i)}$ de U_1 e $u_2^{(i)}$ de U_2;
2. calcula-se: $\theta^{(i)} = 2\pi u_1^{(i)}$ e $r^{(i)} = \sqrt{-2\ln(u_2^{(i)})}$;
3. calcula-se: $x_1^{(i)} = r^{(i)}\cos(\theta^{(i)})$ e $x_2^{(i)} = r^{(i)}\sin(\theta^{(i)})$.

Vale observar que as amostras geradas de X_1 e X_2 (Gaussianas padrão, ou seja, média nula e variância um) podem ser transformadas em amostras de

PUC-Rio - Certificação Digital Nº 0921514/CA

variáveis aleatórias Gaussianas Y_1 e Y_2 com médias μ_1, μ_2 e variâncias σ_1^2, σ_2^2 através de:

$$y_1^{(i)} = \mu_1 + \sigma_1 \, x_1^{(i)} \tag{2-33}$$

$$y_2^{(i)} = \mu_2 + \sigma_2 \, x_2^{(i)}. \tag{2-34}$$

Exemplo 2.9.1 (Método de Box-Muller)**.** Suponha que deseja-se gerar amostras de duas variáveis aleatórias Gaussianas Y_1 e Y_2 através do método de Box-Muller. A média de Y_1 vale $\mu_1 = 2,0$ e, sua variância $\sigma_1^2 = 2,0$. Para X_2, $\mu_1 = 3,0$ e $\sigma_2^2 = 3,0$. Suas funções distribuições de probabilidade são:

$$p_{Y_1}(y_1) = \frac{1}{\sqrt{2\pi\sigma_1^2}} \, e^{-(y_1-\mu_1)^2/2\sigma_1^2} \tag{2-35}$$

$$p_{Y_2}(y_2) = \frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-(y_2-\mu_2)^2/2\sigma_2^2}. \tag{2-36}$$

As amostras são geradas através de uma rotina implementada em MATLAB chamada BoxMuller. Nessa rotina, as amostras das variáveis uniformes unitárias em $[0, 1]$, U_1 e U_2, foram geradas com o comando `rand`. Os gráficos da figura (2.10) comparam os gráficos de frequência das amostras de Y_1 e Y_2 (obtidas com o Método de Box-Muller) com as função densidade de probabilidade p_{Y_1} e p_{Y_2}.

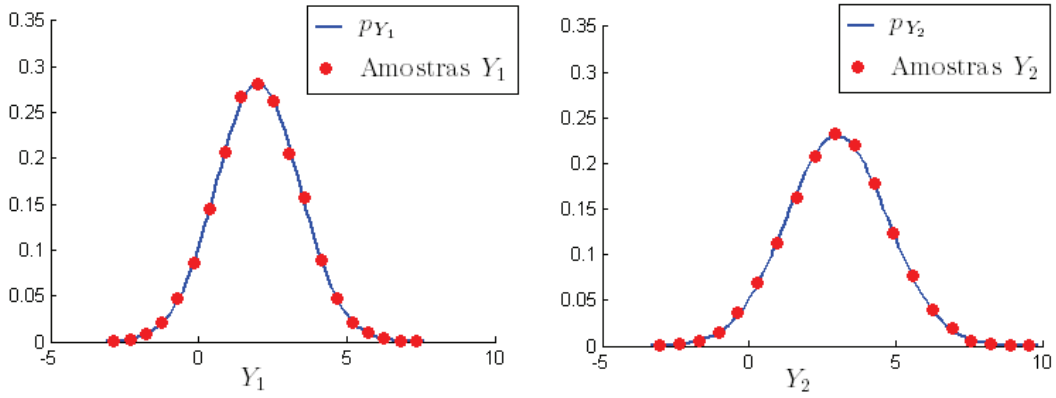

Figura 2.10: (a) Gráficos de frequência de 10000 amostras de Y_1. (b) Gráficos de frequência de 10000 amostras de Y_2.

Na tabela (2.5) pode-se comparar as estatísticas (média, variância e desvio padrão) calculadas com diferentes números de amostras de Y_1 e Y_2. Verifica-se que o Método de Box-Muller obtém boas aproximações para essas estatísticas.

PUC-Rio - Certificação Digital Nº 0921514/CA

	X_1			X_2		
Número de amostras	1.000	5.000	10.000	1.000	5.000	10.000
Média	2.0406	2.0109	1.9970	2.9658	3.0374	3.0030
Variância	1.9476	1.9709	1.9928	3.0270	2.9254	2.9461

Tabela 2.5: Estatísticas das amostras de X_1 e X_2 obtidas pelo Método de Box-Muller.

2.10 Testes de Geradores de Amostras de Variáveis Aleatórias

Quando deseja-se simular o comportamento de uma variável aleatória através de uma sequência numérica, tem-se como principal objetivo a obtenção de amostras que de fato comportem-se de forma similar à variável aleatória.

Nas seções anteriores deste trabalho, foi feito um estudo dos parâmetros dos geradores. Mostraram-se, por exemplo, alguns teoremas que garantem que o período de uma sequência pode ser tão longo quando o desejado. Porém, apesar do período ser um critério importante, não garante que os números obtidos com a sequência simulam se fato uma variável aleatória. Ou seja, se são 'suficientemente' aleatórios.

Por isso, nesta seção são apresentados testes que têm como objetivo verificar a não-aleatoriedade de uma sequência. Isto é, apenas pode mostrar que a sequência não é aleatória (não é capaz de revelar se a sequência é de fato aleatória). Outros testes são mostrados em [12].

A consideração de que uma sequência produzida por um gerador de números aleatórios é aleatória, está sempre associada a realização de um número finito n de testes $T_1,\ T_2,\ \cdots,\ T_n$. Nada pode ser afirmado sobre o comportamento se sequência quando ela for submetida a um teste T_{n+1}.

No entanto, cada avaliação feita permite que se tenha mais confiança sobre a real aleatoriedade da sequência.

2.10.1 Teste do Qui−quadrado

O teste do qui-quadrado [12] é um teste estatístico utilizado para verificar se os resultados associados a um experimento estão de acordo uma especificada distribuição de probabilidade. Abaixo segue um roteiro para a aplicação do método.

Suponha que os possíveis resultados de um experimento são particionados em eventos $E_1,\ E_2,\ \cdots,\ E_k$. Suponha ainda que a probabilidade de um evento E_i seja dada por p_i, com $p_i > 0$ e $\sum_{i=1}^{k} p_i = 1$. Sendo o experimento realizado

PUC-Rio - Certificação Digital Nº 0921514/CA

N vezes, espera-se que o número de ocorrências Y_i do evento E_i seja igual a $(N\ p_i)$.

Uma opção para saber se cada um dos eventos do experimento está ocorrendo de forma adequada, é calcular o quadrado da diferença entre o real número de ocorrências Y_i e o número esperado de realizações $(N\ p_i)$.

A partir dessa conta, pode-se determinar se o experimento como um todo está ocorrendo de forma adequada. Para isto basta fazer o somatório dessa diferença entre o número real de ocorrências e o número esperado de realizações para todos os k eventos do experimento:

$$V = \sum_{i=1}^{k} \frac{(Y_i - N\ p_i)^2}{N\ p_i}. \tag{2-37}$$

Na expressão (2-37), V pode ser aproximada por uma variável aleatória qui-quadrado com $\nu = k - 1$ graus de liberdade. Vale observar que essa aproximação somente é válida quando N é suficientemente grande e, em geral utiliza-se a seguinte regra [12]: N é considerado suficientemente grande, quando o número esperado de realizações de cada evento $(N\ p_i)$ é maior ou igual a 5.

Possíveis valores de V estão tabelados na figura (2.11). A interpretação dessa tabela é feita da seguinte forma: o valor x da tabela em uma determinada linha ν e coluna p indica que a quantidade V da equação (2-37) será menor ou igual a x com uma probabilidade aproximada p.

O teste do qui-quadrado é aplicado a experimentos associados a variáveis aleatórias discretas e contínuas. Detalhes do método aplicado a variáveis contínuas podem ser encontrados no artigo de Watson G. S. [1].

2.10.2 Teste de Kolmogorov−Smirnov

O teste de Kolmogorov−Smirnov [12], assim como o teste do qui-quadrado, também é utilizado para verificar se os elementos da sequência numérica produzida por geradores de variáveis aleatórias estão de acordo uma especificada distribuição de probabilidade. Abaixo segue um roteiro para a aplicação do método.

Suponha que se deseja verificar se as amostras $x_1,\ x_2,\ \cdots,\ x_n$ estão de acordo com uma função distribuição de probabilidade P. Para isso inicialmente calcula-se uma aproximação para a função de distribuição acumulada através da fração entre o número de amostras x_i que são inferiores ou iguais a t e o número total de amostras (2-38).

$$\hat{P}_N(t) = \frac{\#\{i\ \ x_i \leq t\}}{N}. \tag{2-38}$$

Supondo que x_i são realizações independentes, tem-se que:

PUC-Rio - Certificação Digital Nº 0921514/CA

	$p = 0.01$	$p = 0.05$	$p = 0.25$	$p = 0.50$	$p = 0.75$	$p = 0.95$	$p = 0.99$
$\nu = 1$	0.00016	0.00393	0.1015	0.4549	1.323	3.841	6.635
$\nu = 2$	0.02010	0.1026	0.5753	1.386	2.773	5.991	9.210
$\nu = 3$	0.1148	0.3518	1.213	2.366	4.108	7.815	11.34
$\nu = 4$	0.2971	0.7107	1.923	3.357	5.385	9.488	13.28
$\nu = 5$	0.5543	1.1455	2.675	4.351	6.626	11.07	15.09
$\nu = 6$	0.8720	1.635	3.455	5.348	7.841	12.59	16.81
$\nu = 7$	1.239	2.167	4.255	6.346	9.037	14.07	18.48
$\nu = 8$	1.646	2.733	5.071	7.344	10.22	15.51	20.09
$\nu = 9$	2.088	3.325	5.899	8.343	11.39	16.92	21.67
$\nu = 10$	2.558	3.940	6.737	9.342	12.55	18.31	23.21
$\nu = 11$	3.053	4.575	7.584	10.34	13.70	19.68	24.73
$\nu = 12$	3.571	5.226	8.438	11.34	14.84	21.03	26.22
$\nu = 15$	5.229	7.261	11.04	14.34	18.25	25.00	30.58
$\nu = 20$	8.260	10.85	15.45	19.34	23.83	31.41	37.57
$\nu = 30$	14.95	18.49	24.48	29.34	34.80	43.77	50.89
$\nu = 50$	29.71	34.76	42.94	49.33	56.33	67.50	76.15
$\nu > 30$	$\nu + \sqrt{2\nu}\, x_p + \frac{2}{3}x_p^2 - \frac{2}{3} + O(1/\sqrt{\nu})$						

Figura 2.11: Limites do teste do qui-quadrado.

$$\lim_{N\to\infty} \sup_t \left|\hat{P}_N(t) - P(t)\right| = 0 \tag{2-39}$$

Seja:

$$D_N = \sup_t \left|\hat{P}_N(t) - P(t)\right| \tag{2-40}$$

como mostrado ilustrado na figura (2.12), onde P é a função de probabilidade acumulada da distribuição uniforme.

PUC-Rio - Certificação Digital Nº 0921514/CA

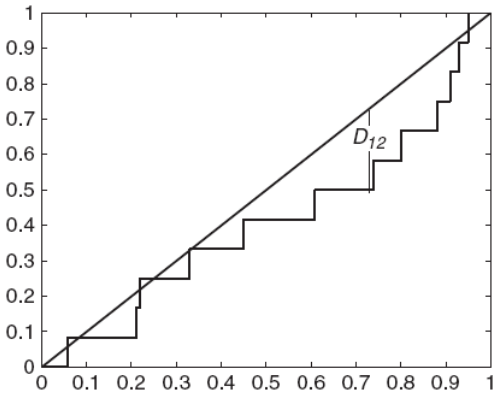

Figura 2.12: Função de probabilidade acumulada P (de uma distribuição uniforme) e função distribuição acumulada aproximada para um experimento com 12 realizações.

A tabela da figura (2.13) apresenta os limites de aceitação de D_N pelo teste de Kolmogorov−Smirnov. Definindo-se $D_N = \left\| \hat{P}_N(t) - P(t) \right\|_\infty$, o valor da tabela $d_{N,\alpha}$ significa $Pr(D_N \geq d_{N,\alpha}) = \alpha$.

	α			
	$p = 0.20$	$p = 0.10$	$p = 0.05$	$p = 0.01$
$n = 1$	0.90	0.95	0.98	0.99
$n = 2$	0.68	0.78	0.84	0.93
$n = 3$	0.56	0.64	0.71	0.83
$n = 4$	0.49	0.56	0.62	0.73
$n = 5$	0.45	0.51	0.56	0.67
$n = 6$	0.41	0.47	0.52	0.62
$n = 7$	0.38	0.44	0.49	0.58
$n = 8$	0.36	0.41	0.46	0.54
$n = 9$	0.34	0.39	0.43	0.51
$n = 10$	0.32	0.37	0.41	0.49
$n = 11$	0.31	0.35	0.39	0.47
$n = 12$	0.30	0.34	0.38	0.45
$n = 15$	0.27	0.30	0.34	0.40
$n = 20$	0.23	0.26	0.29	0.35
$n = 30$	0.19	0.22	0.24	0.29
$n = 35$	0.18	0.21	0.23	0.27
$n = 40$	0.17	0.19	0.21	0.25
$n = 45$	0.16	0.18	0.20	0.24
$n > 45$	$\frac{1.07}{\sqrt{n}}$	$\frac{1.22}{\sqrt{n}}$	$\frac{1.36}{\sqrt{n}}$	$\frac{1.63}{\sqrt{n}}$

Figura 2.13: Limites do teste de Kolmogorov-Smirnov

PUC-Rio - Certificação Digital Nº 0921514/CA