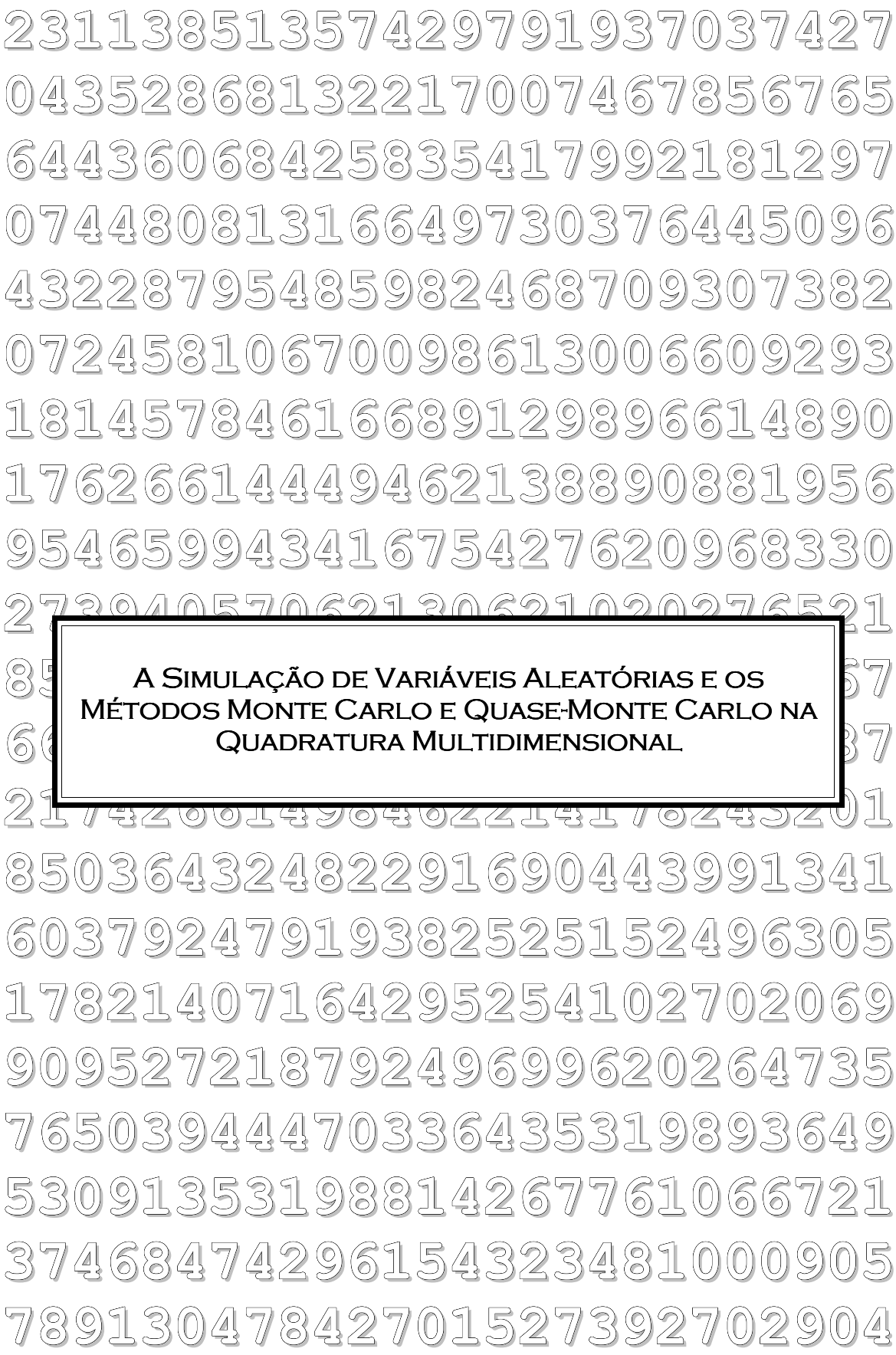

A Simulação de Variáveis Aleatórias e os Métodos Monte Carlo e Quase-Monte Carlo na Quadratura Multidimensional

Adalberto Ayjara Dornelles Filho

A Simulação de Variáveis Aleatórias e os Métodos Monte Carlo e Quase-Monte Carlo na Quadratura Multidimensional

Dissertação apresentada como requisito parcial para o grau de Mestre pelo Curso de Pós-Graduação em Matemática Aplicada.
Convênio Universidade de Caxias do Sul - Universidade Federal do Rio Grande do Sul.
Orientador: Prof. Dr. Oclide José Dotto

Caxias do Sul, 4 de abril de 2000.

Any one who considers arithmetical methods of producing random digits is, of course, in state of sin.
John von Neumann (1951)

Virtually every treatment of multiple integration begins with the lament that quadrature in higher dimentions is hard.
Ronald A. Thisted (1996)

Random numbers should not be generated with a method chosen at random. Some theory should be used.
Donald E. Knuth (1998)

Agradecimento

O presente trabalho é fruto de alguma inspiração e de muita transpiração. Ao longo de sua elaboração muitas pessoas contribuíram de uma forma ou de outra para que ele chegasse a termo com êxito. A todas elas e, em especial,

A meu orientador, Prof. Dr. Oclide José Dotto, pela exigente e criteriosa orientação, pela paciência em mostrar o caminho;

A meus colegas de estudos, José Caleffi, Simone Gonçalves, Lucile Jacobi e Mauren Turra, pelo tempo que passamos juntos descobrindo o mundo e aprendendo matemática;

A meus pais, Adalberto e Maria Isolda, pelo carinho e por me ensinarem a persistir;

A minha esposa, Elisa, pelo amor, pelo companheirismo, pela compreensão, pelo incentivo, por dar sentido ao trabalho e a vida;

meu sincero agradecimento.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Teoremas

Resumo

Monte Carlo é o nome dado de forma geral às técnicas de resolução de problemas numéricos através do uso intensivo de números aleatórios. No trato computacional, esses números não são, de fato, aleatórios, mas **pseudo-aleatórios,** pois são gerados por algoritmos determinísticos que, no entanto, "parecem" aleatórios, isto é, são aprovados em testes de aleatoriedade. Variáveis aleatórias com quaisquer distribuições de probabilidade são então simuladas a partir de números pseudo-aleatórios uniformemente distribuídos no intervalo $(0;1)$ através de certas transformações. Entre as diversas aplicações do método Monte Carlo destaca-se a quadratura numérica multidimensional, que consiste essencialmente em estimar o valor médio da função integranda através do valor médio da função em pontos escolhidos de modo aleatório no interior da região de integração. Técnicas especiais de amostragem permitem a **redução da variância** e, em conseqüência, do erro nos valores estimados. O erro de convergência do método é, no pior caso, de ordem $O\left(n^{-1/2}\right)$. No entanto o uso de pontos amostrais **quase-aleatórios** pode levar a convergência mais rápida de ordem $O\left(n^{-1}\right)$. O presente trabalho descreve uma grande quantidade de algoritmos para obtenção de variáveis pseudo-aleatórias e quase-aleatórias; para a transformação de diversas distribuições de probabilidade e para quadratura multidimensional.

Abstract

Monte Carlo is the name usually given to numerical problems resolution techniques by intensive use of random numbers. In computer procedures, this numbers are not, in fact, random but **pseudo-random** because they are generated by deterministic algorithms, but "look like" random, that is, they pass on randomness tests. Such random variables with any probability distribution are simulated on pseudo-random numbers with uniform distribution in $(0;1)$ by certain transformations. Among a diversity of Monte Carlo methods applications, a special one is the multidimensional numeric quadrature which consists essentially of estimating tha integrand function mean value by the mean that function at random points in the integration region. Sampling techniques allow a variance reduction and hence an estimated error reduction. The error convergence order is, in the worst case, $O\left(n^{-1/2}\right)$. However quasi-random sampling points could bring a faster convergence order of $O\left(n^{-1}\right)$. The present work describes a wide quantity of algorithms for producing pseudo-random and quasi-random variables; for transforming a diversity of probability distributions, and for multidimensional quadrature.

1 Introdução.

1.1 Apresentação.

O papel do método Monte Carlo (MC) em todas as ciências tem aumentado de importância nos últimos anos. Esse método é ferramenta importante de áreas de rápido crescimento como a matemática computacional, a física computacional e outras ciências computacionais. O crescente desenvolvimento do poder de processamento dos computadores e, especialmente, das técnicas de simulação têm mostrado que a computação é uma terceira via de descobrimento nas ciências naturais ao lado da teoria e da experimentação tradicionais. Gentle (1998) relata que em 1983, 25% dos artigos publicados no conceituado *Journal of the American Statistical Association* incluíam experimentos pelo método MC. Em 1995 essa proporção aumentou para 49%.

O ponto fundamental do método Monte Carlo é a geração de números pseudo-aleatórios. Esse tema acompanha a evolução computacional desde os seus primeiros passos e ainda hoje é tema para produção científica. Entre as aplicações do método, destaca-se o cálculo de integrais multidimensionais por amostragem pseudo-aleatória que, segundo muitos autores, é única alternativa importante aos métodos de quadratura numérica. Técnicas de redução de variância têm sido desenvolvidas no intuito de diminuir o erro de truncamento inerente ao processo. Recentemente, a técnica Quase-Monte Carlo tem-se mostrado competente no tratamento de problemas de quadratura multidimensional.

No presente trabalho procuramos descrever dois aspectos do método: a simulação de variáveis pseudo-aleatórias e sua aplicação na integração multidimensional.

O Capítulo 1 faz uma breve introdução da expressão Monte Carlo, sua origem histórica e um apanhado geral das aplicações do método. O Capítulo 2 descreve as formas de geração de números aleatórios. Ênfase especial é dada aos algoritmos mais utilizados na simulação de números pseudo-aleatórios. O Capítulo 3 mostra as técnicas de transformação da variável aleatória uniforme em outras variáveis discretas ou contínuas com diferentes distribuições de probabilidade. O Capítulo 4 trata da aplicação do método Monte Carlo no cálculo de integrais multidimensionais. Discute, ainda, algumas técnicas de redução de variância. O Capítulo 5 enfoca as seqüências de baixa discrepância e o método Quase-Monte Carlo. O Capítulo 6 mostra alguns exemplos de quadratura multidimensional usando as técnicas descritas e comparando resultados. O Apêndice A expõe de forma simplificada e com o auxílio de alguns exemplos as várias aplicações do método Monte Carlo na simulação. O Apêndice B

traz programas para a implementação dos algoritmos descritos ao longo do texto no pacote matemático MATLAB.

Alguns aspectos do presente trabalho que julgamos relevantes e de mérito são os seguintes:

- o método da partição regular (PR) parece ser uma alternativa óbvia para o problema da quadratura multidimensional; para dimensão $d = 1$, esse método é realmente mais eficiente que o método Monte Carlo (MC), no entanto não o é para $d > 2$, fato que passa desapercebido a Press *et alii* (1992), que afirmam que o método da partição regular transforma o método MC em *a deterministic quadrature scheme - albeit a simple one - whose fractional error decreases at least as fast as* n^{-1} *(even faster if the functions goes to zero smoothly at the boundaries of the sampled region ...); s*e o método PR converge a uma taxa maior, então o que o impede sua utilização? Press *et alii* (1992) apontam que *the trouble with a grid is that one has to decide in advance how fine it should be;* essa justificativa não nos pareceu forte o suficiente para descartar o método PR; e ainda é contraditória com Morokoff & Caflisch (1995) que descartam o método PR, pois *... grids suffer from several dificulties; first, in high dimensions, the number of points required to create even a coarse mesh is exponentially large in dimention;* alguns experimentos númericos (como o da Figura 5.1) nos levaram a conclusão de que, de fato, a convergência do método não é satisfatória; para tornar mais rigorosa essa análise enunciamos e provamos o Teorema 5.2 que estabelece a ordem de convergência $O(n^{1/d})$ do método da partição regular; muitos autores referem-se a essa razão de convergência, mas não encontramos nenhuma prova formal de tal enunciado;
- o Teorema 3.4 no qual se apóia a técnica da aceitação-rejeição é enunciado por alguns autores mas sem o oferecimento de prova; neste trabalho fornecemos uma prova através do determinante Jacobiano;
- além da simples descrição das técnicas, desenvolvemos uma grande quantidade de algoritmos, 30 ao todo; escrevemos esses algoritmos em pseudo-código, de forma que possam ser facilmente implementados em alguma linguagem de programação; ainda, todos esses algoritmos foram implementados e testados usando o pacote matemático MATLAB; uma listagem completa encontra-se no Apêndice B; acreditamos que esses algoritmos e programas possam ser bastante úteis ao leitor;
- as técnicas de Monte Carlo são utilizados em ampla variedade de aplicações, além da quadratura numérica; uma classe particularmente importante de aplicação é denominada genericamente de **simulação;** em brevíssima escala, o Apêndice A descreve alguns exemplos de aplicação do método Monte Carlo em simulação.

1.2 Algoritmos probabilísticos: Monte Carlo e Las Vegas.

Existe uma ampla classe de algoritmos, denominados **algoritmos probabilísticos** que, em algum momento e em maior ou menor quantidade, utilizam números aleatórios.

Esses algoritmos são geralmente divididos em duas amplas classes: Monte Carlo e Las Vegas (Motwani & Raghavan, 1995).

A expressão **Monte Carlo** (MC) é empregada para denominar algoritmos probabilísticos que produzem alguma solução para determinado problema com alguma margem de erro, geralmente previsível. O algoritmo de Miller-Rabin é um exemplo típico (Rabin, 1980). Esse algoritmo determina se um dado número n (ímpar e maior que 4) é primo ou não. A grosso modo, o algoritmo realiza um teste de divisibilidade (utilizando o pequeno teorema de Fermat) de n por um número a escolhido aleatoriamente entre 2 e $n-2$. Se n é primo, o algoritmo responde corretamente em 100% dos casos. No entanto, se n é composto, o algoritmo responde corretamente em 75% dos casos (Brassard & Bratley, 1996).

Já a expressão **Las Vegas** (LV) é empregada para denominar algoritmos probabilísticos que produzem, sempre, uma solução correta. O algoritmo clássico de ordenação denominado **Quicksort** (Hoare, 1962) é um exemplo típico. Esse algoritmo ordena uma lista de elementos recursivamente, particionando-a a partir de um elemento denominado pivô. Esse elemento pode ser escolhido de diversos modos, inclusive aleatoriamente. O algoritmo sempre ordena corretamente a lista, indepententemente de qual elemento-pivô for escolhido. A diferença entre os diversos modos de escolha desse pivô tem influência apenas no tempo de execução do algoritmo (Smith, 1989).

A denominação dos métodos é devida às cidades de Monte Carlo, no principado de Mônaco, às margens do mar Mediterrâneo e de Las Vegas, sudeste do estado de Nevada, Estados Unidos da América. Essas cidades são mundialmente famosas pelos seus luxuosos hotéis e cassinos. Neles, destaca-se a roleta, um dos mecanismos mais simples de produção de números aleatórios. A expressão Monte Carlo foi introduzido na literatura por Metropolis & Ulam (1949) e a expressão Las Vegas, por Babai (1979). Alguns autores, entretanto, não fazem distinção entre algoritmos do tipo MC ou LV, já que por definição o método Las Vegas é um método Monte Carlo com erro probabilístico zero.

1.3 Algoritmos probabilísticos: aplicações.

Algoritmos probabilísticos são utilizados em uma ampla série de aplicações, comentam-se as que seguem.

1.3.1 Simulação.

Modelos computacionais são usados para simular fenômenos naturais, que em maior ou menor grau são descritos por variáveis aleatórias. A simulação computacional

cobre vastos campos, como o estudo de física nuclear (a passagem de um nêutron através da parede de um reator está sujeita a colisões aleatórias com as moléculas da parede) e a pesquisa operacional (chamadas telefônicas chegam ao acaso a uma central de distribuição). Apesar dos modelos de simulação poderem oferecer os mais variados graus de complexidade, sua ideia básica é, de fato, simples. Olson *et alii* (1990) relatam simulações computacionais desenvolvidas por estudantes de ensino médio. No Apêndice A são descritos com maior detalhe alguns exemplos de aplicação do método MC em simulação.

1.3.2 Criptografia.

A codificação de mensagens utiliza números (denominados **chaves de criptografia**) para o embaralhamento de mensagens. Essas chaves são determinadas, em geral, a partir de números aleatórios. O algoritmo de criptografia RSA (Rivest, Shamir e Adleman) de chave pública é típico: chaves são determinadas a partir de números primos de 100 a 200 dígitos escolhidos aleatoriamente (Rivest *et alii*, 1973).

1.3.3 Experimentação.

Elementos dispostos ao acaso são utilizados em alguns experimentos, onde se quer estudar sistemas intrinsecamente desordenados. Por exemplo, uma rede de difração com raias espaçadas aleatoriamente (desordenadas) cria efeitos que são análogos à difração de raios X em cristais onde a desordem é devida a vibrações térmicas não-correlacionadas (Licínio *et alii*, 1998).

1.3.4 Programação.

Valores escolhidos aleatoriamente são uma boa fonte de dados para análise de eficiência de algoritmos computacionais. Por exemplo, pode-se testar um algoritmo de determinação de zeros polinomiais usando polinômios cujos coeficientes são escolhidos ao acaso (Camargo *et alii*, 1995), ou ainda, verificar um método de ordenação usando listas de valores escolhidos ao acaso. Matrizes com elementos aleatórios são amplamente usadas para teste de algoritmos em álgebra linear (Higham, 1996), principalmente pelo fato de serem, em geral, muito bem condicionadas (Edelman, 1988).

1.3.5 Tomada de decisão.

Geralmente, algoritmos devem tomar decisões a partir de critérios fixos e bem definidos, porém, ocasionalmente, critérios aleatórios são possíveis. Por exemplo, em teoria de grafos, denomina-se **corte** um conjunto de arestas que, se removidas, tornam o grafo não-conexo. O **corte-mínimo** de um grafo é o corte de menor cardinalidade. Uma **contração** é a união de vértices pela remoção de arestas comuns. Algumas técnicas de determinação do corte-mínimo de um grafo realizam contrações escolhidas de forma aleatória (Motwani & Raghavan, 1995).

1.3.6 Recreação.

Boa parte dos jogos está mais ou menos associada à disposição casual dos seus elementos: jogar dados, girar roletas, etc. Knuth (1998) descreve um algoritmo para gerar permutações aleatórias a partir de um conjunto ordenado de elementos. Isso equivale ao embaralhamento de um conjunto de cartas.

1.3.7 Amostragem.

Quando se tem uma quantidade muito grande de possibilidades em um sistema, eventualmente é muito difícil, ou mesmo impossível, examiná-las todas. No entanto, uma pequena amostra escolhida ao acaso pode representar muito bem o comportamento médio desse sistema. Por exemplo, não se pode entrevistar todos os eleitores de uma grande cidade, mas pode-se inferir resultados bastante satisfatórios se entrevistada uma amostra dessa população escolhida ao acaso.

1.3.8 Análise numérica.

Embora de natureza absolutamente não-estatística, alguns problemas analíticos são resolvidos utilizando-se o método MC. É o caso do problema da quadratura numérica multidimensional, que ocorre com freqüência em problemas envolvendo desde a estrutura atômica e molecular (Wagner & Ceperley, 1996) até análise econômica (Papageorgiou & Traub, 1997). Ou mesmo a resolução de determinados tipos de sistemas de equações (Hammersley & Handscomb, 1964).

1.4 O método Monte Carlo.

No campo da análise numérica, o método MC consiste, basicamente, na resolução de problemas analíticos através de amostragem aleatória. Pode ser usado tanto na resolução de problemas de formulação intrinsecamente analítica quanto na simulação de problemas puramente estatísticos. O método é usado especialmente naqueles casos em que o número de fatores incluídos no problema é tão grande que a sua solução analítica é demorada ou, em termos práticos, impossível. A idéia principal do método consiste em construir um modelo estocástico compatível com o problema analítico e simulá-lo inteiramente. Em todos os casos, um elemento de aleatoriedade é introduzido de acordo com regras definidas. Em seguida um número, eventualmente grande, de amostragens é coletado, seus resultados observados, e finalmente uma análise estatística é realizada (Fröberg, 1966).

Uma das principais vantagens do método é que, acima de tudo, problemas de grande complexidade podem ser tratados de forma mais simplificada e modificações e ajustes podem ser efetuados mais facilmente. Uma das principais desvantagens é, geralmente, a pequena precisão dos resultados, comparados com o grande número de simulações

necessárias.

Entre os métodos que tomam por base a avaliação de n pontos em um espaço amostral de dimensão d para a avaliação de uma solução aproximada, o método MC introduz um erro com decaimento na ordem $O(n^{-1/2})$, enquanto que, na ausência de alguma estrutura especial explorável, os outros métodos têm erros com decaimento na ordem $O(n^{-1/d})$, pelo menos.

1.5 Histórico.

Em 1777, George Louis Leclerc (1707-1788), Conde de Buffon, propôs e resolveu o seguinte problema. Dada uma agulha de comprimento L, jogada ao acaso sobre uma superfície plana horizontal, contendo linhas paralelas espaçadas regularmente entre si por uma distância $d > L$, determinar a probabilidade p de que a agulha intercepte uma dessas linhas. Admite-se que **ao acaso** signifique que a posição x com $0 \leq x \leq d$ do centro da agulha e a sua orientação φ com $0 \leq \varphi \leq 2\pi$ sejam variáveis aleatórias uniformes e independentes. Buffon (1777) mostrou que a probabilidade procurada é $p = \frac{2L}{\pi d}$.

Em 1812, uma nova luz sobre esse problema foi lançada por Pierre S. Laplace (1749-1827), que propôs a inversão da relação para $\pi = \frac{2L}{pd}$. O que Laplace obteve foi uma maneira de determinar o valor de π através de um processo estocástico (Laplace, 1886). Não se sabe se Laplace realizou ou não o procedimento experimental para determinar o valor de π. No entanto, muitos depois dele o fizeram. Beckmann (1974) comenta que tal atividade tornou-se um passatempo intelectual popular na época, como para um certo capitão Fox que jogou uma agulha sobre um tabuleiro milhares de vezes enquanto se recuperava de ferimentos sofridos durante a guerra civil americana (1861-1865). Estimativas do valor de e (número de Euler) podem ser feitas usando técnicas semelhantes (Mohazzabi, 1998).

Em 1899, John W. S. Rayleigh (1842-1919) mostrou que um passeio aleatório (*random walk*) unidimensional, sem paredes absorventes, poderia fornecer uma solução aproximada para uma equação diferencial parabólica.

Em 1901, William Thompson (1824-1907), Lord Kelvin, gerou 5000 trajetórias aleatórias para estudar o efeito de colisão elástica entre partículas e paredes de diversos formatos (Allen & Tildesley, 1987).

Em 1908, William S. Gosset (1876-1937) publicou, com o pseudômino *Student*, artigos referentes à estimativa dos coeficientes de correlação em sua distribuição t com a ajuda de amostragens experimentais, usando dados biométricos escritos em cartões

(Student, 1908a e Student, 1908b).

Em 1931, Andrey N. Kolmogorov (1903 – 1987) mostrou a relação entre os processos estocásticos de Markov e certas equações integro-diferenciais.

Durante a década de 1930, Enrico Fermi (1901-1958) trabalhou, usando procedimentos de simulação, na previsão teórica de resultados experimentais da difusão de nêutrons. Embora não tenha publicado artigos específicos sobre sua técnica de resolução de problemas, alguns autores reconhecem nesse trabalho o embrião do método Monte Carlo. É de salientar-se que, para realizar seus cálculos, Fermi dispunha apenas de um dispositivo mecânico, já que o primeiro computador só foi desenvolvido alguns anos mais tarde (Metropolis, 1987). Pelas demonstrações da existência de novos elementos radioativos, produzidos por irradiação de nêutrons, e pela descoberta correlata de reações nucleares induzidas por nêutrons lentos, Fermi recebeu em 1938 o prêmio Nobel de física.

A partir da década de 1940, o uso efetivo do método Monte Carlo como ferramenta de pesquisa foi impulsionado pelo esforço de guerra e a construção da bomba atômica. Nos estágios iniciais dessas investigações John von Neumann (1903-1957) e Stanislaw M. Ulam (1909-1984) referiam-se a certos métodos denominados **roleta russa** para tratar problemas analíticos através de seus equivalentes estatísticos. A denominação **método Monte Carlo** foi cunhada por Nicholas C. Metropolis (1914-) e usada como título em artigo fundamental (Metropolis & Ulam, 1949) descrevendo a simulação de problemas probabilísticos relativos à difusão aleatória de nêutrons em material físsil. Metropolis foi o responsável pela construção do computador MANIAC (*Mathematical And Numerical Integrator And Computer*), sucessor do ENIAC (*Electronic Numerical Integrator And Calculator*), nos laboratórios de Los Álamos, Estados Unidos da América. Computadores mais rápidos e confiáveis permitiram o aprimoramento do método MC, uma vez que gerar números aleatórios constituiu uma das primeiras tarefas a ser implementada nesses computadores primitivos. A resolução de equações de estado bidimensionais em problemas de muitos corpos por Metropolis *et alii* (1953) resultou no que que é hoje conhecido como **amostragem de importância**, técnica que reduz erros estatísticos e aprimora os resultados obtidos pelo método MC, amplamente utilizada atualmente (Anderson, 1986). O trabalho pioneiro de Metropolis é reconhecido através do prêmio que leva seu nome, oferecido anualmente pela *American Physics Society* à melhor tese de doutoramento na área da física computacional.

Durante a década de 1970, o desenvolvimento da teoria da complexidade computacional começou a providenciar um emprego mais preciso e racional do método Monte Carlo. Foi identificada uma classe de problemas de dimensão d para a qual o

tempo para a obtenção de uma solução exata cresce exponencialmente com d, como, por exemplo, a determinação do volume de um corpo convexo no espaço Euclidiano d-dimensional ou a determinação do permanente de uma matriz. No entanto, o método MC pode estimar uma solução para esses problemas, dentro de uma tolerância especificada, em um tempo polinomial em d (Pllana, 1999).

2 Geradores de números aleatórios.

2.1 Simulação de variáveis aleatórias.

Se X é uma variável aleatória **discreta** finita, que assume m valores $x_1, x_2, ..., x_m$, com probabilidades respectivas $p_1 := P(X = x_1)$, $p_2 := P(X = x_2)$, $p_m := P(X = x_m)$, o problema de **simular** a variável X consiste em gerar uma seqüência de n termos $(y_i) := (y_1, y_2, y_3, ..., y_n)$ de forma que:
(1) para todo $y_i, 1 \leq i \leq n$, exista um x_j, $1 \leq j \leq m$, tal que $y_i = x_j$;
(2) se q_j é o número de termos de (y_i) iguais a x_j, então, para todo $1 \leq j \leq m$,

$$\lim_{n\to\infty} \frac{q_j}{n} = p_j.$$

Se, por outro lado, X é uma variável aleatória **contínua**, cujos valores formam o intervalo $[a; b]$, com função densidade de probabilidade (FDP) f, o problema de simular a variável X consiste em gerar uma seqüência de n termos $(y_i) := (y_1, y_2, y_3, ..., y_n)$ de forma que:
(1) para todo y_i, $1 \leq i \leq n$, tenha-se $a \leq y_i \leq b$;
(2) para quaisquer a', b' com $a \leq a' \leq b' \leq b$, se $q(a', b')$ é o número de termos de (y_i) tais que $a' \leq y_i \leq b'$, então

$$\lim_{n\to\infty} \frac{q(a', b')}{n} = \int_{a'}^{b'} f(t)dt.$$

Neste Capítulo discute-se a simulação de uma variável aleatória U, uniformemente distribuída no intervalo $[0; 1]$, denominada **variável aleatória uniforme**. Esta variável tem FDP f definida por

$$f(u) := \begin{cases} 1, & \text{se } 0 \leq u \leq 1 \\ 0, & \text{em outro caso.} \end{cases}$$

Cada possível valor assumido por uma variável aleatória é denominado **ocorrência** ou **extração**. O processo que simula uma extração u da variável uniforme U é denominado **gerador uniforme** (GU) e a seqüência $(u_i) := (u_1, u_2, ..., u_n)$ obtida é denominada **seqüência uniforme**. O Capítulo 3 apresenta métodos de transformação da variável U para outras variáveis X (discretas ou contínuas) com outros tipos de distribuição de probabilidade.

De fato, algumas técnicas de simulação geram valores u_i nos intervalos $(0;1)$ ou $[0;1)$. Embora esse detalhe não influencie significativamente a teoria aqui exposta, o fato de esses valores poderem ou não ser 0 ou 1 merece atenção especial no desenvolvimento de programas, pois podem levar a resultados imprecisos (Gentle, 1998).

Existem, basicamente, três métodos distintos para construir um GU: tabelas, dispositivos físicos e algoritmos.

2.1.1 Tabelas.

Uma das muitas maneiras de obter um GU é através do uso de tabelas especialmente construídas para esse fim. Essas tabelas são compostas por uma coleção de dígitos decimais ou binários, com distribuição de probabilidade uniforme. Esses números são obtidos por sorteio em algum dispositivo físico (como, por exemplo, uma roleta) especialmente projetado. A Tabela 2.1 mostra 400 dígitos aleatórios uniformemente distribuídos. Por comodidade, os números foram dispostos em grupos de 5 dígitos.

86515	90795	66155	66434	56558	12332	94377	57802
69186	03393	42502	99224	88955	53758	91641	18867
41686	42163	85181	38967	33181	72664	53807	00607
86522	47171	88059	89342	67248	09082	12311	90316
72587	93000	89688	78416	27589	99528	14480	50961
52452	42499	33346	83935	79130	90410	45420	77757
76773	97526	27256	66447	25731	37525	16287	66181
04825	82134	80317	75120	45904	75601	70492	10274
87113	84778	45863	24520	19976	04925	07824	76044
84754	57616	38132	64294	15218	49286	89571	42903

Tabela 2.1: 400 dígitos aleatórios com distribuição uniforme (Sobol, 1983, p.64).

De tabelas de m dígitos como a Tabela 2.1 é possível obter a seqüência uniforme $(u_i) := (u_1, u_2, ..., u_n)$ com $n = m/k$ termos. Basta tomar grupos de k dígitos seqüenciais e formar uma seqüência de termos da forma $u_i := 0.u_{1+k(i-1)}u_{i+1}...u_{ik}$. Por exemplo, $(u_i) = (0.86515, 0.90795, 0.66155, ..., 0.42903)$ para $k = 5$. Desse modo, obtém-se uma seqüência de 80 termos distintos.

Pode-se obter outra seqüência com período maior $n = m$, tomando k dígitos seqüenciais e formando a seqüência de termos da forma $u_i := 0.u_iu_{i+1}...u_{i+k-1}$. Por exemplo, $(u_i) = (0.86515, 0.65159, 0.51590, ..., 0.42903)$, para $k = 5$. Desse modo,

obtém-se uma seqüência de período 400.

Muitas tabelas com números aleatórios, de todos os tamanhos, já foram publicadas. Knuth (1998) cita uma tabela publicada por Leonard H. C. Tippett em 1927 com 40.000 números "tomada ao acaso de relatórios de censo" (op.cit., p.3) , como sendo, provavelmente, a mais antiga. A Tabela de Rand Co. (1955) com 1.000.000 de números é, provavelmente, a mais extensa impressa. Abramowitz & Stegun (1965) mostram 2500 números extraídos dessa última.

O uso de tabelas de números aleatórios, assim como tabelas de logaritmos ou tabelas de funções trigonométricas, caíram rapidamente em desuso com o advento do computador, devido ao custo do armazenamento. Se a tabela é pequena, obtêm-se seqüências de período curto, e, se ela é grande, consome uma grande quantidade de memória. No entanto, o avanço tecnológico e a diminuição dos custos computacionais nos dias atuais permite o uso e armazenamento de bilhões de dígitos binários exaustivamente testados. Marsaglia (1995) produziu um CDROM com 4.8 bilhões de bits aleatórios gerados tanto por algoritmos quanto por diversas fontes de ruído, de circuitos eletrônicos a músicas "rap".

2.1.2 Dispositivo físico.

Pode-se obter números aleatórios usando algum dispositivo físico acoplado ao computador. O princípio de funcionamento de tal dispositivo é, por exemplo, o seguinte: um componente eletrônico (válvula, antena, etc.) capta ruído eletromagnético do meio ambiente; esse ruído é convertido em um sinal elétrico que oscila aleatoriamente em torno de um determinado nível de referência; um contador verifica quantas vezes o sinal cruza esse nível de referência em um certo intervalo de tempo; se esse número de vezes for par, gera-se um dígito binário 0, e, se for ímpar, gera-se um dígito binário 1; assim é gerada uma seqüência aleatória de dígitos binários $(b_i) = (0,0,1,0,1,1,1,0,1,0,0,1,1,0,1,0,1,0,1,...)$, podendo então ser convertida para números aleatórios em outras bases, se necessário.

Teoricamente, esse método pode gerar números realmente aleatórios, tal como uma roleta honesta. Porém, possíveis desvios na distribuição probabilística desses números são de difícil detecção, o que pode comprometer a qualidade dos valores gerados (Sobol, 1983). Pode-se contornar o problema fazendo, periodicamente, uma análise dos valores gerados e verificar se suas características são satisfatórias, mas o procedimento demanda tempo computacional.

É interessante notar que, se na seqüência original a probabilidade $p := P(b_i = 0)$ for diferente da probabilidade $q := P(b_i = 1) = 1 - p$, pode-se obter um nova seqüência (c_i) (mais homogênea) do sequinte modo, proposto por von Neumann em 1951 (Fröberg,

1966; Brassard & Bratley, 1996):
(1) a cada dois dígitos gerados da seqüência original b_i, b_{i+1}, obtém-se um dígito c_i da nova seqüência;
(2) descartam-se as combinações (00) e (11), cujas probabilidades são p^2 e q^2, respectivamente;
(3) substituem-se as combinações (01) e (10), cujas probabilidades são pq e qp, respectivamente, por (1) e (0).

Uma forte desvantagem dos dispositivos físicos é que os computadores que os utilizam são únicos e os programas não podem ser executados em outros computadores, perdendo-se assim a portabilidade dos algoritmos. Além disso, esses dispositivos tendem a ser mais lentos que o resto do programa, gerando uma perda de velocidade geral.

Outra característica indesejável desse tipo de gerador é sua não-reprodutibilidade. Muitas vezes, no processo de depuração e validação de um algoritmo, deseja-se utilizar uma seqüência de números aleatórios duas ou mais vezes, ou ainda, usar a mesma seqüência de valores em dois algoritmos diferentes. Geradores físicos não permitem repetição de seqüências de números aleatórios. Pode-se contornar esse problema armazenando uma determinada seqüência para posterior utilização, mas isso é equivalente à utilização de tabelas.

2.1.3 Algoritmos.

A idéia principal na geração de números aleatórios é a seguinte: não importa se uma seqüência de números **é realmente** aleatória ou não, basta que **pareça** aleatória. Isso significa que pode ser conveniente substituir roletas, dados, moedas, tabelas, etc, por algoritmos computacionais, na geração desses números.

Os números gerados por esses algoritmos são denominados **pseudo-aleatórios**. Isso porque esses números são gerados por um método intrinsecamente determinista, não-aleatório, porém parecem aleatórios (Scheid, 1991). A pseudo-aleatoriedade de uma seqüência de números é avaliada por uma série de testes teóricos e empíricos e um bom algoritmo deve gerar seqüências que sejam aprovadas nesses testes (Marsaglia, 1995; Knuth, 1998; Gentle, 1998 e Triola, 1999).

São diversos os tipos de algoritmos empregados na geração de números pseudo-aleatórios. Alguns deles serão vistos na seção 2.3. O algoritmo de um GU deve produzir uma seqüência (u_i) que simule uma extração da variável aleatória uniforme U. Em geral, os termos de (u_i) são obtidos fazendo $u_i := x_i/m$, onde x_i são termos da seqüência inteira (x_i) que assumem seus valores no intervalo $[0; m-1]$. A cada execução do GU, um novo valor de (x_i) é calculado a partir de k valores anteriores por

alguma fórmula recursiva

$$x_i \leftarrow F(x_{i-1}, x_{i-2}, ..., x_{i-k}),$$

onde F denota um conjunto de operações, tanto de aritmética computacional (adição, subtração, multiplicação, divisão, módulo, etc.) quanto de manipulação de registros (deslocamento de bits, inversão binária, truncamento de registro, etc.). Os k termos iniciais da seqüência (x_i) são denominados **sementes de simulação**. Algoritmos desse tipo são também denominados **seminuméricos**, pois requerem que tanto propriedades de cálculo numérico e simbólico quanto as características da máquina e da linguagem utilizada sejam consideradas pelos algoritmos (Knuth, 1998).

2.2 Qualidades esperadas de um Gerador Uniforme.

O sucesso dos métodos MC depende da qualidade dos números com os quais trabalham. Por qualidade entende-se, basicamente, a garantia de que as seqüências de números pseudo-aleatórios geradas pelos GU satisfaçam as propriedades de uniformidade, aleatoriedade, reprodutibilidade, portabilidade, período longo e velocidade. Algumas aplicações do método MC, como a integração numérica, são criticamente dependentes dessa qualidade; outras, como jogos, nem tanto. Em algumas aplicações, como certos problemas de simulação, as maiores incertezas são causadas pelos modelos matemáticos utilizados e não tanto pelos GU utilizados.

2.2.1 Uniformidade.

A uniformidade é a qualidade essencial do GU. Espera-se que qualquer seqüência $(u_i) := (u_1, u_2, ..., u_n)$ gerada pelo GU tenha função de distribuição de probabilidade uniforme, ou seja, preencha uniformemente o intervalo $[0;1]$. Se esse intervalo é dividido em k subintervalos de comprimento $1/k$, espera-se que a freqüência relativa com que cada subintervalo é preenchido pelos pontos u_i, $1 \leq i \leq n$, seja a mesma $(1/k)$ para todos os subintervalos.

A uniformidade também deve ser verificada em outras dimensões. A partir da seqüência $(u_1, u_2, ..., u_n)$ gerada pelo GU forma-se o vetor d-dimensional $\mathbf{u}_1$ tomando-se os seus d primeiros termos: $\mathbf{u}_1 := (u_1, u_2, ..., u_d)$. A seqüência $(\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_m)$ de m vetores d-dimensionais é formada tomando-se os termos subseqüentes: $\mathbf{u}_i := (u_{d(i-1)+1}, u_{d(i-1)+2}, ..., u_{di})$, $1 \leq i \leq m$. Espera-se que esses pontos preencham de forma uniforme o hipercubo unitário d-dimensional. Da mesma forma, se esse hipercubo é dividido em k sub-regiões retangulares, todas de volume $1/k$, espera-se que a freqüência relativa com que cada sub-região é preenchida pelos pontos $\mathbf{u}_i$ seja a mesma $(1/k)$ para todas essas sub-regiões. Isso está relacionado com o conceito de **discrepância,** descrito com maior detalhe na subseção 5.2.1. A Figura 2.1 mostra um preenchimento típico do espaço bidimensional por 200 pontos cujas coordenadas são

produzidas pelo gerador uniforme:

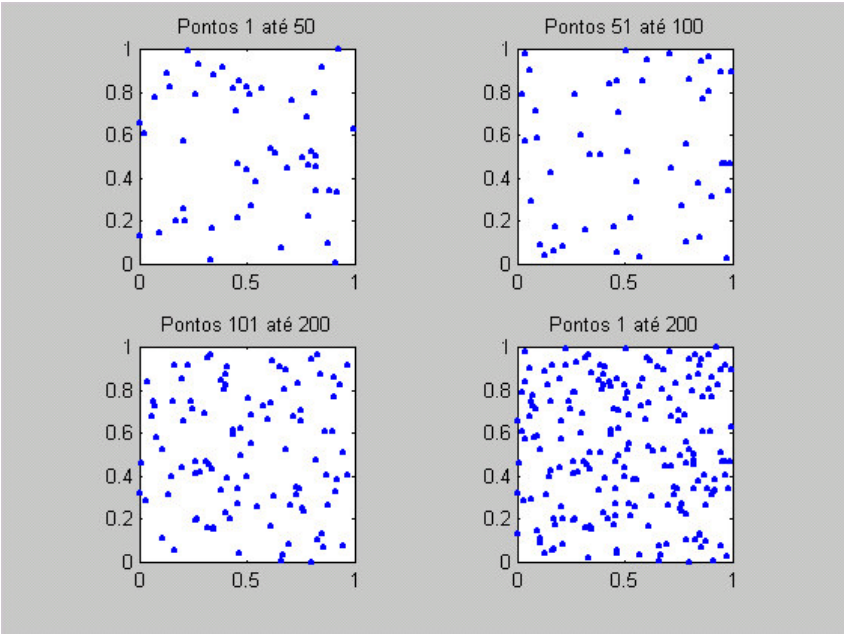

Figura 2.1: Preenchimento bidimensional de um gerador uniforme.

2.2.2 Aleatoriedade.

A aleatoriedade de uma seqüência $(u_i) := (u_1, u_2, ..., u_n)$ é medida pela correlação entre seus termos. Quanto menor for essa correlação, maior é a aleatoriedade. Uma correlação de ordem $k \geq 1$ é uma correlação entre os termos u_i e u_{i-k}, $k \leq i \leq n$, da seqüência.

Por exemplo, considere-se o passeio aleatório de uma partícula em um eixo unidimensional. A seqüência (u_i) pode ser usada para simular o movimento da partícula do seguinte modo: no passo i, $1 \leq i \leq n$, a partícula desloca-se para a direita se $u_i > 0.5$, ou para a esquerda, caso contrário; e ainda, se $u_{i+1} > 0.5$, então essa nova posição é aceita ou, caso contrário, rejeitada. Na ocorrência de correlação nula dos termos da seqüência, a partícula desloca-se com igual probabilidade tanto para a direita quanto para esquerda. No entanto, se existir uma correlação de ordem 1 entre os termos dessa seqüência, haverá uma tendência viciosa de deslocamento. Se, por exemplo, houver uma correlação positiva, então deslocamentos para a esquerda terão preferência sobre deslocamentos para a direita. No caso bidimensional, correlações de ordem 1 ou 2 são indesejáveis.

O cálculo de integrais d-dimensionais pelo método MC é particularmente sensível a

correlações de ordem $d-1$ (Gentle, 1998). Ferrenberg *et alii* (1992) e Grassberger (1993) discutem efeitos indesejáveis de correlações sutis em GU, considerados satisfatórios pela literatura.

2.2.3 Reprodutibilidade e portabilidade.

Devido a sua natureza recursiva, as seqüências geradas por um GU são essencialmente reprodutíveis. Basta que sejam usados os mesmos valores iniciais. Essa característica é desejável principalmente no decurso das depurações de programas, já que alguns erros computacionais ocorrem somente com combinações específicas de números. Essa característica dos GU não é compartilhada pelos dispositivos físicos, descritos na subseção 2.1.2.

No entanto, devido às características de cada máquina e de cada algoritmo, o mesmo GU com os mesmos valores iniciais, sendo executado em máquinas diferentes, pode não gerar a mesma seqüência de números, ou até mesmo nem ser executado. Essa característica define a portabilidade de um GU. É desejável que um GU seja portátil.

2.2.4 Período.

Em uma máquina de representação numérica binária de p bits, um conjunto finito de 2^p valores numéricos (estados) distintos são representáveis. Portanto aqui existe um conjunto finito de valores distintos gerados por um GU. geralmente, após a geração de L termos, a seqüência $(x_i) := (x_1, x_2, ..., x_L)$ retorna ao seu valor original $(x_{L+1} = x_1)$ e repete-se o ciclo. O valor L é denominado **período** do gerador. Um período finito L induz uma correlação de ordem L inevitável: $x_{L+i} = x_i,\ 1 \leq i$.

É desejável que o valor L seja o maior possível. Se o gerador produz uma seqüência (x_i) com $0 \leq x_i \leq m-1,\ 1 \leq i \leq n$, então o período desse gerador é dito **maximal** se $L = m$, isto é, se todos os valores no intervalo $[0;\ m-1]$ são gerados antes de haver repetição. Para alguns geradores, o conceito de período deve ser estendido, pois sementes de simulação diferentes podem gerar ciclos de período diferente.

O conhecimento do período do gerador utilizado é importante. Deve-se evitar exaurir o gerador, isto é, usar seqüências do tamanho de seu período. Em termos práticos, isso corresponde a não ultrapassar 10% do período, que deve ser $L > 10^9$ (Gentle, 1998).

2.2.5 Velocidade.

Sempre é desejável que os GU sejam rápidos. E, de fato, sua maioria é. A maior parte dos GU usa apenas alguns ciclos de máquina para gerar um número pseudo-aleatório, consumindo apenas uma fração do tempo total de cálculo de um programa. Em geral, a velocidade de determinado gerador não é preocupação importante, a menos que ele

seja extremamente lento ou o restante do programa seja extremamente rápido.

2.3 Algoritmos para um Gerador Uniforme.

Existe um grande número de algoritmos para um gerador uniforme (GU). Entre esses podem-se citar os métodos de Congruência Linear, Deslocamento de Registro, Fibonacci, Congruência Inversa, métodos combinados, etc. Alguns, como o Meio-de-Quadrado, têm interesse apenas histórico, enquanto outros, como o de Congruência Linear, são amplamente utilizados modernamente. Nesta seção, faz-se uma descrição desses algoritmos.

Quase todos os algoritmos GU utilizam fórmulas recursivas para a construção de seqüências de números pseudo-aleatórios. Os termos iniciais dessas seqüências, as sementes de simulação, são geralmente números inteiros e positivos, escolhidos pelo usuário ou pelo computador. Caso o número seja escolhido pelo computador, geralmente utiliza-se o relógio interno para fornecer números diferentes a cada execução do programa.

Uma condição importante em todos os algoritmos é que as operações aritméticas com inteiros devem ser feitas exatamente, sem arredondamentos. Em alguns ambientes de programação matemática, como MAPLE, por exemplo, essa preocupação é desnecessária pois o ambiente é simbólico e trabalha com aritmética de precisão estendida: as operações aritméticas são efetuadas com a quantidade de dígitos necessária.

Em outros ambientes de programação, esse detalhe deve ser levado em conta, pois existe uma limitação na representação numérica de números inteiros positivos. É importante conhecer o maior número inteiro positivo M tal que todos os inteiros do intervalo $[0;M]$ tenham representação exata na máquina. Supondo que a máquina utilize um registro de m dígitos binários para a representação de números inteiros, o valor de M é $2^m - 1$ (Higham, 1996). Por exemplo, a linguagem C usa um registro de 16 bits para o tipo **unsigned int** ($M = 2^{16} - 1 = 65535$), e um registro de 32 bits para o tipo **unsigned long** ($M = 2^{32} - 1 = 4294967295$) (Schildt, 1996). O MATLAB, que se apóia na aritmética padrão IEEE (1985), usa um registro de 53 bits ($M = 2^{53} - 1 = 9007199254740991$).

2.3.1 Gerador meio-de-quadrado.

Um dos primeiros algoritmos para obtenção de números pseudo-aleatórios foi sugerido por Jonh von Neumann por volta de 1946 (Metropolis, 1987).

A seqüência $(u_i) := (u_1, u_2, ..., u_n)$ de termos uniformemente distribuídos no intervalo

$(0;1)$ é obtida a partir da seqüência de inteiros $(x_i) := (x_1, x_2, ..., x_n)$, onde $1 \leq x_i \leq 10^m - 1$, com $1 \leq i \leq n$, é um número de m dígitos (decimais, na concepção original de von Neumann). O valor de m é escolhido de forma que $10^m - 1 \leq M$, onde M é o maior inteiro representável na máquina. Os termos u_i são calculados por $u_i = x_i/10^m$. Nesse algoritmo, os valores $u_i = 0$ e $u_i = 1$ não são obtidos.

Os termos x_i são calculados tomando-se os dígitos intermediários de x_{i-1}^2. Por exemplo, para números de 4 dígitos, se $x_1 = 1234$, então $x_1^2 = 01\underline{522}756$. O número seguinte na seqüência, x_2, é obtido tomando-se os 4 dígitos intermediários de x_1^2, ou seja: $x_2 = 5227$, e assim por diante. Continuando esse procedimento, obtém-se a seqüência $(x_i) = (1234, 5227, 3215, 3362, ...)$, e $(u_i) = (0.1234, 0.5227, 0.3215, 0.3362, ...)$.

Embora esse gerador pareça aceitável, foi logo abandonado por duas razões fundamentais. Primeiro porque a distribuição dos termos de (u_i) não é uniforme, tendo uma preponderância de pequenos valores entre os números gerados (Sobol, 1983). Segundo porque as seqüências não têm período maximal, apresentando, de fato, muitas subseqüências convergentes (Knuth, 1998).

Algoritmo 2.1: Gerador Meio-de-Quadrado (GMQ).
$U \leftarrow GMQ$
global X {variável global, semente de simulação}
$m \leftarrow 7$ {No MATLAB com $M = 2^{53} - 1$ tem-se $m := 7$}
$X \leftarrow X^2$
$X \leftarrow X \div 10^{\lfloor m/2 \rfloor}$ {trunca dígitos à direita. A operação $\div$ denota a divisão inteira}
$X \leftarrow X \bmod 10^m$ {trunca dígitos à esquerda}
$U \leftarrow X/10^m$

2.3.2 Gerador de congruência linear.

O gerador de congruência linear, proposto por Lehmer (1951), foi um dos primeiros a ter uso efetivo na simulação de variáveis pseudo-aleatórias, sendo até hoje um dos mais populares. As propriedades desse gerador foram e continuam sendo as mais estudadas, tanto teórica quanto empiricamente. A maior parte da teoria sobre geradores de números pseudo-aleatórios toma por base esse gerador. A seqüência (x_i) é obtida a partir da fórmula de recorrência

$$x_i := ax_{i-1} + c \pmod{m}.$$

Os termos u_i ,$1 \leq i \leq n$, da seqüência uniforme, são calculados por $u_i := x_i/m$. Se $c = 0$, o gerador é dito **multiplicativo.** Se $c \neq 0$, o gerador é dito **aditivo**. Se forem

utilizados, por exemplo, os valores $a := 12$, $c := 0$, $m := 31$ e $x_1 := 9$ (Gentle, 1998), obtém-se a seqüência

$$(x_i) = (9, 15, 25, 21, 4, 17, 18, 30, 19, 11, 8, 3, 5, 29, 7, \quad (2.1)$$
$$22, 16, 6, 10, 27, 14, 13, 1, 12, 20, 23, 28, 26, 2, 24, ...). \quad \#$$

Após o termo 24, a seqüência repete-se, pois $24 \times 12 \pmod{31} = 9$.

A escolha adequada de a, c e m define as características teóricas e empíricas do gerador. O período máximo do gerador é m se $c \neq 0$, e $m-1$ se $c = 0$. O valor de m é geralmente escolhido igual a 2^k, o que assegura que o tempo de processamento será mais rápido, ou igual a um número primo, usualmente do tipo $m = 2^k - 1$, um **número primo de Mersene**.

Se m é uma potência de 2, o período máximo do gerador multiplicativo é $m/4$, e é obtido para os valores de a tais que $a \equiv \pm 3 \pmod{8}$ (Knuth, 1998). Os dígitos da representação binária dos termos da seqüência (x_i) de tais geradores apresentam regularidade. Os bits menos significativos apresentam período 1, isto é, são sempre os mesmos. Os segundos bits menos significativos apresentam período 2. Os bits seguintes apresentam períodos respectivos $4, 8, 16, ...$ De modo geral os períodos dos bits estão relacionados às potências dos fatores primos de m, se m for composto (Gentle,1998).

Um número a é dito **raiz primitiva**, módulo m, se, e somente se, a é tal que o menor valor positivo de k com

$$a^k \equiv 1 \pmod{m}$$

é $m-1$. O gerador tem período maximal $m-1$ se, e somente se, m for um número primo e a for uma raiz primitiva, módulo m. No exemplo da seqüência (2.1), $a = 12$ é raiz primitiva $\pmod{31}$.

A estrutura dos termos da seqüência (x_i) é bastante rígida. Marsaglia (1968) estabeleceu que, se a seqüência $(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m)$ de pontos d-dimensionais fosse formada, como descrito na subseção 2.2.1, todos seriam uma combinação linear de d vetores base. Os pontos assim distribuídos repousam sobre planos $(d-1)$-dimensionais em número máximo de, aproximadamente, $m^{1/d}$ (Press *et alii*, 1992). A Figura 2.2 mostra a distribuição bidimensional dos pontos, cujas coordenadas são obtidas da seqüência (2.1).

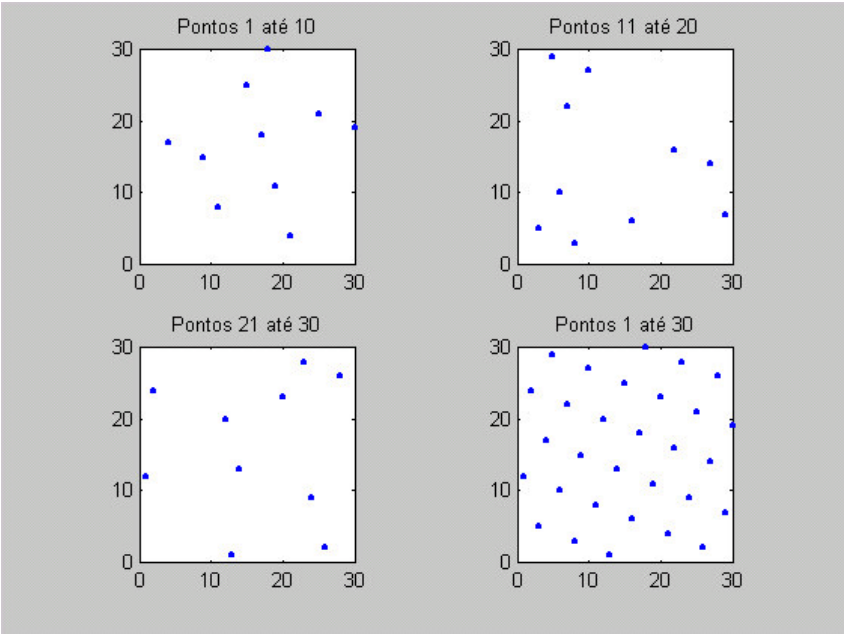

Figura 2.2: Preenchimento bidimensional pela seqüência (2.1).

Para que tal estrutura não interfira negativamente na qualidade do gerador, é necessário que a escolha de m e a satisfaça certos critérios analíticos e estatísticos (o valor de c é irrelevante se a e m forem apropriados). Knuth (1998) faz uma exaustiva descrição desses critérios, incluindo o denominado **Teste Espectral**, que verifica as distâncias entre planos $(d-1)$-dimensionais paralelos. No entanto, a literatura é inconclusiva a respeito desses valores. Os valores originalmente propostos por Lehmer (1951) são $a = 5^{17}$, $c = 0$ e $m = 2^{40}$,obtendo uma seqüência de período $m/4 = 2^{38}$. Park & Miller (1988) afirmam que um padrão mínimo de qualidade é obtido com os valores $a = 7^5 = 16807$ e $m = 2^{31} - 1 = 2147483647$. A função `rand` do MATLAB (versão 4) utiliza esses valores. Para computadores de pequeno porte, Scheid (1991) propõe $a = 25173$, $c = 13849$ e $m = 65536$, resultando um gerador de período maximal. Knuth (1998) propõe os valores $a = 48271$ e $m = 2^{31} - 1$. A função `rand()` do MAPLE (versão 5) utiliza os valores $a = 427419669081$, $c = 0$ e $m = 10^{12} - 11 = 999999999989$ (um número primo). Gentle (1998) chama a atenção para o fato de que os valores $a = 950706376$ e $m = 2^{31} - 1$ apresentam bons resultados, mas não podem ser implementados em algums ambientes computacionais, como o MATLAB e C, devido a erros de arredondamento descritos no início da seção 2.3. L'Ecuyer (1999) fez um apanhado geral dos valores de a, c e m e da qualidade dos geradores assim obtidos.

Algoritmo 2.2: Gerador de Congruência Linear (GCL).
$U \leftarrow GCL$
global X {variável global, semente de simulação}
$a \leftarrow 16807$; $c \leftarrow 1$; $m \leftarrow 2147483647$ {valores de Park & Miller (1988)}

$X \leftarrow aX + c \pmod{m}$
$U \leftarrow X/m$

2.3.3 Geradores combinados.

Também é possível combinar duas seqüências (x_i) e (y_i) de números pseudo-aleatórios obtidos a partir de geradores distintos, ou com sementes de simulação distintas, para obter outra seqüência (z_i). Essa seqüência é obtida pela relação

$$z_i := x_i * y_i \quad \text{para} \quad 1 \leq i \leq n,$$

onde $*$ representa, tipicamente, uma operação de adição ou multiplicação $(\mathrm{mod}\, m)$ ou ainda uma adição binária $(\mathrm{mod}\, 2)$.

Se os períodos L_X e L_Y de cada seqüência original são primos relativos, então o período da seqüência combinada será $L_Z := L_X \times L_Y$. Wichmann & Hill (1987) propuseram a combinação de três geradores de congruência linear de períodos distintos $L_1 \approx L_2 \approx L_3 \approx 10^4$, para a obtenção de outro gerador de período $L \approx 10^{12}$, como mostrado no algoritmo seguinte.

Algoritmo 2.3: Gerador de Congruência Linear Combinado (GCLC).
$U \leftarrow GCLC$
global X, Y, Z {variáveis globais, sementes de simulação}
$a_X \leftarrow 171$
$a_Y \leftarrow 172$
$a_Z \leftarrow 170$
$m_X \leftarrow 30269$
$m_Y \leftarrow 30307$
$m_Z \leftarrow 30323$
$X \leftarrow a_X X \bmod m_X$
$Y \leftarrow a_Y Y \bmod m_Y$
$Z \leftarrow a_Z Z \bmod m_Z$
$U \leftarrow \frac{X}{m_X} + \frac{Y}{m_Y} + \frac{Z}{m_Z}$
$U \leftarrow U - \lfloor U \rfloor$ {trunca parte inteira}

2.3.4 Geradores de congruência inversa.

De modo semelhante ao gerador de congruência linear, os geradores de congruência inversa produzem uma seqüência (u_i), a partir da seqüência (x_i), fazendo $u_i := x_i/m$. Cada termo de (x_i) é obtido de forma recursiva, como sugerido por Eichenauer & Lehn (1986):

$$x_i := a\overline{x}_{i-1} + c \pmod{m}.$$

Ou ainda de forma explícita, de acordo com Niederreiter (1994):

$$x_i := \overline{ai + c} \pmod{m}.$$

Nas duas formas, $\overline{s}$ denota o inverso de $s, (\mathrm{mod}\, m)$. Se s é um número inteiro positivo, $\overline{s}$ é o valor tal que $s\overline{s} \equiv 1(\mathrm{mod}\, m)$. Por exemplo, se $s = 5$ e $m = 7$, então $\overline{s} = 3$, pois $3 \times 5 \equiv 1(\mathrm{mod}\, 7)$. O valor de $\overline{s}$ pode ser calculado através do algoritmo estendido de Euclides para o Máximo Divisor Comum (MDC) de s e m. Se m é primo, logo MDC$(s,m) = 1$, existem $\overline{s}$ e $\overline{m}$ tais que $s\overline{s} + m\overline{m} = 1$.

Esse gerador parece ter boas propriedades (Gentle, 1998). A principal vantagem desse tipo de gerador é que pontos $\mathbf{u}_i := (u_i, u_{i-1}, ..., u_{i-d+1})$ preenchem o espaço d-dimensional de modo a evitar hiperplanos $(d-1)$-dimensionais como no caso dos geradores de congruência linear. A principal desvantagem é o tempo t de cálculo da inversão da ordem de $t \approx \log m$.

Algoritmo 2.4: Inverso Congruente (IC)**.**

$\overline{s} \leftarrow IC(s,m)$
$\mathbf{u} \leftarrow (1,0,m)$ {$\mathbf{u}$ é vetor de três elementos $u_1 \leftarrow 1$, $u_2 \leftarrow 0$, $u_3 \leftarrow m$}
$\mathbf{v} \leftarrow (0,1,x)$
$q \leftarrow \left\lfloor \frac{u_3}{v_3} \right\rfloor$
$\mathbf{w} \leftarrow \mathbf{u} - q\mathbf{v}$
enquanto $s_3 > 0$
    $\mathbf{u} \leftarrow \mathbf{v}$
    $\mathbf{v} \leftarrow \mathbf{w}$
    $q \leftarrow \left\lfloor \frac{u_3}{v_3} \right\rfloor$
    $\mathbf{w} \leftarrow \mathbf{u} - q\mathbf{v}$
fim
se $v_2 > 0$
    $\overline{s} \leftarrow v_2$
senão
    $\overline{s} \leftarrow v_2 + m$
fim

Algoritmo 2.5: Gerador de Congruência Inversa (GCI)**.**

$U \leftarrow GCI$
global X {variável global, semente de simulação}

$a \leftarrow 16807$
$c \leftarrow 1$
$m \leftarrow 2147483647$
$\overline{X} \leftarrow IC(X,m)$ {algoritmo 2.5: IC}
$X \leftarrow a\overline{X} + c \bmod m$
$U \leftarrow X/m$

2.3.5 Gerador de Fibonacci.

De modo semelhente ao gerador de congruência linear, o gerador de Fibonacci produz uma seqüência uniforme (u_i) a partir da seqüência (x_i) fazendo $u_i := x_i/m$, $1 \leq i \leq n$. Cada termo x_i é obtido a partir da combinação de alguns termos anteriores. Sua forma recursiva é a seguinte:

$$x_i = c_k x_{i-k} + c_{k-1} x_{i-k+1} + ... + c_1 x_{i-1} \pmod{m}. \tag{2.2}$$

Se m é primo, esse gerador pode ser relacionado com o polinômio primitivo

$$P(z) = z^k + c_1 z^{k-1} + ... + c_{k-1} z + c_k,$$

sobre o corpo de Galois $\mathbf{G}(m)$ definido sobre os inteiros $0, 1, ..., m-1$ com adição e multiplicação usuais seguidas de redução $\bmod m$. O polinômio é dito **primitivo** se é irredutível (não pode ser fatorado sobre $\mathbf{G}(m)$) e se possui uma raiz que é elemento primitivo do corpo com m^k elementos (Knuth, 1998). A Teoria de Números relacionada a esse polinômio estabelece que, se os primeiros k termos de (x_i) não são todos nulos e o polinômio é primitivo, então o período da seqüência definida por (2.2) tem período $m^k - 1$ (Gentle, 1998).

A literatura tem estudado esse tipo de gerador com algum interesse. Vários módulos foram propostos, inclusive $m = 2^{31} - 1$. Valores de k desde $k = 5$ (L'Ecuyer *et alii*, 1993) até $k = 1279$ (Mascagni *et alii*, 1995) foram propostos. Nesse último caso, todos os c_i são nulos, exceto $c_{1279} = c_{1063} = 1$. Geralmente, os primeiros k termos da seqüência (x_i), as sementes de simulação, são obtidos por outro gerador. Se $m = 2^p$, o período desse gerador é, no máximo, $(2^k - 1)2^{p-1}$ (Brent, 1994). Segundo o manual *on line* do MATLAB (versão 5), sua função `rand` utiliza um gerador desse tipo com $k = 35$ e período 2^{1492}. Knuth (1998) propõe $k = 55$, com todos os c_i nulos, exceto $c_{55} = c_{24} = 1$ e $m = 2^{31} - 1$. O algoritmo abaixo implementa a sugestão de Knuth.

Algoritmo 2.6: Gerador de Fibonacci (GF)**.**
$U \leftarrow GF$
global X {variável global, seqüência com 55 termos $x_1, x_2, ..., x_{55}$}

```
global j            {variável global, inicializada com j := 24}
global k            {variável global, inicializada com k := 55}
m ← 2147483647
X_k ← X_j + X_k  mod m
U ← X_k/m
j ← j − 1
se j = 0
    j ← 55
fim
k ← k − 1
se k = 0
    k ← 55
fim
```

A modificação $x_i := x_{i-j}x_{i-k} \pmod{m}$, com $i > k > j$, foi proposta por Marsaglia (1985). Esse gerador apresenta período $2^{k-1}2^{p-3}$, que é 1/4 do correspondente gerador aditivo. No entanto, seu desempenho nos testes estatísticos foi considerado superior.

2.3.6 Gerador de deslocamento de registro.

No Gerador de Deslocamento de Registro, proposto inicialmente por Tausworthe (1965) e generalizado por Lewis & Payne (1973), uma seqüência uniforme (u_i) é obtida a partir da manipulação de sua representação binária (registro computacional).

Inicialmente é gerada uma seqüência $(b_j) := (b_1, b_2, ..., b_m)$ de dígitos binários pseudo-aleatórios com distribuição uniforme, isto é, com probabilidade $P(b_j = 0) = P(b_j = 1) = 0.5$, $1 \le j \le m$. Cada termo de (b_j) é obtido a partir de uma relação de recorrência semelhante à (2.2)

$$b_i = c_k b_{i-k} + c_{k-1} b_{i-k+1} + ... + c_1 b_{i-1} \pmod 2.$$

Se os coeficientes c_i são os de um polinômio primitivo $\pmod 2$, então a seqüência (b_j) tem período maximal $2^k - 1$. Para melhorar a eficiência computacional, para c_i são, geralmente, escolhidos os coeficientes de um trinômio primitivo $\pmod 2$ como, por exemplo, $P(z) = z^4 + z + 1$. Assim, a recorrência fica mais simples com $b_i = b_{i-4} + b_{i-1} \pmod 2$. Uma lista de polinômios primitivos $\pmod 2$ com o menor número de termos e grau $k \le 168$ é dada em Stahnke (1973). Outra lista com todos os polinômios primitivos com grau $k \le 10$ é dada em Press *et alii* (1992).

Cada termo de (u_i) é obtido a partir de uma subseqüência de k termos consecutivos de (b_i),

$$u_i := (0.b_i b_{i+1} ... b_{i+k-1})_2.$$

Esse gerador, se implementado em código de máquina, é excepcionalmente rápido, uma vez que as funções de deslocamento de registro fazem parte do conjunto de instruções dos processadores. Por exemplo, se $k = 4$, $(c_i) = (1,0,0,1)$, os primeiros 4 termos de (b_i) são $1,0,1,0$. Pelo método descrito acima obtêm-se os demais termos de (b_i):

$$\begin{aligned}
b_5 &= c_1b_1 + c_2b_2 + c_3b_3 + c_4b_4 \pmod 2 = 1\times 1 + 0\times 0 + 0\times 1 + 1\times 0 \pmod 2 = 1,\\
b_6 &= c_1b_2 + c_2b_3 + c_3b_4 + c_4b_5 \pmod 2 = 1\times 0 + 0\times 1 + 0\times 0 + 1\times 1 \pmod 2 = 1,\\
b_7 &= c_1b_3 + c_2b_4 + c_3b_5 + c_4b_6 \pmod 2 = 1\times 1 + 0\times 0 + 0\times 1 + 1\times 1 \pmod 2 = 0,\\
b_8 &= c_1b_4 + c_2b_5 + c_3b_6 + c_4b_7 \pmod 2 = 1\times 0 + 0\times 1 + 0\times 1 + 1\times 0 \pmod 2 = 0,\\
&\vdots
\end{aligned}$$

resultando os $2^4 - 1 = 15$ primeiros termos de $(b_i) = (1,0,1,0,1,1,0,0,1,0,0,0,1,1,1,....)$, antes de repetirem-se. Assim, os primeiros termos da seqüência (u_i) são

$$\begin{aligned}
u_1 &= 0.1010_2 = 0.6250,\\
u_2 &= 0.0101_2 = 0.3125,\\
u_3 &= 0.1011_2 = 0.6875,\\
u_4 &= 0.0110_2 = 0.3750,\\
&\vdots
\end{aligned}$$

Em uma implementação computacional, os dígitos binários de (c_i) e (b_i) podem ser armazenados, respectivamente, em uma constante c (no exemplo, $c = 0.1001_2 = 0.5625$) e no último termo de (u_i). O algoritmo descrito abaixo usa essa técnica.

Algoritmo 2.7: Gerador de Deslocamento de Registro (GDR)**.**
$U \leftarrow GDR$
{inicialização}
global U {variável global, semente de simulação, diferente de zero}
$c \leftarrow 0.31415926535898$
$t_c \leftarrow c$
$t_u \leftarrow U$

$bit_t \leftarrow 0$
$bit_v \leftarrow 1$
{determinação do próximo bit}
enquanto $t_c > 0$ ou $t_u > 0$
    $t_c \leftarrow 2t_c$
    se $t_c \geq 1$
        $bit_c \leftarrow 1$
        $t_c \leftarrow t_c - bit_c$
    senão
        $bit_c \leftarrow 0$
    fim
    $t_u \leftarrow 2t_u$
    se $t_u \geq 1$
        $bit_u \leftarrow 1$
        $t_u \leftarrow t_u - bit_u$
    senão
        $bit_u \leftarrow 0$
    fim
    $bit_t \leftarrow (bit_c + bit_c \times bit_t) \bmod 2$
    $bit_v \leftarrow bit_v/2$
fim
{deslocamento do bit}
$U \leftarrow 2U$
se $U \geq 1$
    $U \leftarrow U - 1$
fim
$U \leftarrow U + bit_t \times bit_v$

L'Ecuyer (1996) estudou a combinação (como visto na subseção 2.3.3) de geradores desse tipo para formar um novo gerador de período maior.

3 Simulação de variáveis aleatórias com distribuições de probabilidade discretas e contínuas.

A simulação de uma variável aleatória X com qualquer tipo de distribuição de probabilidade é feita a partir da transformação de uma variável aleatória uniforme U. Neste capítulo, são descritas as técnicas genéricas para simulação de variáveis aleatórias com distribuição discreta (seção 3.2) e contínua (seção 3.3). Ainda são descritas algumas técnicas para simulação de variáveis aleatórias com distribuições específicas, Exponencial e Normal, (seção 3.4).

3.1 Distribuições discretas.

Uma variável aleatória X é dita **discreta** finita quando assume n valores $x_1, x_2, ..., x_n$ com as probabilidades $p_i := P(X = x_i)$, $1 \leq i \leq n$, satisfazendo $\sum_{i=1}^{n} p_i = 1$.

3.1.1 Caso geral (distribuição genérica).

Um algoritmo para a simulação de uma variável discreta com distribuição de probabilidade qualquer toma por base o seguinte teorema:

Teorema 3.1. *Seja* $\mathbf{Q} := (q_0, q_1, q_2, q_3, ..., q_n)$ *com* $q_0 := 0$, $q_1 := q_0 + p_1$, $q_2 := q_1 + p_2$, $q_3 := q_2 + p_3$, ..., $q_n := q_{n-1} + p_n = 1$, *o vetor das probabilidades cumulativas de* X. *Se* U *é uma variável aleatória uniforme, então* $p_i = P(q_{i-1} \leq U < q_i)$, $1 \leq i \leq n$.

Demonstração. Como U é uniformemente distribuída no intervalo $[0;1]$, a probabilidade $P(q_{i-1} \leq U < q_i)$ é dada por $P(q_{i-1} \leq U < q_i) = \int_{q_{i-1}}^{q_i} 1dt = q_i - q_{i-1} = p_i$. □

O algoritmo seguinte simula uma extração da variável aleatória uniforme I, que assume seus valores em $\{1, 2, ..., n\}$ com probabilidades $P(I = 1) = p_1$, $P(I = 2) = p_2$, ..., $P(I = n) = p_n$.

Algoritmo 3.1: Gerador com Distribuição Discreta Genérica (DDG).

$I \leftarrow DDG(p)$
$U \leftarrow GU$ {gerador uniforme}
$I \leftarrow 1$ {contador}
$q \leftarrow p_I$ {cumulativo}
enquanto $U > q$
    $I \leftarrow I + 1$ {próximo intervalo}
    $q \leftarrow q + p_I$

fim

É interessante notar que nesse algoritmo o número T de execuções do laço é determinado pelo vetor de probabilidades $\mathbf{p} := (p_1, p_2, ..., p_n)$, onde $p_1 = P(T = 0)$, $p_2 = P(T = 1)$, ..., $p_n = P(T = n - 1)$. Assim, o número médio $\langle T \rangle$ de execução do laço é dado por $\langle T \rangle = \sum_{i=1}^{n} (i-1)p_i$. Portanto, $\langle T \rangle$ pode ser minimizado se p for ordenado de forma decrescente. Por exemplo, se $\mathbf{p} = (0.1, 0.2, 0.3, 0.4)$, tem-se $\langle T \rangle = 2$, enquanto que, se $\mathbf{p} := (0.4, 0.3, 0.2, 0.1)$, tem-se $\langle T \rangle = 1$.

A Tabela 3.1 mostra as freqüências obtidas e esperadas na simulação de 1000 extrações da variável I que assume os valores $1, 2, 3, 4$, com probabilidades respectivas $0.1, 0.2, 0.3, 0.4$.

I	Freq. Obtida	Freq. Esperada
1	81	100
2	215	200
3	299	300
4	405	400

Tabela 3.1: Simulação de 1000 extrações da variável I.

3.1.2 Caso particular (distribuição uniforme).

Um caso particular da distribuição descrita acima é o da variável aleatória X discreta, que assume os valores $1, 2, 3, ..., n$ com probabilidades iguais: $P(X = 1) = P(X = 2) = , ..., = P(X = n) = \frac{1}{n}$. A simulação de uma variável X com essa distribuição é imediata; basta fazer

$$X = 1 + \lfloor nU \rfloor. \tag{3.1}$$

Teorema 3.2. *A variável X calculada por (3.1) assume os valores $1, 2, 3, ..., n$ com probabilidades iguais.*

Demonstração. A variável U é uniformemente distribuída no intervalo $[0; 1)$. E também nU é uniformemente distribuída no intervalo $[0; n)$. Desse modo, $\lfloor nU \rfloor$ assume os valores $0, 1, 2, ..., n - 1$ com iguais probabilidades e, portando, $X = 1 + \lfloor nU \rfloor$ assume os valores $1, 2, 3, ..., n$ com iguais probabilidades. □

O algoritmo decorrente do Teorema 3.2 é extremamente simples, mas importante, pois simula variáveis aleatórias que ocorrem com grande freqüência, como o resultado

do lançamento de um dado homogêneo, por exemplo. Muitas linguagens de programação possuem rotinas com essa finalidade específica, como a linguagem C, com a rotina `random(n)`, e o MAPLE, com a rotina `rand(1..n)`.

O algoritmo seguinte simula uma extração da variável aleatória X que assume os valores $1, 2, 3, ..., n$ com iguais probabilidades:

Algoritmo 3.2: Gerador com Distribuição Discreta Uniforme (DDU).
$X \leftarrow DDU(n)$
$X = 1 + \lfloor n \times GU \rfloor$

A Tabela 3.2 mostra as freqüências obtidas e esperadas na simulação de 600 extrações da variável X que assume os valores $1, 2, 3, 4, 5, 6$ com iguais probabilidades, como num dado homogêneo, usando o algoritmo DDU.

X	Freq. Obtida	Freq. Esperada
1	89	100
2	104	100
3	106	100
4	96	100
5	96	100
6	109	100

Tabela 3.2: Simulação de um dado homogêneo.

3.2 Distribuições contínuas genéricas.

Seja X uma variável aleatória contínua, cujos valores formam o intervalo $[a; b]$, e tendo uma função densidade de probabilidade f. A função f é não-negativa em $[a; b]$ e nula fora desse intervalo. A FPC de X, $F : (-\infty; \infty) \to [0; 1]$, dada por

$$F(x) := P(X \leq x),$$

tem as propriedades

$$F(x) = \int_{-\infty}^{x} f(t)dt = \begin{cases} 0, & \text{se } x \leq a, \\ p, & \text{com } 0 \leq p \leq 1, \text{ se } a \leq x \leq b, \\ 1, & \text{se } b \leq x. \end{cases} \tag{3.2}$$

Existem dois métodos genéricos para a simulação de variáveis desse tipo que são particularmente simples e são aqui descritos: o método da inversão e o método da aceitação-rejeição (Knuth, 1998; Gentle, 1998). As deduções, que aqui vão ser feitas, podem ser estendidas para o caso em que o intervalo de valores de X é $(-\infty;\infty)$ em vez de $[a;b]$.

3.2.1 Método da inversão.

A técnica no método da inversão consiste em simular a variável aleatória X com FPC F mediante o uso da variável aleatória uniforme U, e a inversão de F:

$$Y := F^{-1} \circ U. \tag{3.3}$$

Para inverter F procede-se assim: como F é estritamente crescente no intervalo $[a;b]$ e $F([a;b]) = [0;1]$, define-se $F^{-1} : (-\infty;\infty) \to [0;1]$ da maneira usual no intervalo $[0;1]$; se $u < 0$, põe-se $F^{-1}(u) := F^{-1}(0) = a$ e, se $1 < u$, põe-se $F^{-1}(u) := F^{-1}(1) = b$.

Teorema 3.3. *A variável aleatória definida em (3.3) tem FDP f, isto é, a FDP de Y é igual a FDP de X.*

Demonstração. A demonstração é trivial. Basta observar que

$$P(Y \leq y) = P(F^{-1}(U) \leq y) = P(X \leq y) = \int_{-\infty}^{y} f(t)dt.$$

A igualdade do meio vem de que, se u e x são tais que $x = F^{-1}(u)$, então

$$F^{-1} \circ U(u) = F^{-1}(u) = x. \square$$

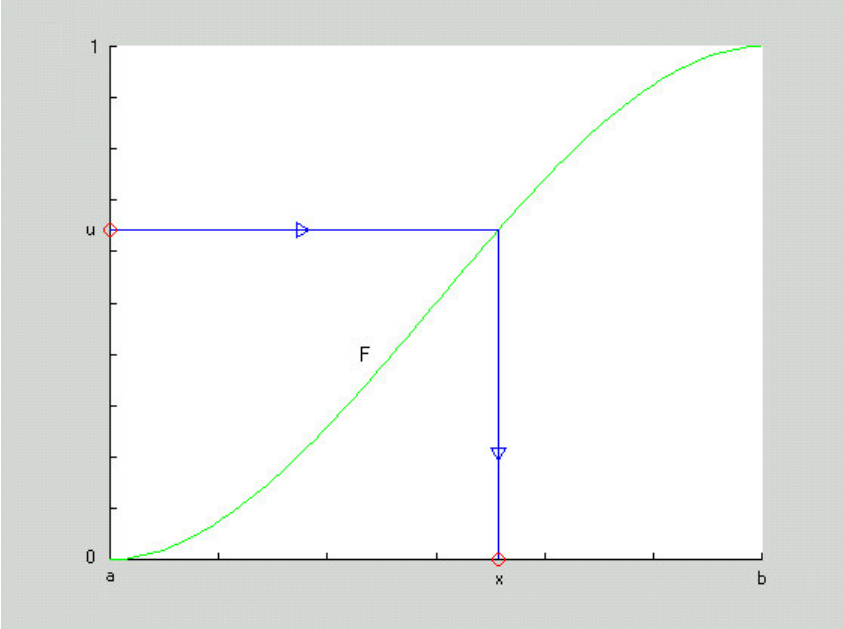

Figura 3.1: O método da inversão: $X = p^{-1}(U)$.

A Figura 3.1 mostra geometricamente a obtenção de X a partir de U.

O algoritmo para obter essa distribuição é imediato. A função F^{-1} deve ser passada como argumento.

Algoritmo 3.3: Gerador com Distribuição Contínua pelo método de Inversão (DCI).
$X \leftarrow DCI(F^{-1})$
$U \leftarrow GU$ {gerador uniforme}
$X \leftarrow F^{-1}(U)$

Seja, por exemplo, a variável aleatória X com FDP $f(x) = 2x$, $x \in [0;1]$. A FPC F correspondente é dada por

$$F(x) = \int_{-\infty}^{x} f(t)dt = \begin{cases} 0, & x < 0 \\ x^2, & x \in [0;1] \\ 1, & x > 1. \end{cases}$$

Essa FPC é facilmente invertida, de modo que a parte principal da fórmula de inversão é $x = F^{-1}(u) = \sqrt{u}$, $u \in [0;1]$. Para simular essa variável, basta simular uma extração de U por um GU e fazer $x := \sqrt{u}$. A Tabela 3.3 mostra as freqüências obtidas e esperadas na simulação de 1000 extrações da variável X.

Intervalo	Freq. Obtida	Freq.Esperada
[0.0; 0.1)	7	10
[0.1; 0.2)	26	30
[0.2; 0.3)	41	50
[0.3; 0.4)	66	70
[0.4; 0.5)	101	90
[0.5; 0.6)	117	110
[0.6; 0.7)	126	130
[0.7; 0.8)	156	150
[0.8; 0.9)	167	170
[0.9; 1.0]	193	190

Tabela 3.3: Simulação da variável X com FDP f dada por $f(x) = 2x$.

Esse método apresenta uma desvantagem importante. Embora seja geralmente pouco trabalhoso obter a FPC F a partir da FDP f, pode ser difícil, ou até mesmo impossível, obter de forma analítica uma fórmula explícita para sua inversa F^{-1}. Por exemplo, a FPC Normal padronizada F, dada por

$$F(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}}\, dt,$$

não pode ser expressa em termos de funções elementares, podendo seus valores somente serem calculados numericamente. Nesse caso, não se pode obter uma fórmula explícita para $x := F^{-1}(u)$, mas apenas fórmulas aproximadas por interpolação.

Em casos como esse, pode ser aconselhável a resolução numérica de $u = F(x)$ por alguma técnica como, por exemplo, a de Newton-Raphson: o valor de x é determinado iterativamente como o limite da seqüência (x_i) dada por

$$x_i = x_{i-1} - \frac{F(x_{i-1}) - u}{f(x_{i-1})}.$$

3.2.2 Método da aceitação-rejeição.

Segundo Sobol (1983), a origem do método da aceitação-rejeição é devida a J. von Neumann. Ele consiste em simular uma seqüência de extrações da variável aleatória X com FDP f desejada, aceitando alguns termos das extrações de uma variável aleatória

Z com uma FDP g, com base no seguinte teorema:

Teorema 3.4. *Sejam f e g duas FDP, como em (3.2), sendo $g > 0$, e g' contínua em $[a;b]$. Seja $(u_i) = (u_1, u_2, ..., u_m)$ uma seqüência de m extrações da variável aleatória uniforme U. Seja $(z_i) = (z_1, z_2, ..., z_m)$ uma seqüência de m extrações da variável aleatória Z, independente de U, com FDP g. Seja c constante positiva tal que $cg(t) \geq f(t)$ para todo t em $[a;b]$. Se a seqüência $(x_i) := (x_1, x_2, ...x_n)$ de n extrações da variável aleatória X é obtida aceitando os termos da seqüência (z_i) que satisfazem $u_i cg(z_i) < f(z_i)$ para $1 \leq i \leq m$, e rejeitando os demais, então*
(i) a FDP de X é f;
(ii) a probabilidade média de aceitação dos termos da seqüência (z_i) é $\frac{1}{c}$.

Demonstração. Considerem-se variáveis aleatórias S e T tais que $(s,t) := (z,\ ucg(z))$. Os pontos (s,t) estão localizados na região $\{(s,t) \mid a \leq s \leq b,\ 0 \leq t \leq cg(s)\}$ (região sob o gráfico de cg). A FDP conjunta de U e Z é $f_{U,Z} = 1 \bullet g(z)$. De acordo com o exposto na subseção 3.3.1 (abaixo), a FDP conjunta de S e T é

$$f_{S,T} = \frac{1}{\left| \frac{\partial(s,t)}{\partial(u,z)} \right|} f_{U,Z}.$$

Mas

$$\frac{\partial(s,t)}{\partial(u,z)} = \begin{vmatrix} \frac{\partial s}{\partial u} & \frac{\partial s}{\partial z} \\ \frac{\partial t}{\partial u} & \frac{\partial t}{\partial z} \end{vmatrix} = \begin{vmatrix} 0 & 1 \\ cg(z) & ucg'(z) \end{vmatrix} = -cg(z).$$

Assim, $f_{S,T} = \frac{1}{c}$ é constante. Portanto, a variável aleatória (S,T) tem distribuição uniforme na região sob o gráfico de cg. Assim sendo, aceitar z_i tal que $u_i cg(z_i) \leq f(z_i)$ equivale a aceitar z_i como termo de (x_i) se e somente se o ponto (s_i, t_i) pertence à região $\{(s,t) \mid a \leq s \leq b,\ \ 0 \leq t \leq f(s)\}$ (região sob o gráfico de f). Logo, a probabilidade P_A de aceitação dos termos da seqüência (z_i) como termos da seqüência (x_i) é igual à razão entre a área sob o gráfico de cg e a área sob o gráfico de f:

$$P_A = \frac{\int_a^b f(t)dt}{\int_a^b cg(t)dt} = \frac{1}{c},$$

o que demonstra (ii).

Para qualquer s em $[a;b]$, a probabilidade $P(a < X < s)$ é igual à razão entre as áreas das regiões $\{(x,t) \mid a \leq x \leq s,\ 0 \leq t \leq f(x)\}$ e $\{(x,t) \mid a \leq x \leq b,\ 0 \leq t \leq f(x)\}$, isto é,

$$P(X < s) = P(a < X < s) = \frac{\int_a^s f(t)dt}{\int_a^b f(t)dt} = \int_a^s f(t)dt = \int_{-\infty}^s f(t)dt$$

que mostra (i).□

A Figura 3.2 mostra, geometricamente, esse resultado.

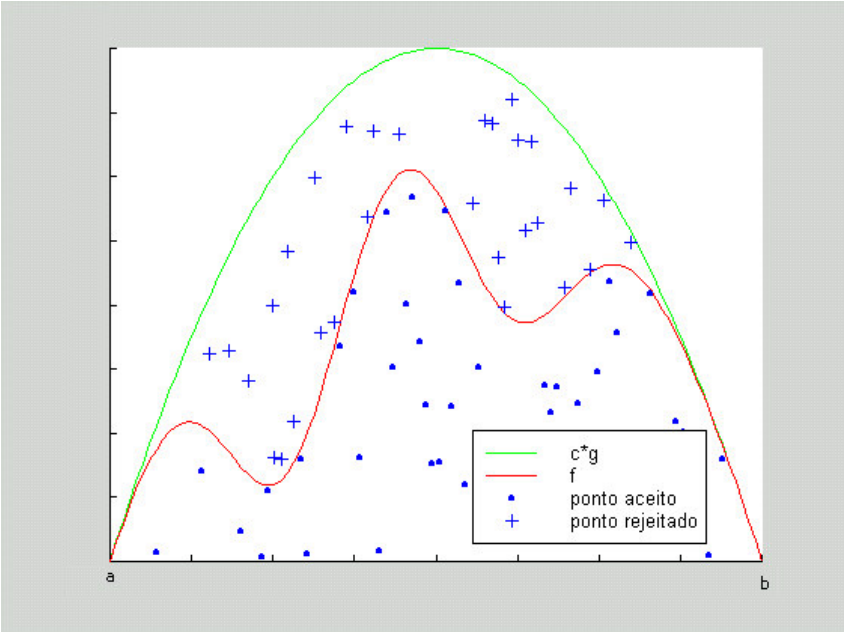

Figura 3.2: O método da aceitação-rejeição.

A justificativa desse método consiste em ser possível simular uma variável aleatória Z de modo mais simples (com, por exemplo, o método da inversão) do que simular a variável aleatória X diretamente. A FDP g deve ser tal que a constante c seja o mais próximo possível de 1. Desse modo, minimiza-se a quantidade de termos rejeitados. No algoritmo seguinte, GZ é uma rotina que deve ser escrita pelo usuário, que fornece Z com a FDP g desejada.

Algoritmo 3.4: Gerador com Distribuição Contínua pelo método da Aceitação-Rejeição (DCAR).
$X \leftarrow DCAR(f, g, c, GZ)$
$Z \leftarrow GZ$ {gerador da seqüência Z}
enquanto $GU \times c \times g(Z) > f(Z)$ {... se Z for rejeitado}
    $Z \leftarrow GZ$ {gerador da seqüência Z}
fim
$X \leftarrow Z$

Por exemplo, deseja-se simular a variável X com FDP f dada por

$$f(x) = \begin{cases} 4xe^{-2x}, & x \geq 0 \\ 0, & x < 0, \end{cases}$$

(distribuição gama), usando o método de aceitação-rejeição. Embora f esteja definida no intervalo $[-\infty;\infty)$, é possível, para esse exemplo, desconsiderar valores de $X < 0$ (pois $F(x) = 0,$ se $x < 0$) e também $X > 5$ (pois $F(x) > 0.9995,$ se $x > 5$). Desse modo, pode-se obter X a partir de outra variável Z com FDP g linear dada por

$$g(x) = \begin{cases} 0, & x < 0 \\ \frac{2}{5}\left(1 - \frac{1}{5}x\right), & x \in [0;5] \\ 0, & x > 5, \end{cases}$$

que pode ser obtida pelo método de inversão, fazendo $Z = 5\left(1 - \sqrt{U}\right)$, onde U é a variável aleatória uniformemente distribuída no intervalo $[0;1]$. Fazendo $c = 2.1$, tem-se $cg(x) > f(x)$, para todo $0 \leq x \leq 5,$ como mostra a Figura 3.3.

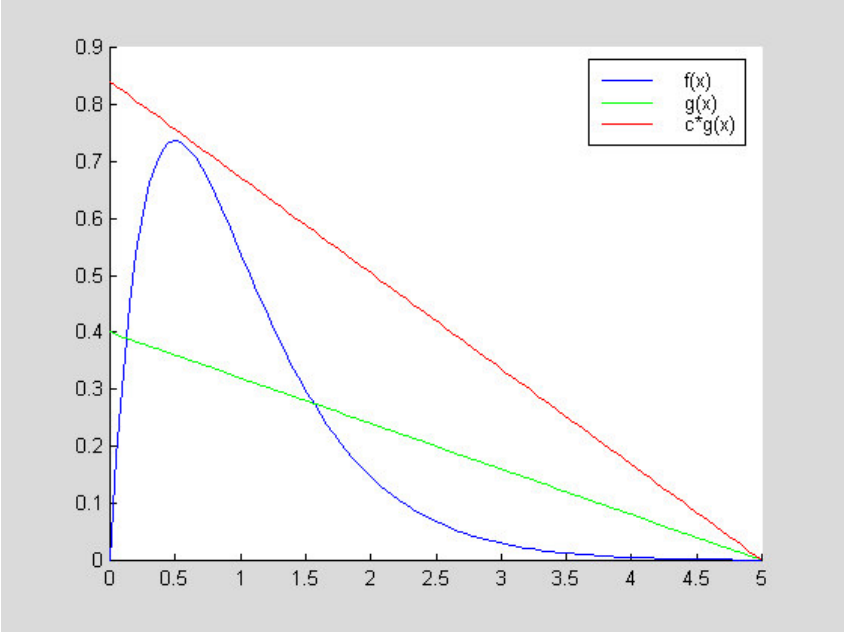

Figura 3.3: O método da aceitação-rejeição para X com FDP $f(x) = 4xe^{-2x}$.

A Tabela 3.4 mostra as freqüências obtidas e esperadas na simulação de 1000

extrações da variável X. Note-se que, para a simulação de $n = 1000$ extrações da variável X, espera-se ser necessário simular $m = cn = 2.1 \times 1000 = 2100$ extrações da variável Z. No exemplo, simulou-se a extração de $m = 2142$ valores de Z.

Intervalo	Freq. Obtida	Freq. Esperada
$[0;1)$	599	594
$[1;2)$	305	314
$[2;3)$	81	74
$[3;4)$	11	4
$[4;5]$	4	3

Tabela 3.4: Simulação da variável X com FDP f dada por $f(x) = 4xe^{-2x}$.

Se uma FDP $f : [a;b] \to \Re$ tem imagem $f([a;b]) = [0;h]$ para algum $h > 0$, é possível simplificar a FDP g tornando-a uniformemente distribuída no intervalo $[a;b]$, isto é, tomando $g(x) = \frac{1}{b-a}$. Essa simplificação necessita de um número de extrações de Z geralmente mais elevado, pois $c = H(b-a)$. Outra variação da técnica consiste em particionar o intervalo $[a;b]$ em n subintervalos menores e definir g por segmentos de reta nesses subintervalos. Isso pode reduzir significativamente o número de extrações de Z (Abramowitz & Stegun, 1970).

3.2.3 Outras técnicas.

As técnicas mostradas nas seções anteriores são de aplicação genérica. No entanto, devido às propriedades específicas de cada distribuição de probabilidade, pode-se simular variáveis aleatórias com algoritmos específicos.

Seja, por exemplo, a variável aleatória X de FDP f dada por $f(x) := 2x$, $x \in [0;1]$, como no exemplo da subseção 3.2.1. Pode-se simular a váriável X com

$$X := \max\{U_1, U_2\}, \tag{3.4}$$

onde U_1, U_2 são variáveis aleatórias uniformes independentes. Verifica-se isso a partir do seguinte argumento: Seja $\mathbf{S} := (U_1, U_2)$ uniformemente distribuída na região $Q := [0;1]\times[0;1]$. Seja $Q_x := [0;x]\times[0;x]$ uma sub-região retangular de Q. A probabilidade $P(\mathbf{S} \in Q_x) = x^2$, que é igual a $P(\max\{U_1, U_2\} \leq x) = P(X \leq x) = \int_0^x f(t)dt$. Assim, se $\int_0^x f(t)dt = x^2$, $f(x) = 2x$, quando $x \in [0;1]$.

3.3 Distribuições d-dimensionais.

Se $X_1, X_2, ..., X_d$ são variáveis aleatórias independentes com FDP $f_1, f_2, ..., f_d$, respectivamente, então define-se $\mathbf{X} := (X_1, X_2, ..., X_d)$ como uma variável aleatória d-dimensional com FDP conjunta $f := f_1 f_2 ... f_d$. Como a variável $\mathbf{X}$ tem componentes independentes, pode ser simulada através da simulação independente das variáveis $X_1, X_2, ..., X_d$ pelas técnicas descritas anteriormente. Nos casos em que isso não é possível, existem técnicas de simulação específicas.

3.3.1 Método da transformação.

No caso das variáveis d-dimensionais, vale um resultado interessante (Hoel *et alii*, 1971; Press *et alii*, 1992). Se $\mathbf{Y} := (Y_1, Y_2, ..., Y_d)$ é variável aleatória d-dimensional definida em termos de $\mathbf{X} := (X_1, X_2, ..., X_d)$ por um conjunto de transformações

$$Y_1 := h_1(X_1, X_2, ..., X_d),\ Y_2 := h_2(X_1, X_2, ..., X_d),\ ...,\ Y_d := h_d(X_1, X_2, ..., X_d),$$

então a FDP conjunta g de $\mathbf{Y}$ é dada por

$$g = \frac{1}{\left| \frac{\partial(Y_1, Y_2, ..., Y_d)}{\partial(X_1, X_2, ..., X_d)} \right|} f,$$

onde f é a FDP conjunta de $\mathbf{X}$, e

$$\frac{\partial(Y_1, Y_2, ..., Y_d)}{\partial(X_1, X_2, ..., X_d)} = \begin{vmatrix} \frac{\partial Y_1}{\partial X_1} & \frac{\partial Y_1}{\partial X_2} & ... & \frac{\partial Y_1}{\partial X_d} \\ \frac{\partial Y_2}{\partial X_1} & \frac{\partial Y_2}{\partial X_2} & ... & \frac{\partial Y_2}{\partial X_d} \\ ... & ... & ... & ... \\ \frac{\partial Y_d}{\partial X_1} & \frac{\partial Y_d}{\partial X_2} & ... & \frac{\partial Y_d}{\partial X_d} \end{vmatrix},$$

é o **determinante Jacobiano** de $\mathbf{Y}$ em relação a $\mathbf{X}$. Isso, é claro, quando há suficiente regularidade.

Assim sendo, é possível obter uma variável aleatória $\mathbf{Y}$ com FDP conjunta g a partir de outra variável aleatória $\mathbf{X}$ com FDP conjunta f e de um conveniente conjunto de transformações $h_1, h_2, ..., h_d$, tais que as relações acima sejam satisfeitas. Como exemplo desta técnica, descreve-se a obtenção de variáveis aleatórias com FDP conjunta Normal pelo método descrito na subseção 3.4.2.

3.3.2 Método da inversão.

O método da inversão unidimensional (subseção 3.2.1) pode ser entendido da seguinte maneira: para obter uma variável aleatória X com FDP f, a partir de uma variável aleatória uniforme U, deve-se determinar um função $U := h(X)$ tal que $\frac{dU}{dX} = f(x)$. Mas a solução dessa equação diferencial é trivial pois $h = F$, sendo F primitiva de f.

Assim, X é obtida pela inversão $X = h^{-1}(U) = F^{-1}(U)$.

No caso d-dimensional, deve-se determinar um conjunto de transformações $U_1 = h_1(X_1, X_2, ..., X_d),\ U_2 = h_2(X_1, X_2, ..., X_d),\ ...,\ U_d = h_d(X_1, X_2, ..., X_d)$ tais que

$$\left| \frac{\partial(U_1, U_2, ..., U_d)}{\partial(X_1, X_2, ..., X_d)} \right| = f.$$

E, em seguida, por inversão, obter

$$X_1 = h_1^{-1}(U_1, U_2, ..., U_d), X_2 = h_2^{-1}(U_1, U_2, ..., U_d), ..., X_d = h_d^{-1}(U_1, U_2, ..., U_d).$$

3.3.3 Método da aceitação-rejeição.

O método da aceitação-rejeição também pode ser estendido para o caso das distribuições d-dimensionais descritas acima. Nesse caso, a variável aleatória d-dimensional $\mathbf{X}$ com FDP conjunta f é obtida a partir da variável uniforme U e de uma variável aleatória d-dimensional $\mathbf{Z}$ com FDP conjunta g, independente de U, de modo que $cg(\mathbf{t}) \geq f(\mathbf{t})$ para todo $\mathbf{t}$ admissível e para alguma constante $c > 0$.

Algoritmo 3.5: Gerador com Distribuição d-dimensional pelo método de Aceitação-Rejeição (DdAR).

$[\mathbf{x}] \leftarrow DdAR(f, g, c)$
$\mathbf{z} \leftarrow g$ {$\mathbf{z}$ é obtido com FDP g por algum método}
enquanto $GU \bullet c \bullet g(\mathbf{z}) > f(\mathbf{x})$
    $\mathbf{z} \leftarrow g$ {nova extração, se rejeita a primeira}
fim
$\mathbf{x} \leftarrow \mathbf{z}$

3.3.4 Outras técnicas.

Outra técnica bastante utilizada é a denominada **técnica de Metropolis**, introduzida por Metropolis *et alii* (1953). Essa técnica consiste em gerar uma seqüência de pontos $(\mathbf{p}_i)$ através de um passeio aleatório d-dimensional. Cada novo termo $\mathbf{p}_i$ é aceito como extração da variável $\mathbf{X}$ se $f(\mathbf{p}_{i+1}) \geq uf(\mathbf{p}_i)$ onde u é extração da variável aleatória uniforme U. A FDP da variável $\mathbf{X}$ será então f. O algoritmo abaixo implementa a técnica de Metrópolis. Nesse algoritmo, f é uma FDP d-dimensional, $\mathbf{a}$ e $\mathbf{b}$ são vetores com os limites da região de amostragem, d é a dimensão e $\mathbf{x}$, o último valor da variável.

Algoritmo 3.6: Gerador com Distribuição d-dimensional pelo método de Metropolis (DMM).

[x]$\leftarrow$ $DMM(f, \mathbf{a}, \mathbf{b}, d, \mathbf{x})$
{inicialização}
$s \leftarrow (b-a)/4$ {passo}
{amostragem}
$Ok_1 \leftarrow falso$ {flag}
enquanto não Ok_1
    {escolhe novo ponto}
    $Ok_2 \leftarrow falso$ {flag}
    enquanto não Ok_2
        $Ok_2 \leftarrow verdadeiro$
        para $j \leftarrow 1 : d$
            $nx_j \leftarrow x_j + s_j(2GU - 1)$ {novo ponto}
            se $nx_j < a_j$ OU $nx_j > b_j$ {está dentro de **[a; b]**}
                $Ok_2 \leftarrow falso$
                break
            fim
        fim
    fim
    {aceitação-rejeição de Metropolis}
    se f(**nx**)/f(**x**) $\geq GU$ {aceita?}
        x $\leftarrow$ **nx**
        $Ok_1 \leftarrow verdadeiro$
    fim
fim

Existem muitas variações para o algoritmo básico de Metropolis. Gentle (1998) cita algumas modificações e generalizações. Esses algoritmos são usados com freqüência em uma ampla classe de algoritmos de simulação, denominados **algoritmos Monte Carlo com Cadeias de Markov** (*Markov Chain Monte Carlo*).

3.4 Distribuições especiais.

Para fixar idéias, expõem-se, nessa seção, algumas técnicas para simular variáveis aleatórias com FDP **Exponencial** e **Normal**, que ocorrem com grande freqüência. Algumas das técnicas aqui mostradas são particularizações das técnicas de inversão ou aceitação-rejeição, outras possuem justificativa própria, descrita sucintamente.

3.4.1 Distribuição Exponencial.

Técnica de Inversão. A variável aleatória X com FDP Exponencial

$$f(t) := \begin{cases} \frac{1}{\mu} e^{-\frac{t}{\mu}}, & t \geq 0, \\ 0, & t < 0, \end{cases} \tag{3.5}$$

de valor médio μ e variância μ^2, pode ser simulada a partir da variável uniforme U pela técnica de inversão:

$$u = \int_0^x f(t)dt = \begin{cases} 1 - e^{-\frac{x}{\mu}}, & x \geq 0 \\ 0, & x < 0. \end{cases}$$

Logo $x = -\mu \ln(1-u)$, se $u \in [0;1]$ e $x = 0$ se $u \notin [0;1]$.

Nesse caso $1-u$ pode ser substituído por u, pois as variáveis aleatórias U e $U' := 1 - U$ possuem a mesma FDP uniforme.

Algoritmo 3.7: Gerador com Distribuição Exponencial (DExp).
$X \leftarrow DExp(\mu)$
$X = -\mu \ln(GU)$ {GU: gerador uniforme}

Por sua extrema simplicidade, a técnica de inversão é, provavelmente, a mais utilizada, embora Abramovitz & Stegun (1970) e Knuth (1998) citem outras técnicas, que evitam o uso da função logarítmica. O Algoritmo 3.7 é, em essência, o utilizado pela rotina `exprnd` do *toolbox* `stats` do MATLAB.

3.4.2 Distribuição Normal.

Para simular a variável aleatória X de FDP Normal,

$$f(t) := \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(t-\mu)^2}{2\sigma^2}}, \tag{3.6}$$

com valor médio μ e desvio padrão σ, faz-se

$$X := \mu + \sigma Y,$$

onde Y também tem FDP Normal, mas valor médio $\mu = 0$ e desvio padrão $\sigma = 1$. Para obter a variável Y, recorre-se às seguintes técnicas.

Técnica da soma finita. A técnica da soma finita consiste em simular um número finito de extrações $(u_i) := (u_1, u_2, ..., u_n)$ da variável uniforme U e escrever

$$z_n := \frac{(u_1 + u_2 + ... + u_n) - \frac{n}{2}}{\sqrt{\frac{n}{12}}}.$$

Pelo Teorema do Limite Central, quanto maior o número de parcelas dessa soma, mais a FDP da variável Z aproxima-se da FDP Normal da variável Y. Os termos $\frac{n}{2}$ e $\sqrt{\frac{n}{12}}$ são fatores de correção para que o valor médio e o desvio padrão de Z sejam 0 e 1, respectivamente (Abramowitz & Stegun, 1970; Ruckdeschel, 1981). Essa técnica é, portanto, aproximada. A diferença entre a FDP de Z e a FDP Normal diminui à medida que n aumenta. A variável Z assume valores no intervalo simétrico $[-a;a]$; assim, a probabilidade $p_Z := P[(Z < -a) \cup (a < Z)]$ da variável Z assumir valores fora desse intervalo é zero, enquanto que a probabilidade $p_Y := P[(Y < -a) \cup (a < Y)]$ não é zero. Para $n = 12$, o intervalo de Z é $[-6;6]$ e $p_Y = 1.97 \times 10^{-9}$; para $n = 24$, o intervalo de Z é $[-6\sqrt{2}\,;6\sqrt{2}\,]$ e $p_Y = 2.15 \times 10^{-17}$, que são aproximações razoáveis. No entanto, esse método é bastante lento em comparação com os demais.

Técnica de Aceitação-Rejeição via distribuição Exponencial. A técnica de aceitação-rejeição via distribuição Exponencial simula a variável Y de FDP Normal (3.6) a partir de outra variável Z de FDP Exponencial (3.5). Nesse caso, o valor da constante de majoração é $c = e^{1/2}/\sqrt{2\pi}$. Inicialmente, simula-se uma extração z da variável aleatória Z através da técnica de inversão

$$z = -\ln(u_1),$$

onde u_1 é uma extração da variável uniforme. Em seguida aceita-se z se e somente se

$$u_2 c g(z) < f(z).$$

Nesse passo, a comparação pode ser simplificada:

$$u_2 < e^{-\frac{(z-1)^2}{2}}.$$

Agora, os valores de z aceitos têm valores apenas positivos (Abramovitz & Stegun, 1970). Então, fazendo

$$y := \begin{cases} z, & \text{se } u_3 \geq 0.5, \\ -z, & \text{se } u_3 < 0.5, \end{cases}$$

obtêm-se valores positivos e negativos.

Técnica da Transformação Polar. A técnica denominada transformação polar é um método simples e eficiente, proposto por Box *et alii* (1958). Duas extrações y_1 e y_2 (independentes) da variável Y são simuladas diretamente a partir de duas extrações u_1 e u_2 da variável uniforme U, fazendo

$$\begin{aligned} y_1 &:= \sqrt{-2\ln u_1}\,\sin(2\pi u_2), \\ y_2 &:= \sqrt{-2\ln u_1}\,\cos(2\pi u_2). \end{aligned}$$

De acordo com o exposto na subseção 3.3.1, y_1 e y_2 são obtidas por inversão de

$$\begin{aligned} u_1 &= e^{-\left(y_1^2+y_2^2\right)/2}, \\ u_2 &= \frac{1}{2\pi}\arctan\frac{y_2}{y_1} \end{aligned}$$

e satisfazem

$$\begin{vmatrix} \frac{\partial u_1}{\partial y_1} & \frac{\partial u_1}{\partial y_2} \\ \frac{\partial u_2}{\partial y_1} & \frac{\partial u_2}{\partial y_2} \end{vmatrix} = -\left[\frac{1}{\sqrt{2\pi}}e^{-y_1^2/2}\right]\left[\frac{1}{\sqrt{2\pi}}e^{-y_2^2/2}\right]$$

O algoritmo seguinte implementa essa técnica.

Algoritmo 3.8: Gerador com Distribuição Normal (DNormal).
$(x_1,x_2) \leftarrow DNormal(\mu,\sigma)$
$u_1 \leftarrow GU$
$u_2 \leftarrow GU$
$y_1 \leftarrow \sqrt{-2\ln u_1}\,\sin(2\pi u_2)$
$y_2 \leftarrow \sqrt{-2\ln u_1}\,\cos(2\pi u_2)$
$x_1 \leftarrow \mu + \sigma y_1$
$x_2 \leftarrow \mu + \sigma y_2$

Técnica de Inversão. A técnica de inversão simples, geralmente, não se mostra eficiente, pois requer a inversão de

$$u(x) := \int_{-\infty}^{x} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(t-\mu)^2}{2\sigma^2}}\,dt,$$

que é sempre trabalhosa, pois é necessário calcular numericamente a integral. A técnica usual é interpolar valores conhecidos. No MAPLE, por exemplo, o comando `stats[random, normald] (n, 'default', 'inverse')` realiza essa interpolação.

4 O método Monte Carlo na quadratura multidimensional

4.1 Quadratura unidimensional.

A quadratura numérica é a principal ferramenta usada para a obtenção de respostas aproximadas para integrais definidas que não podem ser resolvidas analiticamente. Denomina-se **quadratura numérica unidimensional** ao processo pelo qual a integral

$$I := \int_a^b f(x)dx$$

da função $f : [a;b] \rightarrow \Re$ é calculada, de forma aproximada, pela avaliação da função f em um número finito de pontos no intervalo $[a;b]$. A forma geral dessa aproximação é

$$I = Q + E,$$

onde Q é dito **quadratura** de f e denota o valor aproximado de I. O valor E denota o **erro de truncamento** da aproximação, a diferença entre I e Q.

4.1.1 Fórmulas de quadratura.

As **fórmulas de quadratura** são geralmente escritas como

$$Q = \sum_{i=0}^{m} w_i f(x_i), \tag{4.1}$$

onde os x_i com $a \leq x_0 < x_1 < x_2 < ... < x_m \leq b$ são denominados **nodos** da quadratura e os w_i são denominados **pesos** da quadratura.

Os nodos podem ser escolhidos de vários modos. Se são igualmente espaçados, a igualdade (4.1) é denominada fórmula de quadratura de **Newton-Cotes**. Se $x_0 = a$ e $x_m = b$, então a fórmula é chamada de quadratura de **Newton-Cotes fechada**.

No caso da quadratura de Newton-Cotes fechada, se os nodos forem escolhidos como sendo $x_i = a + ih,\ i = 0,...,m,$ com $h := (b-a)/m,$ então os pesos w_i da quadratura podem ser obtidos pelo seguinte sistema de equações (Dotto, 1998):

$$\begin{cases} w_0 + w_1 + w_2 + ... + w_m = mh \\ w_1 + 2w_2 + 3w_3 + ... + mw_m = \dfrac{m^2 h}{2} \\ w_1 + 2^2 w_2 + 3^2 w_3 + ... + m^2 w_m = \dfrac{m^3 h}{3} \\ ... \\ w_1 + 2^m w_2 + 3^m w_3 + ... + m^m w_m = \dfrac{m^{m+1} h}{m+1}. \end{cases} \tag{4.2}$$

Para $m = 1, 2, 3, 4$, as fórmulas de quadratura de Newton-Cotes denominadas de regra do **Trapézio**, **Simpson 1/3**, **Simpson 3/8** e **Boole** são, respectivamente,

$$\begin{aligned} Q_1 &= (b-a)\tfrac{1}{2}[f(x_0) + f(x_1)], \\ Q_2 &= (b-a)\tfrac{1}{6}[f(x_0) + 4f(x_1) + f(x_2)], \\ Q_3 &= (b-a)\tfrac{1}{8}[f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3)], \\ Q_4 &= (b-a)\tfrac{1}{90}[7f(x_0) + 32f(x_1) + 12f(x_2) + 32f(x_3) + 7f(x_4)]. \end{aligned} \tag{4.3}$$

Se os nodos são escolhidos como os zeros de determinados polinômios ortogonais (Legendre, Chebychev, Jacobi, etc.) a fórmula é dita quadratura de **Gauss** (Mathews, 1992). No caso da quadratura de Gauss-Legendre, além dos $m+1$ pesos w_i também os $m+1$ nodos x_i são incógnitas. Nesse tipo de quadratura abandona-se a idéia do espaçamento uniforme dos nodos, procurando otimizar suas localizações. Para a determinação das $2m+2$ incógnitas, pesos e nodos, requer-se que a quadratura seja exata para todos os polinômios de grau até $2m+1$ no intervalo I de ortogonalidade. Essa restrição de intervalo não implica perda de generalidade, pois o intervalo de integração $[a;b]$ pode ser aplicado de forma biunívoca sobre I. Obtém-se um sistema semelhante a (4.2),

$$\begin{cases} w_0 + w_1 + w_2 + ... + w_m = 2, \\ w_0 x_0 + w_1 x_1 + w_2 x_2 + ... + w_m x_m = 0, \\ ... \\ w_0 x_0^i + w_1 x_1^i + w_2 x_2^i + ... + w_m x_m^i = \begin{cases} 0 & \text{se } i \text{ é ímpar}, \\ \frac{2}{i+1} & \text{se } i \text{ é par}, \end{cases} \\ ... \\ w_0 x_0^{2m+1} + w_1 x_1^{2m+1} + w_2 x_2^{2m+1} + ... + w_m x_m^{2m+1} = 0, \end{cases}$$

que é não-linear, mas apresenta solução única e com todos os pesos positivos (Davis & Rabinowitz, 1975).

De modo geral, um conjunto de nodos e pesos pode ser determinado para cada conjunto de polinômios ortogonais. Seja (p_n) uma seqüência de polinômios ortogonais normalizados num intervalo $[a;b]$, determinada por sua função peso ω, com

$$p_n(x) = a_n x^n + a_{n-1}x^{n-1} + ... + a_1 x + a_0.$$

Se $x_1, x_2, ..., x_n$ são os zeros de p_n, então existem constantes positivas $w_1, w_2, .., w_n$ (pesos de quadratura) tais que

$$\int_a^b \omega(x)p(x)dx = w_1 p(x_1) + w_2 p(x_2) + ... + w_n p(x_n),$$

para todo o polinômio $p(x)$ de grau menor ou igual a $2n-1$. Os pesos w_i são dados por

$$w_i = -\frac{a_{n+1}}{a_n} \frac{1}{p_{n+1}(x_i)p_n'(x_i)}.$$

Se f é a função integranda, escrevendo $f(x) = \omega(x)g(x)$, então

$$\int_a^b f(x)dx = \int_a^b \omega(x)g(x)dx = Q + E,$$

com

$$Q = \sum_{i=0}^{m} w_i g(x_i). \tag{4.4}$$

4.1.2 O problema do valor médio e o método Monte Carlo.

O Teorema do Valor Médio para integrais afirma que, se f for uma função contínua no intervalo $[a;b]$, então existe pelo menos um ponto $c \in [a;b]$ tal que

$$\int_a^b f(x)dx = (b-a)f(c).$$

O valor $f(c)$ é dito **valor médio** da função f no intervalo $[a;b]$ (Ávila, 1993).

O ponto notável é que fórmulas de quadratura como (4.1), (4.3) ou (4.4) podem ser entendidas como métodos para o cálculo aproximado do valor médio da função integranda. Nos casos da quadratura de ordem m (isto é, com polinômio interpolador

de ordem m) de Newton-Cotes e Gauss, as fórmulas de quadratura fornecem o valor médio exato de todos os polinômios de grau até m e $2m-1$, respectivamente.

A idéia central do método Monte Carlo na quadratura unidimensional é fazer uma estimativa do valor médio de f através de uma amostragem aleatória de seus valores no intervalo $[a;b]$. Em sua forma simplificada, o método MC escolhe n pontos x_i no intervalo $[a;b]$ de forma eqüiprovável, isto é, com FDP uniforme. Dessa forma, os pesos w_i são tomados todos como $w_i = \frac{1}{n}$. A fórmula de quadratura do método MC pode ser expressa por

$$Q = (b-a)\langle f \rangle, \tag{4.5}$$

onde $\langle f \rangle$ denota a **média aritmética** dos valores de f amostrados,

$$\langle f \rangle := \frac{1}{n}\sum_{i=1}^{n} f(x_i). \tag{4.6}$$

Sendo o método MC um método estocástico, não é possível determinar o erro E cometido na aproximação da integral. Entretanto, uma estimativa e desse erro pode ser feita por

$$E \approx e := \pm(b-a)\sqrt{\frac{\langle f^2 \rangle - \langle f \rangle^2}{n}}\,, \tag{4.7}$$

onde $\langle f^2 \rangle$ denota o **valor quadrático médio** dos valores de f amostrados,

$$\langle f^2 \rangle := \frac{1}{n}\sum_{i=1}^{n} [f(x_i)]^2.$$

Do ponto de vista estatístico, essa estimativa de erro e é o **desvio padrão do valor médio** de Q, que pode ser tomado como uma estimativa razoável de E, se o número n de pontos amostrais for grande, tipicamente $n > 30$, de modo a ter-se Q com distribuição Normal. Se a distribuição de Q é Normal, o valor exato da integral está contido nos intervalos $[Q-e;Q+e]$, $[Q-2e;Q+2e]$ e $[Q-3e;Q+3e]$ com confiança estatística de 68%, 95% e 99%, respectivamente (Vuolo, 1996). Em decorrência da desigualdade de Chebyshev, para qualquer outra distribuição de Q, o valor da integral está contido nos intervalos $[Q-2e;Q+2e]$ e $[Q-3e;Q+3e]$ com confiança estatística de 75% e 89%, respectivamente (Graham *et alii*, 1994).

4.1.3 Um exemplo de quadratura unidimensional.

A título de ilustração do método MC, resolve-se um problema simples de quadratura unidimensional: a determinação do valor da integral elíptica completa de segunda classe

$$f(\varphi) = \int_0^{\frac{\pi}{2}} \sqrt{1 - \operatorname{sen}^2\varphi \operatorname{sen}^2\theta}\, d\theta.$$

Essa integral é um bom teste para quadraturas numéricas, pois existe mais de um método para sua avaliação numérica (método da média aritmética-geométrica, por exemplo), possibilitando a conferência de resultados (Corbit, 1996). Além disso, devido a sua utilização em uma ampla variedade de problemas, existem muitos programas matemáticos que fornecem valores para comparação. Por exemplo, para $\varphi = \pi/4$, os pacotes matemáticos MATLAB e MAPLE fornecem, respectivamente, os valores $f(\pi/4) = 1.35064388104768$ e $f(\pi/4) = 1.3506438810476755025$. Abramowitz & Stegun (1970) fornecem $f(\pi/4) = 1.350643881047676$.

Usando-se o método MC exposto acima, pode-se estimar o valor de $f(\pi/4)$ usando uma amostragem aleatória de n valores no intervalo $[0, \pi/2]$. A Tabela 4.1 mostra um resumo dos valores obtidos para o valor aproximado Q de acordo com (4.5), estimativas do erro e de acordo com (4.7) e os erros exatos $E := f(\pi/4) - Q$, usando-se diferentes valores para n.

n	Q	e (estimado)	E (exato)
5	1.27129939406883	0.06906655233864	+0.07934448697885
50	1.35632451479221	0.02025306452529	−0.00568063374453
500	1.35244080488016	0.00719312217385	−0.00179692383248
5000	1.34668017267048	0.00230408119149	+0.00396370837720

Tabela 4.1: Estimativa do valor de $f(\pi/4)$ pelo método Monte Carlo.

Observando a Tabela 4.1, verifica-se que o método MC de quadratura unidimensional não parece ser muito competitivo. Outras técnicas, como Newton-Cotes ou Gauss, são, de fato, muito mais eficientes. A aplicação da regra de Newton-Cotes para $m = 4$ (Regra de Boole) com apenas 5 nodos (pontos amostrais) já fornece o valor $Q[f] = 1.35046410473225$ com um erro de apenas $e[f] = 0.000179776315426$. No método MC, a convergência é lenta, pois, conforme (4.7), a convergência da integral é de ordem $O(n^{-1/2})$.

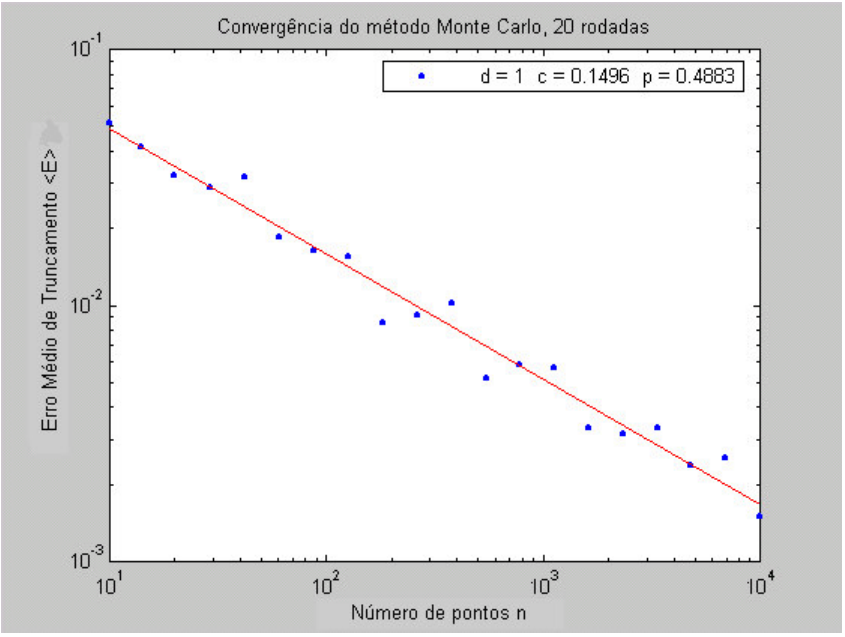

Figura 4.1: A convergência $O[n^{-1/2}]$ do método Monte Carlo.

A Figura 4.1 mostra em um gráfico na escala log-log o desempenho do método Monte Carlo. Cada ponto $(n,\langle E\rangle_n)$, onde n é o número de pontos amostrais e $\langle E\rangle_n$ é o erro médio de truncamento correspondente na avaliação de $f(\pi/4)$. Como o valor de E é diferente para cada amostra diferente e assume valores positivos e negativos, o valor médio $\langle E\rangle$ é tomado como a raiz média quadrática do erro E para amostras distintas (rodadas) de mesmo tamanho n, $\langle E\rangle = \sqrt{\frac{1}{m}\sum_{i=1}^{m} E_i^2}$, como sugerem Morokoff & Caflisch (1995). A reta (em escala log-log) é o ajustamento pelos mínimos quadrados da curva $E = cn^{-p}$, baseado no conjunto de dados $\{(n,\langle E\rangle_n)\}$, obtidos pelo método MC. O valor de $p \approx 0.5$ mostra a convergência $O(n^{-1/2})$ estimada por (4.5).

Observação. *As Figuras 4.2, 4.3, 4.4, 4.5 e 4.7 mostradas abaixo são semelhantes à Figura 4.1. As descrições da escala log-log, do erro médio de truncamento e da reta de ajuste são as mesmas. Evita-se, portanto, repeti-las.*

4.1.4 A justificativa do método Monte Carlo.

Considerando o exposto na subseção anterior, uma pergunta fundamental impõe-se: *se o método MC tem convergência tão lenta em comparação com outros métodos, qual sua justificativa?* A resposta a essa indagação pode ser entendida a partir das seguintes considerações.

O cálculo de integrais unidimensionais é uma tarefa cujos procedimentos numéricos são relativamente simples e estão exaustivamente descritos em vasta literatura. Existe

uma miríade de técnicas, para as quais a convergência é muito melhor que a do método MC descrito. Apenas para citar algumas: o método trapezoidal composto é $O(n^{-2})$; o método de Simpson composto é $O(n^{-4})$; o método de Romberg-Richardson com k refinamentos é $O(n^{-(2k+2)})$; o método de Gauss-Legendre é $O(2^{2n+1}(n!)^4/(2n+1)[(2n)!]^3)$ (Abramowitz & Stegun, 1970). Definitivamente, o método MC não é útil para quadraturas unidimensionais. No entanto, o cálculo de integrais d-dimensionais é um problema bem mais complexo, onde o método pode ser útil.

Em alguns casos, a dimensionalidade da quadratura pode ser diminuída. Isso é possível quando a região de integração e a função integranda possuem alguma simetria em comum, por exemplo, simetria esférica, cilíndrica, hiperbólica, etc. Essa redução de dimensionalidade também é possível no caso das chamadas integrais iteradas (Abramowitz & Stegun, 1970, eq. 25.4.58-59). Mas essa redução de dimensão é solução apenas para algumas integrais.

Uma das técnicas determinísticas mais importantes para quadratura d-dimensional produz a denomimada **Fórmula de Quadratura com Produto Cartesiano,** que consiste em realizar a quadratura em uma dimensão por vez, utilizando as regras de quadratura unidimensional. Por exemplo, para integrar $f : [a;b]\times[c;d]\rightarrow \Re$ com alguma fórmula de m pontos, determinam-se os nodos $x_1, x_2, ..., x_m$ em $[a;b]$ e os nodos $y_1, y_2, ..., y_m$ em $[c;d]$, obtendo

$$\int_{[a;b]\times[c;d]} f(x,y)dxdy \approx Q = \sum_{i=1}^{m} w_i f_i(x_i),$$

onde

$$f_i(x_i) = \sum_{j=1}^{m} w_j f(x_i, y_j).$$

Para $d > 2$ o procedimento é semelhante (Davis & Rabinowitz, 1975).

As dificuldades de técnicas determinísticas como essa são as seguintes:

1. o número n de pontos de avaliação de f cresce exponencialmente com a dimensão ($n = m^d$), o que implica estimativas de erro da ordem $O(n^{1/d})$ (Gentle, 1998);
2. para dimensão $d = 1$, a fronteira de integração é definida apenas por dois números, os extremos inferior e superior do intervalo; já a região de integração no espaço d-dimensional é definida por fronteiras $(d-1)$-dimensionais que por si próprias podem ser terrivelmente complicadas, podem ser não-convexas ou não-simplesmente conexas;
3. fórmulas de quadratura podem ser desenvolvidas para dimensões $d = 2$ ou $d = 3$ e para regiões de integração específicas, como cubo, esfera, simplex, etc. (Abramowitz & Stegun, 1970); no entanto, técnicas gerais para tratamento de

regiões de integração mais genéricas são de difícil obtenção;

4. o desenvolvimento de fórmulas de quadratura de Gauss estão relacionadas com famílias de polinômios ortogonais; a teoria desses polinômios parece ser ainda pouco desenvolvida para dimensão $d > 1$ (Thisted, 1996);
5. por fim, essas regras em geral não permitem adicionar novos pontos de amostragem sem perder as informações anteriores, uma vez que o peso e a posição dos nodos são distintas para cada m distinto, se bem que já existem alguns resultados nesse sentido (Gori & Michelli, 1996).

Diante dessas dificuldades, o método MC apresenta-se como um método alternativo, especialmente se um erro relativamente grande é tolerável, se a dimensão é grande ($d > 10$) ou quando a região de integração é complicada. Os melhores resultados são obtidos quando a função integranda é suave, com derivadas de valor baixo (Press *et alii,* 1992).

O método MC descrito acima adapta-se facilmente ao caso multidimensional e apresenta algumas vantagens em relação às adaptações descritas acima. Entre elas, pode-se citar as seguintes:

1. a sua **simplicidade,** que se reflete em uma implementação computacional também simples, como o algoritmo 4.1;
2. a **estabilidade** de sua convergência; embora lenta, sua convergência de ordem $O(n^{-1/2})$ mantém-se estável, independentemente da dimensionalidade da quadratura (Gentle, 1998);
3. a **confiabilidade** da estimativa de erro; embora relativamente grande, a estimativa de erro é bastante confiável, devido a sua natureza estatística.

4.2 O método Monte Carlo multidimensional.

O Teorema do Valor Médio para integrais pode ser estendido para o caso d-dimensional: se $f : V \to \Re$ é contínua e $V \subset \Re^d$ é um compacto e conexo, então o valor da integral

$$I := \int_V f(\mathbf{x}) dV = vf(\mathbf{c}),$$

onde $f(\mathbf{c})$ é o valor médio da função, $\mathbf{c}$ algum ponto em V e v o volume da região V.

A quadratura pelo método Monte Carlo consiste em determinar uma estimativa da integral I através de uma estimativa para o valor médio de f. A estimativa para o valor médio de f é calculada por

$$\langle f \rangle := \frac{1}{n} \sum_{i=1}^{n} f(\mathbf{x}_i),$$

média aritmética dos valores de f em pontos $\mathbf{x}$ escolhidos de forma aleatória e uniforme no interior de V. Assim, a estimativa para a integral é

$$Q := v\langle f \rangle. \tag{4.8}$$

Como no caso unidimensional, o erro de truncamentro pode ser estimado por

$$e := v\sqrt{\frac{\langle f^2 \rangle - \langle f \rangle^2}{n}}, \tag{4.9}$$

onde $\langle f^2 \rangle$ é dado por

$$\langle f^2 \rangle = \frac{1}{n} \sum_{i=1}^{n} [f(\mathbf{x}_i)]^2.$$

Se a região de integração V é de difícil amostragem aleatória (como regiões com contornos complicados) ou de volume v desconhecido, pode-se encontrar uma região de amostragem W que contenha V, que seja mais facilmente amostrada e que tenha volume conhecido, como regiões d-retangulares, por exemplo. Então define-se $g : W \to \Re$ por

$$g(\mathbf{x}) = \begin{cases} f(\mathbf{x}), & \text{se } \mathbf{x} \in V, \\ 0, & \text{se } \mathbf{x} \in W - V, \end{cases} \tag{4.10}$$

e aplica-se o método MC para aproximar $\int_W g$ em lugar de $\int_V f$.

Deve-se tentar fazer a diferença $W - V$ a menor possível, já que os valores nulos de g fazem crescer o erro estimado em (4.9). Pontos escolhidos fora de V não contêm informação. Então, é reduzido o valor efetivo do número n de pontos escolhidos. A estimativa de erro em (4.7) leva isso em consideração.

Se a função integranda f é constante e a região de integração V igual à região de amostragem W, então o valor da estimativa de erro em (4.7) torna-se zero. De fato, isso significa que a integral é calculada exatamente. Essa situação ideal é rara, mas se for possível, através de uma mudaça de variável, diminuir o quanto possível o intervalo de variação da função integranda e tomar a região de amostragem W a mais próxima possível da região de integração V, a precisão do método MC vai aumentar

(Press *et alii*, 1992).

Observação. *Nos algoritmos descritos neste e o no próximo capítulos, admite-se que a região de integração é d-retangular,* $V := [a_1;b_1]\times[a_2;b_2]\times...\times[a_d;b_d]$ *definida pelos vértices* $\mathbf{a} = (a_1, a_2, ..., a_d)$ *e* $\mathbf{b} = (b_1, b_2, ..., b_d)$ *ou que é uma região de amostragem que a contém. Do mesmo modo, admite-se que f é a função integranda ou é a função g definida por (4.10).*

O Algoritmo 4.1, descrito abaixo, implementa o método MC d-dimensional com n pontos de amostragem, e determina estimativas para o valor da quadratura Q e do erro e.

Algoritmo 4.1: Quadratura d-dimensional pelo método Monte Carlo(QMC).
$[Q, e] \leftarrow QMC(f, a, b, d, n)$,
{inicialização}
$s \leftarrow 0$ {acumulador para $< f >$}
$t \leftarrow 0$ {acumulador para $< f^2 >$}
$L_i \leftarrow b_i - a_i$ {arestas da região de integração}
$w \leftarrow L_1 L_2 ... L_d$ {volume da região de integração}
{amostragem}
para $i \leftarrow 1 : n$,
    para $j \leftarrow 1 : d$,
        $x_j \leftarrow a_j + L_j \times GU$ {GU: gerador uniforme}
    fim
    $s \leftarrow s + f(\mathbf{x})$
    $t \leftarrow t + f^2(\mathbf{x})$
fim
{estimativas}
$mf \leftarrow s/n$ {$< f >$}
$mf2 \leftarrow t/n$ {$< f^2 >$}
$Q \leftarrow v \times mf$;
$e \leftarrow v \times \sqrt{\frac{mf2 - mf^2}{n}}$

As Figuras 4.2 e 4.3 mostram em gráficos na escala log-log o desempenho do método Monte Carlo para $d > 1$. Na Figura 4.2, as marcas (•), (+), (×) e (□) representam os pontos $(n, \langle E \rangle_n)$ na avaliação da integral d-dimensional $I_d = \left(\frac{3}{2}\right)^d \int_{V_d} \sqrt{x_1 x_2 ... x_d}\, dx_1 dx_2 ... dx_d = 1$ com $V_d = [0,1]\times[0,1]\times...\times[0,1]$ para $d = 5$, $d = 10$, $d = 15$ e $d = 20$, respectivamente.

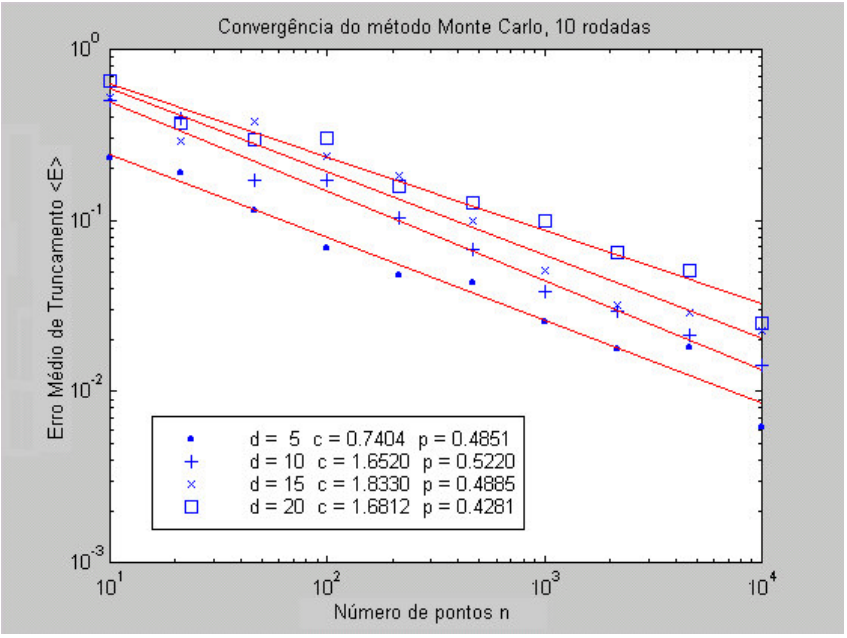

Figura 4.2: Convergência do método MC para $d > 1$.

Note-se que, enquanto a dimensionalidade da quadratura aumenta, o valor de p, da curva de ajuste $E = cn^{-p}$, mantém-se estável, próximo a 0.5. Esse comportamento mantém-se para dimensões mais elevadas.

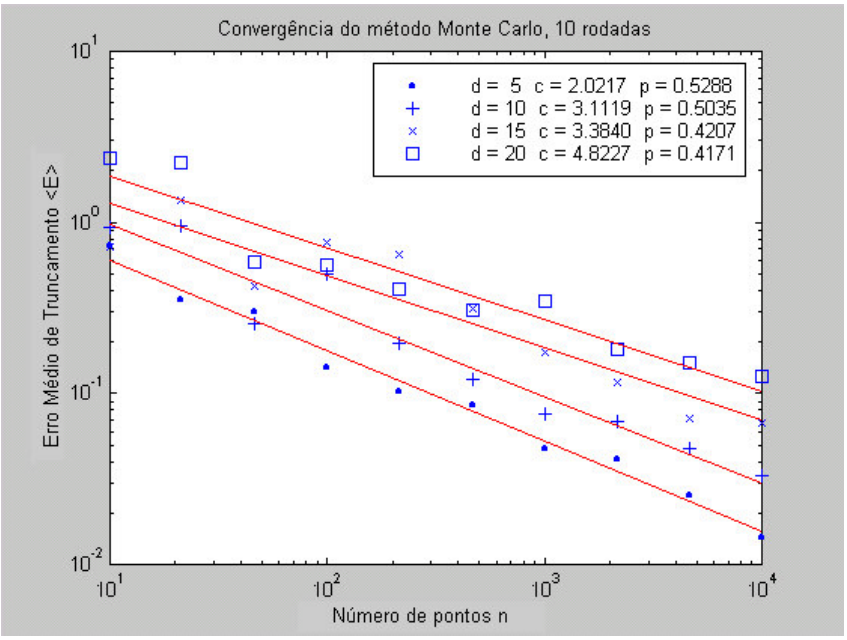

Figura 4.3: Convergência do método MC para $d > 1$.

Na Figura 4.3, as marcas (•), (+), (×) e (□) representam os pontos $(n, \langle E \rangle_n)$ na avaliação da integral d-dimensional $I_d = (e - \frac{1}{e})^{-d} \int_{V_d} e^{x_1} e^{x_2} ... e^{x_d} dx_1 dx_2 ... dx_d = 1$ com

$V_d = [-1,1]\times[-1,1]\times...\times[-1,1]$ para $d = 5$, $d = 10$, $d = 15$ e $d = 20$, respectivamente. Note-se que, como a função integranda tem derivada muito grande em torno de $\mathbf{x} = (1,1,...,1)$, à medida que a dimensionalidade aumenta, o valor de p tende a afastar-se de 0.5. Esse comportamento acentua-se para dimensões mais elevadas.

Uma segunda versão do Algoritmo 4.1 pode ser feita em função da confiabilidade da estimativa do erro e. Pode-se modificar o critério de parada do algoritmo: parar de amostrar quando o erro estimado e for menor que um certo valor da tolerância e_{max} fornecido.

Algoritmo 4.2: Quadratura d-dimensional pelo método Monte Carlo, versão 2 (QMC2).

$[Q,e,n] \leftarrow QMC2(f,\mathbf{a},\mathbf{b},d,e_{max})$,
{inicialização}
$s \leftarrow 0$ {acumulador para $< f >$}
$t \leftarrow 0$ {acumulador para $< f^2 >$}
$L_i \leftarrow b_i - a_i$ {arestas da região de integração}
$v \leftarrow L_1 L_2 ... L_d$ {volume da região de integração}
$e \leftarrow \infty$ {estimativa de erro}
$n \leftarrow 0$ {número de pontos de amostragem}
{amostragem Monte Carlo}
enquanto $e > e_{max} \vee n < 5$,
    {ponto de amostragem}
    para $i \leftarrow 1 : d$
        $x_i \leftarrow a_i + L_i \times GU$
    fim
    $n \leftarrow n + 1$
    {avaliação da função}
    $s \leftarrow s + f(\mathbf{x})$
    $t \leftarrow t + f^2(\mathbf{x})$
    {estimativas}
    $mf \leftarrow s/n$ $\{< f >\}$
    $mf2 \leftarrow t/n$ $\{< f^2 >\}$
    $Q \leftarrow v \times mf$;
    $e \leftarrow v \times \sqrt{\frac{mf2 - mf^2}{n}}$
fim

4.3 Redução de variância.

A desvantagem fundamental do método MC é sua fraca convergência, de ordem $O(n^{-p})$, com $p = 1/2$. Como esse método é intrinsecamente de amostragem estatística e o erro da quadratura é associado ao desvio padrão (ou, equivalentemente, à

variância), dado por (4.9), o valor de p não pode ser aumentado. No entanto, existem técnicas, denominadas coletivamente de **redução de variância**, que permitem diminuir levemente o valor da estimativa de erro.

4.3.1 Técnica da amostragem de importância.

Na técnica denominada **amostragem de importância** (*importance sampling*), os pontos $\mathbf{x}$ de amostragem não são escolhidos de modo uniforme, mas distribuídos de acordo com alguma FDP p escolhida adequadamente.

Se a função integranda f puder ser escrita como $f(\mathbf{x}) = h(\mathbf{x})p(\mathbf{x})$ com h contínua no compacto V e p com as propriedades de uma função densidade de probabilidade, isto é,

$$\begin{cases} p(\mathbf{x}) \geq 0 & \text{se } x \in V, \\ p(\mathbf{x}) = 0 & \text{se } x \notin V, \\ \int_V p(\mathbf{x})dV = 1, \end{cases}$$

então pelo Teorema do Valor Médio Generalizado para integrais,

$$I = \int_V f(\mathbf{x})dV = \int_V h(\mathbf{x})p(\mathbf{x})dV = h(\mathbf{c})\int_V p(\mathbf{x})dV = h(\mathbf{c}).$$

A técnica de amostragem de importância consiste em determinar uma estimativa Q para a integral I através de uma estimativa $\langle h \rangle$ para o valor médio $h(\mathbf{c})$. A estimativa para o valor médio de h é dada por

$$Q := \langle h \rangle := \frac{1}{n}\sum_{i=1}^{n} h(\mathbf{x}_i),$$

média aritmética dos valores de h em pontos $\mathbf{x}$ escolhidos de forma aleatória e com FDP p no interior de V.

Como em (4.7), a estimativa para o erro é dada por

$$e := \sqrt{\frac{\langle h^2 \rangle - \langle h \rangle^2}{n}}\,.$$

A idéia do método é escolher uma FDP p tal que (i) pontos com essa FDP sejam de

fácil obtenção, como, por exemplo, alguma técnica vista no Capítulo 3; e (ii) a variância de $h := f/p$ seja menor que a variância de f, o que é conseguido se $|p - f|$ for o menor possível. De modo ótimo, a variância será zero se h for constante. Note-se que essa diminuição da variância é feita às custas da duplicação do número de avaliações funcionais (Davis & Rabinowitz, 1975).

Algoritmo 4.3: Quadratura Monte Carlo com amostragem de importância(QMCI).
$[Q, e] \leftarrow QMCI(f, p, \mathbf{a}, \mathbf{b}, d, n)$
{inicialização}
$s_1 \leftarrow 0$ {acumulador para $< h >$}
$s_1 \leftarrow 0$ {acumulador para $< h^2 >$}
{amostragem}
para $i \leftarrow 1 : n$
    $\mathbf{x} \leftarrow ...$ {$\mathbf{x}$ é calculado com distribuição multivariada p}
    $h \leftarrow f(\mathbf{x})/p(\mathbf{x})$
    $s_1 \leftarrow s_1 + h$
    $s_2 \leftarrow s_2 + h^2$
fim
{estimativas}
$m_h \leftarrow s_1/n$ {$< h >$}
$m_{h^2} \leftarrow s_2/n$ {$< h^2 >$}
$Q \leftarrow m_h$
$e \leftarrow \sqrt{(m_{h^2} - m_h^2)/n}$

Na Figura 4.4 as marcas (•) e (+) representam os pontos $(n, \langle E \rangle_n)$ na avaliação da integral $I_5 = (e - \frac{1}{e})^{-5} \int_{V_5} e^{x_1} e^{x_2} ... e^{x_5} dx_1 dx_2 ... dx_5 = 1$ com $V_5 = [-1, 1]^5$. Os pontos (•) representam valores obtidos pelo método MC e as cruzes (+) representam os valores obtidos pelo médodo MC com amostragem de importância (MCAI) utilizando a FDP

$$p_d(\mathbf{x}) := \frac{\frac{1}{2^d}(e^{2d} - 1)\prod_{i=1}^{d}(x_i + 1) + 1}{(e^{2d} - 1) + 2^d},$$

escolhida de modo que $|p - f|$ seja pequeno. A reta é o gráfico da curva-erro $E = cn^{-p}$ com ajustamento pelos mínimos quadrados ao conjunto de dados $\{E, n\}$ obtidos pelo método MC. Note-se que o erro cometido pelo método Monte Carlo com amostragem de importância é levemente inferior ao erro cometido pelo método MC (com exceção de um caso, nesse exemplo).

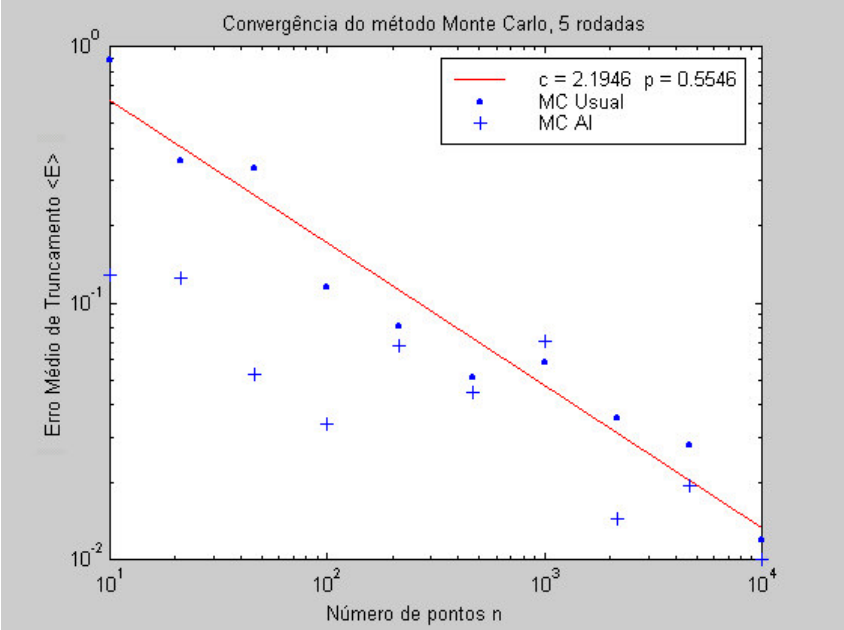

Figura 4.4: Convergência do método MC com amostragem de importância.

4.3.2 Técnica da amostragem antitética

Na técnica MC os n pontos de amostragem são independentes. Uma técnica simples de redução de variância consiste em amostrar n pontos não-independentes. Esses pontos $\mathbf{x}$ e $\mathbf{x}'$ são valores correspondentes a variáveis aleatórias $\mathbf{X}$ e $\mathbf{X}'$ relacionadas por

$$\mathbf{X}' := \mathbf{a} + \mathbf{b} - \mathbf{X}, \tag{4.11}$$

onde $\mathbf{a} := (a_1, a_2, ..., a_d)$ e $\mathbf{b} := (b_1, b_2, ..., b_d)$ são os vértices da região retangular de integração. As variáveis $\mathbf{X}$ e $\mathbf{X}'$ são simétricas (antitéticas) em relação ao ponto central da região de integração e são denominadas **variáveis antitéticas** (Gentle, 1998). A estimativa para o valor médio de f passa a ser dada por

$$\langle f \rangle := \frac{1}{n} \sum_{i=1}^{n/2} [f(\mathbf{x}_i) + f(\mathbf{x}_i')].$$

Se $\phi := f(\mathbf{x}_i)$, $\phi' := f(\mathbf{x}_i')$, a variância de $\langle f \rangle$ é proporcional à

$$\begin{aligned} Var[\phi + \phi'] &= Esp[\phi + \phi' - Esp[\phi + \phi']]^2 \\ &= Esp[\phi - Esp[\phi]]^2 + Esp[\phi' - Esp[\phi']]^2 + 2Esp[(\phi - Esp[\phi])(\phi' - Esp[\phi'])] \\ &= Var[\phi] + Var[\phi'] + 2Cov[\phi, \phi']. \end{aligned}$$

Se ϕ e ϕ' são independentes, $Cov[\phi,\phi']= 0$ e $Var[\phi+\phi']= Var[\phi]+Var[\phi']$. Mas, sob certas condições (f tem forma aproximada de um plano d-dimensional, por exemplo), ϕ e ϕ' são variáveis aproximadamente antitéticas: $\phi' \approx c-\phi$, onde c é alguma constante. Então a covariância $Cov[\phi,\phi']$ será negativa,

$$\begin{aligned} Cov[\phi,\phi'] &= Esp[\phi\phi']-Esp[\phi]Esp[\phi'] \\ &\approx Esp[\phi(c-\phi)]-Esp[\phi]Esp[c-\phi] \\ &= cEsp[\phi]-Esp[\phi^2]-cEsp[\phi]+Esp[\phi]Esp[\phi] \\ &= -Esp[\phi^2]+Esp[\phi]Esp[\phi] \\ &= -Var[\phi]\leq 0, \end{aligned}$$

o que reduz a variância para $Var[\phi+\phi']\approx Var[\phi]-Var[\phi']$.

Algoritmo 4.4: Quadratura Monte Carlo com amostragem antitética (QMCA).

$[Q,e]\leftarrow QMCA(f,\mathbf{a},\mathbf{b},d,n)$,
{inicialização}
$s \leftarrow 0$ {acumulador para $< f >$}
$t \leftarrow 0$ {acumulador para $< f^2 >$}
$L_i \leftarrow b_i - a_i$ {arestas da região de integração}
$w \leftarrow L_1 L_2 ... L_d$ {volume da região de integração}
{amostragem}
para $i \leftarrow 1 : n$,
    para $j \leftarrow 1 : d$,
        $x_j \leftarrow a_j + L_j \times GU$ {primeiro ponto}
    fim
    $s \leftarrow s + f(\mathbf{x})$
    $t \leftarrow t + f^2(\mathbf{x})$
    $\mathbf{x} \leftarrow \mathbf{a} + \mathbf{b} - \mathbf{x}$ {segundo ponto (antitético)}
    $s \leftarrow s + f(\mathbf{x})$
    $t \leftarrow t + f^2(\mathbf{x})$
fim
{estimativas}
$mf \leftarrow s/n$ {$< f >$}
$mf2 \leftarrow t/n$ {$< f^2 >$}
$Q \leftarrow v \times mf$;
$e \leftarrow v \times \sqrt{\frac{mf2-mf^2}{n}}$

Na Figura 4.5, as marcas (•) e (+) representam os pontos $(n,\langle E\rangle_n)$ na avaliação da integral $I_5 = \left(\frac{3}{2}\right)^5 \int_{V_5} \sqrt{x_1 x_2 ... x_5}\, dx_1 dx_2 ... dx_5 = 1$ com $V_5 = [0,1]^5$. Os pontos (•)

representam valores obtidos pelo método MC e as cruzes (+) representam os valores obtidos pelo método MC com amostragem antitética. Note-se que o erro cometido pelo método MC com amostragem antitética é levemente inferior ao erro cometido pelo método MC. A dimensão da integral utilizada é $d = 5$. Para outras dimensões o comportamento é semelhante.

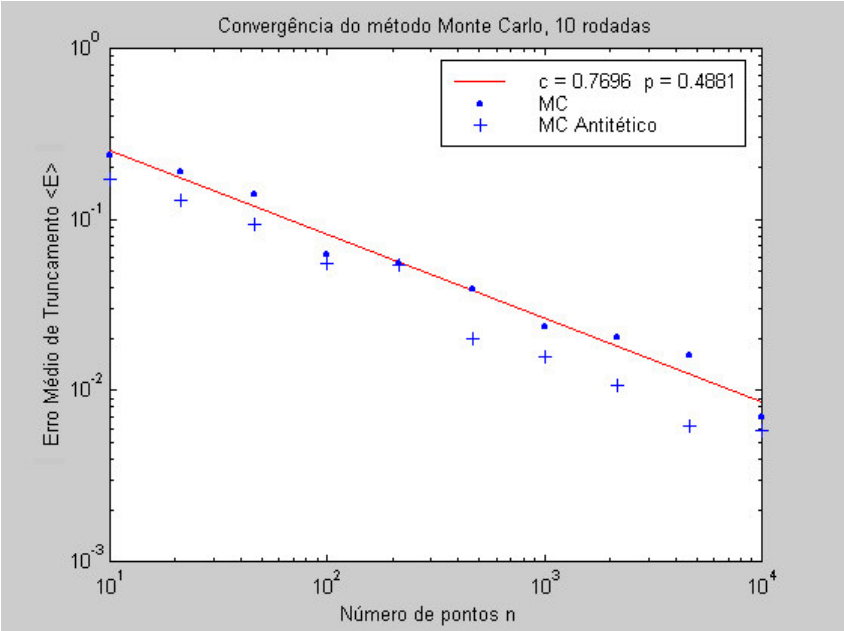

Figura 4.5: Convergência do método MC com amostragem antitética.

Variáveis aleatórias antitéticas com outras distribuições de probabilidade também podem ser geradas. Com essas outras distribuições, a relação (4.11) não pode, no entanto, ser usada. Se o método da inversão (subseção 3.2.1) é usado para gerar a variável $X := F^{-1}(U)$, então extrações $\mathbf{x}$ e $\mathbf{x}'$, antitéticas, podem ser geradas com $\mathbf{x} = F^{-1}(\mathbf{u})$ e $\mathbf{x}' := F - 1(\mathbf{u}')$ onde $\mathbf{u}' = \mathbf{a} + \mathbf{b} - \mathbf{u}$ (Gentle,1998).

4.3.3 Técnica da amostragem estratificada.

Na técnica denominada **amostragem estratificada**, a regra é amostrar mais densamente a função onde sua variância é maior. Isso é feito dividindo-se a região de integração em sub-regiões (estratos) onde a quantidade de pontos amostrais é proporcional à variação da função. Esse princípio é o mesmo utilizado em teoria de amostragem em que a alocação é proporcional à variância da amostra.

Se a região de integração é subdividida em n_p sub-regiões, e em cada uma dessas avaliam-se Q_i e e_i, então

$$Q = \sum_{i=1}^{n_p} Q_i \quad \text{e} \quad e = \sqrt{\sum_{i=1}^{n_p} e_i^2}\,.$$

Assim, a amostragem em cada sub-região pode ser feita até que o critério $e_1 \approx e_2 \approx ... \approx e_{n_p} < e\sqrt{n_p}$ seja satisfeito.

Algoritmo 4.5: Quadratura Monte Carlo com amostragem Estratificada(QMCE).
$[Q, e, n] \leftarrow QMCE(fun, \mathbf{a}, \mathbf{b}, d, e_{\max}, \mathbb{P})$
{inicialização}
$N_P \leftarrow P_1 P_2 ... P_d$ {número de sub-regiões}
para $i \leftarrow 1 : d$
    $d_i \leftarrow (b_i - a_i)/P_i$ {vetor diagonal}
fim
$Q \leftarrow 0,\ e \leftarrow 0,\ n \leftarrow 0$
{amostragem estratificada}
para $i \leftarrow 0 : N_P - 1$ {para cada sub-região...}
    {combinação}
    para $j \leftarrow 1 : d$
        $c_j \leftarrow i \bmod P_j$
        $i \leftarrow (i - c_j)/P_j$
    fim
    {vértices e volume da sub-região}
    para $j \leftarrow 1 : d$
        $a'_i \leftarrow a_i + d_i c_i$
        $b'_i \leftarrow a_i + d_i$
    fim
    {amostragem Monte Carlo}
    $[Q_{sub}, e_{sub}, n_{sub}] \leftarrow QMC2(FUN, \mathbf{a}, \mathbf{b}, d, e_{max}/\sqrt{N_p})$
    {acumulação de estimativas}
    $Q \leftarrow Q + Q_{sub}$
    $e \leftarrow \sqrt{e^2 + e_{sub}^2}$
    $n \leftarrow n + n_{sub}$
fim

A Figura 4.6 mostra a distribuição dos pontos amostrais na avaliação da integral $I_2 = (e - \frac{1}{e})^{-2} \int_{[-1,1]\times[-1,1]} e^{x_1} e^{x_2} dx_1 dx_2 = 1$ para um mesmo erro estimado $e_{max} = 0.02$. No método MC, pontos em toda a região de integração são amostrados com a mesma densidade. Já no método MC com amostragem estratificada (com $4 \times 4 = 16$ sub-regiões), a maior densidade de pontos amostrais situa-se próxima ao ponto $\mathbf{x} = (1, 1)$, pois ali a função integranda tem variância maior. Como resultado usual, o

método MC com a amostragem estratificada realiza um número menor de avaliações da função integranda do que o método MC.

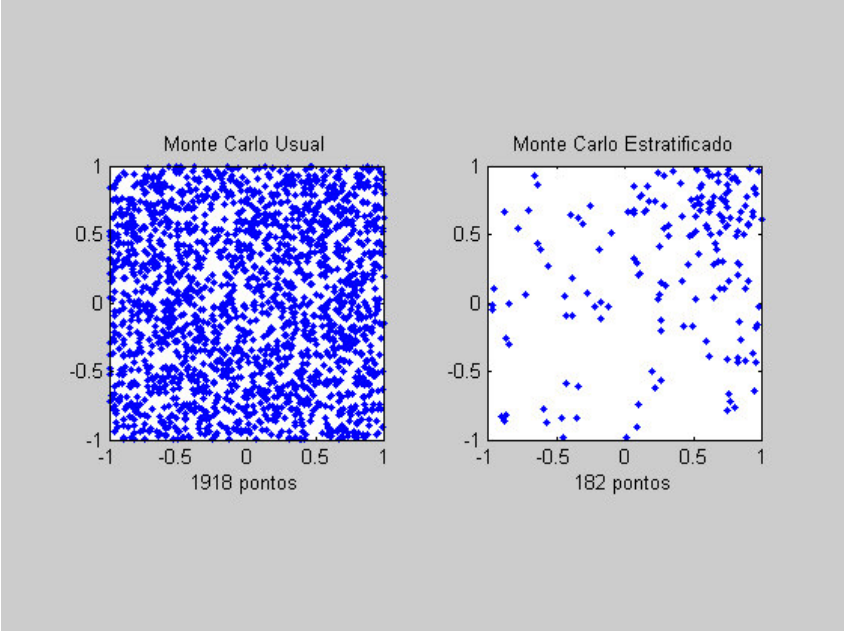

Figura 4.6: Distribuição dos pontos amostrais no método MC e MC estratificado.

4.3.4 Outras técnicas.

Existem muitas outras técnicas de redução de variância, além das técnicas descritas acima. Entre elas pode-se citar a técnica da **amostragem estratificada recursiva** (Press *at alli*, 1992) que realiza sucessivas bissecções na região de integração, uma dimensão por vez, ou **amostragem estratificada adaptável** (Thisted, 1996), que subdivide a região de integração de modo a minimizar o intervalo de variação da função; ou ainda a técnica de **amostragem de hipercubo latino** (Gentle, 1998), onde um mínimo de pontos amostrais, evitando-se mutuamente, é determinado de modo a cobrir a região de integração.

Outra técnica interessante é a da **amostragem de importância adaptável** (Press *et alii*, 1992). Essa técnica consiste em dividir a região de integração em S^d sub-regiões de forma regular, isto é, dividindo regularmente cada dimensão em S subintervalos. Uma função peso d-dimensional separável $g(x_1,x_2,...,x_d) := g_1(x_1)g_2(x_2)...g_d(x_d)$ é construída de forma adaptável. Cada função g_i tem valor constante, mas distinto em cada um dos S subintervalos. Esses valores são armazenados em uma matriz G de $S \times d$ elementos inicialmente iguais. A função integranda f é avaliada em cada uma das S^d sub-regiões e os elementos de G são recalculados de modo que $g \propto |f|$. Após cada uma das M iterações, os valores de Q e e são calculados de modo semelhante ao descrito na

subseção 4.3.1 a partir de $h := \frac{f}{g}$ e finalmente combinados em uma única estimativa por

$$Q^* := \sum_{i=1}^{M} \frac{Q_i}{e_i^2} / \sum_{i=1}^{M} \frac{1}{e_i^2} \quad \text{e} \quad e^* := \left(\sum_{i=1}^{M} \frac{1}{e_i^2} \right)^{-1/2}.$$

O valor

$$P := \frac{\chi^2}{M} = \frac{1}{M-1} \sum_{i=1}^{M} \frac{(Q_i - Q^*)^2}{e_i^2}$$

é calculado e, se for significativamente maior que 1, então os resultados das iterações são inconsistentes e as respostas, suspeitas. O algoritmo seguinte implementa essa abordagem adaptável:

Algoritmo 4.6: Quadratura Monte Carlo com amostragem de Importância Adaptável (QMCIA).

$[Q, e, P] \leftarrow QMCIA(f, \mathbf{A}, \mathbf{B}, S, M, N)$
{inicialização}
$d \leftarrow tamanho(\mathbf{A})$ {dimensão}
$\mathbf{Diag} \leftarrow (\mathbf{B} - \mathbf{A})/S$ {diagonal}
$Vol \leftarrow Diag_1 \times Diag_2 \times ... \times Diag_d$ {volume das sub-regiões}
para $i \leftarrow 1 : d$,
    para $j \leftarrow 1 : S$,
        $G_{ij} \leftarrow 1/(B_i - A_i)$ {FDP inicial}
    fim
fim
{amostragem}
para $i \leftarrow 1 : M$, {para cada iteração}
    {inicialização}
    para $j \leftarrow 1 : d$,
        para $k \leftarrow 1 : S$,
            $sg_{jk} \leftarrow 0$ {somador para G}
        fim
    fim
    $nt \leftarrow 0$ {número total de pontos}
    $sh \leftarrow 0$ {somador para h}
    $sh2 \leftarrow 0$ {somador para h^2}
    para $j \leftarrow 1 : S^d$ {para cada sub-região}
        {combinação para enumeração da sub-região}
        $temp \leftarrow j - 1$
        para $k \leftarrow 1 : d$
            $c_{k,1} \leftarrow temp \pmod{S}$
            $temp \leftarrow (temp - c_{k,1})/S$

{vértices da sub-região}
para $k \leftarrow 1 : d$
$a_k \leftarrow A_k + Diag_k \times c_k$
fim
$\mathbf{b} \leftarrow \mathbf{a} + \mathbf{Diag}$
{peso, valor de $g(x_1, x_2, ..., x_d)$ na sub-região}
$\mathbf{c} \leftarrow \mathbf{c} + 1$
$p \leftarrow 1$
para $k \leftarrow 1 : d$
$p \leftarrow p \times G_{k,c_k}$
fim
{amostragem}
$ns \leftarrow \lfloor N \times p \times Vol + 0.5 \rfloor$ {número de pontos na sub-região}
$nt \leftarrow nt + ns$
$sf \leftarrow 0$
para $k \leftarrow 1 : ns$
{ponto amostral}
para $l \leftarrow 1 : d$
$x_{l,1} \leftarrow a_l + Diag_l \times GU$
fim
$f \leftarrow F(x)$
$sf \leftarrow sf + f$
$h \leftarrow f/p$
$sh \leftarrow sh + h$
$sh2 \leftarrow sh2 + h \times h$
fim
{prepara novo vetor FDP}
se $ns \neq 0$,
para $k \leftarrow 1 : d$
$sg_{k,c_k} \leftarrow sg_{k,c_k} + \left| sf \,/\, ns \right|$
fim
fim
fim {fim da amostragem na sub-região}
{acumulação de estimativas}
$mh \leftarrow sh \,/\, nt$ $\{< h >\}$
$mh2 \leftarrow sh2 \,/\, nt$ $\{< h^2 >\}$
$QQ_i \leftarrow mh$
$ee_i \leftarrow \sqrt{(mh2 - mh^2) \,/\, nt}$
{novo vetor FDP}
para $j \leftarrow 1 : d$
$tot_j \leftarrow 1/\left(Diag_j \times \sum_{k=1}^{S} sg_{jk}\right)$
fim

para $j \leftarrow 1 : d$

para $k \leftarrow 1 : S$

$G_{j,k} \leftarrow sg_{j,k} \times tot_j$

fim

fim

fim {fim da iteração}

{melhor estimativa}

$t \leftarrow \sum_{i=1}^{M} \frac{1}{ee_i^2}$

$Q \leftarrow \sum_{i=1}^{M} \frac{QQ_i}{ee_i^2} / \sum_{i=1}^{M} \frac{1}{ee_i^2}$

$e \leftarrow \frac{1}{\sqrt{t}}$

$P \leftarrow \frac{1}{M-1} \sum_{i=1}^{M} \frac{(QQ_i - Q)^2}{ee_i^2}$

Na Figura 4.7, as marcas (•) e (+) representam os pontos $(n, \langle E \rangle_n)$ na avaliação da integral $I_5 = \left(\frac{3}{2}\right)^5 \int_{V_5} \sqrt{x_1 x_2 ... x_5}\, dx_1 dx_2 ... dx_5 = 1$ com $V_5 = [0,1]^5$. Os pontos (•) representam valores obtidos pelo método MC e as cruzes (+), os valores obtidos pelo método MC com amostragem de importância adaptável. Note-se que o erro cometido pelo método MC com amostragem adaptável diminui em relação ao erro cometido pelo método MC, à medida que o número de pontos amostrais aumenta.

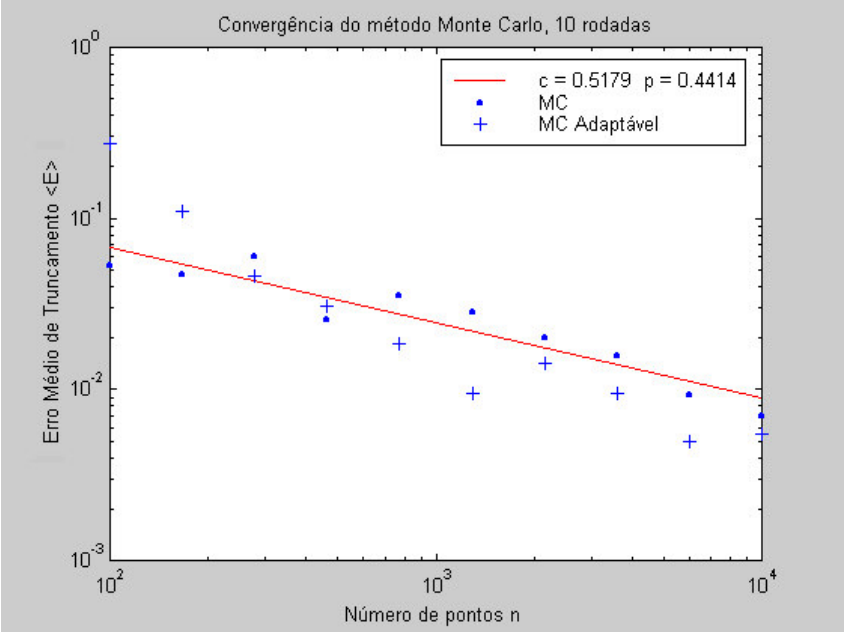

Figura 4.7: Convergência do método MC com amostragem de importância adaptável.

5 O método Quase-Monte Carlo na quadratura multidimensional

Como visto no Capítulo 4, a principal desvantagem do método Monte Carlo no cálculo de integrais multidimensionais é sua taxa de convergência de ordem $O(n^{-1/2})$. Essa ordem de convergência está associada à natureza intrinsecamente aleatória dos pontos de amostragem. Seria possível aumentar a taxa de convergência do método diminuindo essa aleatoriedade? Uma tentativa óbvia é usar um conjunto de pontos amostrais de forma absolutamente regular, que constitui o método da **Partição Regular,** que será visto na seção seguinte. No entanto, essa tentativa de aprimoramento através da redução total da aleatoriedade dos pontos amostrais não obtém sucesso. De forma talvez surpreendente, diminuir a aleatoriedade dos pontos amostrais, sem recair na partição regular, tem-se mostrado um método satisfatório. Esse método é denominado **Quase-Monte Carlo** (QMC) e será discutido a partir da seção 5.2.

5.1 Método da partição regular.

A idéia central do método MC para o cálculo de integrais unidimensionais é fazer uma estimativa do valor médio da função f no intervalo $[a;b]$, através da média $\langle f\rangle$ dos valores de f em n pontos escolhidos de forma aleatória nesse intervalo e, assim, aproximar a integral $\int_a^b f$ por $(b-a)\langle f\rangle$. No entanto, é também possível escolher esses pontos de forma regularmente espaçada. Essa escolha transformaria o método MC, estocástico, em um método determinístico mais simples. Essa nova abordagem é, de fato, uma regra de Newton-Cotes composta de ordem $m = 0$, denominada **regra dos retângulos** (Cláudio & Marins, 1994). A ordem de convergência desse método é estabelecida pelo seguinte teorema:

Teorema 5.1. *Seja* $f : [a;b] \to \Re$ *uma função de classe* C^1 e $\{x_i\} := \{a + (i-1)h\}$, $i = 1,...,n+1$, *uma partição regular do intervalo* $[a;b]$. *Sejam* $p_i := x_i$, $i = 1,...,n$, *os extremos inferiores dos subintervalos da partição. Então o erro de truncamento da aproximação* $Q := h\sum_{i=1}^{n} f(p_i)$ *para a integral* $\int_a^b f(x)dx$ *tem convergência* $O(n^{-1})$.

Demonstração. Pelo Teorema do Valor Médio para derivadas, para cada $x \in [p_i;p_{i+1}]$, existe $\xi_i(x) \in [p_i;x]$ tal que

$$f(x) = f(p_i) + f'(\xi_i(x))(x - p_i).$$

Daí,

$$\begin{aligned} I_i &:= \int_{p_i}^{p_{i+1}} f(x)dx \\ &= \int_{p_i}^{p_{i+1}} f(p_i)dx + \int_{p_i}^{p_{i+1}} f'(\xi_i(x))(x-p_i)dx. \end{aligned}$$

A função $x \mapsto \xi_i(x)$ é contínua no subintervalo $[p_i; p_{i+1}]$. Como f' é contínua, a função $x \mapsto f'(\xi_i(x))$ também é contínua nesse intervalo. Como $x - p_i$ não muda de sinal nesse mesmo intervalo, então, pelo Teorema do Valor Médio para integrais,

$$\begin{aligned} I_i &= f(p_i)(p_{i+1} - p_i) + f'(\eta_i) \int_{p_i}^{p_{i+1}} (x - p_i)dx \\ &= f(p_i)h + f'(\eta_i)\frac{h^2}{2}, \end{aligned}$$

para algum $\eta_i \in [p_i, p_{i+1}]$. Portanto I, valor da integral de f sobre o intervalo $[a; b]$, é dado por

$$\begin{aligned} I &= \sum_{i=1}^{n} I_i \\ &= \sum_{i=1}^{n} f(p_i)h + \sum_{i=1}^{n} \frac{1}{2} f'(\eta_i)h^2 \\ &= h\sum_{i=1}^{n} f(p_i) + \frac{h^2}{2}\sum_{i=1}^{n} f'(\eta_i). \end{aligned}$$

Como f' é contínua, pelo Teorema do Valor Intermediário existe um $\eta \in [a; b]$ tal que

$$f'(\eta) = \frac{1}{n}\sum_{i=1}^{n} f'(\eta_i).$$

Assim I pode ser expresso como

$$\begin{aligned} I &= h\sum_{i=1}^{n} f(p_i) + \frac{h^2}{2} n f'(\eta) \\ &= \underbrace{h\sum_{i=1}^{n} f(p_i)}_{Q} + \underbrace{\left[\frac{(b-a)^2}{2} f'(\eta)\right]\frac{1}{n}}_{E}. \end{aligned}$$

Como f' é limitada, por ser contínua em um intervalo compacto, tem-se, para o erro de truncamento E,

$$|E| \leq \left[\frac{(b-a)^2}{2} \sup\left|f'(\eta)\right|\right]\frac{1}{n}.$$

O valor da expressão do último colchete é constante; logo o erro E tem convergência de ordem $O(n^{-1})$. □

Esse método é, portanto, melhor que o método MC unidimensional, cuja convergência é $O(n^{-1/2})$. No entanto, sua extensão para o espaço d-dimensional não é tão eficaz. Essa extensão é feita tomando-se n pontos $\mathbf{p}_j$ de uma **partição regular d-dimensional** da região de integração V. O teorema abaixo estabelece a ordem de convergência dessa regra.

Teorema 5.2. *Seja*

1. $f : V \to \Re$ *uma função de classe* C^1 *definida numa região* $V := [a_1;b_1]\times[a_2;b_2]\times ...\times[a_d;b_d]$ *de volume* v;
2. *para cada* $i = 1,...,d$, *uma partição regular* $\{x_{i\,1},x_{i\,2},...,x_{i\,n_i},x_{i\,n_i+1}\}$ *do intervalo* $[a_i;b_i]$ *com* $x_{i\,j} := a_i + (j-1)h$, $j = 1,...,n_i+1$;
3. $\left(\mathbf{p}_j\right)_{j=1}^{n}$, *uma enumeração do produto cartesiano* $\{x_{11},x_{12},...,x_{1n_1}\}\times\{x_{21},x_{22},...,x_{2n_2}\}\times ...\times\{x_{d1},x_{d2},...,x_{d\,n_d}\}$.

Então a ordem de convergência da aproximação $Q := \frac{v}{n}\sum_{j=1}^{n} f(\mathbf{p}_j)$ *para a integral* $I := \int_V f dv$ *é* $O(n^{-1/d})$, *na melhor hipótese.*

Demonstração. A região de integração V é dividida em $n = n_1n_2...n_i$ sub-regiões, cujo volume é $\Delta v := v/n$. Seja $(\Delta V_j)_{j=1}^{n}$ a enumeração das sub-regiões tal que

$$\Delta V_j := [x_{1j};x_{1j}+h_1]\times[x_{2j};x_{2j}+h_2]\times...\times[x_{dj};x_{dj}+h_d] \quad \text{e} \quad \mathbf{p}_j := (x_{1j},x_{2j},...,x_{dj}).$$

Pelo Teorema do Valor Médio para derivadas, para qualquer ponto $\mathbf{x} := (x_1,x_2,...,x_d) \in \Delta V_j$, existe um ponto $\xi_j(\mathbf{x})$ tal que

$$f(\mathbf{x}) = f(\mathbf{p}_j) + \sum_{i=1}^{d}(x_i - x_{ij})\frac{\partial f\left(\xi_j(\mathbf{x})\right)}{\partial x_i}.$$

O valor da integral de $f(\mathbf{x})$ em ΔV_j é

$$I_j := \int_{\Delta V_j} f(\mathbf{p}_j)dv + \int_{\Delta V_j}\sum_{i=1}^{d}(x_i - x_{ij})\frac{\partial f\left(\xi_j(\mathbf{x})\right)}{\partial x_i}dv.$$

A primeira integral é imediata pois o integrando é constante. A segunda requer

alguma manipulação. Como f é da classe C^1, cada uma das parcelas é integrável. Logo, pela linearidade da integração, pode-se inverter a ordem da integração com o somatório:

$$I_j = f(\mathbf{p}_j)\Delta v + \sum_{i=1}^{d} \int_{\Delta V_j} (x_i - x_{ij}) \frac{\partial f\big(\xi_j(\mathbf{x})\big)}{\partial x_i} dv.$$

A função $\mathbf{x} \mapsto \xi_j(\mathbf{x})$ é contínua em ΔV_j, e $(x_i - x_{ij})$ não muda de sinal em ΔV_j. Então pelo Teorema do Valor Médio para integrais multidimensionais (Davis & Rabinowitz, 1975), existe um ponto $\eta_j \in \Delta V_j$ tal que

$$I_j = f(\mathbf{p}_j)\Delta v + \sum_{i=1}^{d} \frac{\partial f(\eta_j)}{\partial x_i} \int_{\Delta V_j} (x_i - x_{ij}) dv.$$

Expressando a integral do segundo membro como integral iterada, obtém-se

$$I_j = f(\mathbf{p}_j)\Delta v + \sum_{i=1}^{d} \frac{\partial f(\eta_j)}{\partial x_i} \int_{p_{dj}}^{p_{dj}+h_d} \ldots \int_{p_{2j}}^{p_{2j}+h_2} \int_{p_{1j}}^{p_{1j}+h_1} (x_i - x_{ij}) dx_1 dx_2 \ldots dx_d.$$

Com a mudança de variável $u_i := x_i - x_{ij}$, as integrais tornam-se

$$I_j = f(\mathbf{p}_j)\Delta v + \sum_{i=1}^{d} \frac{\partial f(\eta_j)}{\partial x_i} \int_0^{h_d} \ldots \int_0^{h_2} \int_0^{h_1} u_i \quad du_1 du_2 \ldots du_d,$$

do que resulta

$$I_j = f(\mathbf{p}_j)\Delta v + \sum_{i=1}^{d} \frac{\partial f(\eta_j)}{\partial x_i} h_1 . h_2 \ldots h_{i-1} . \frac{1}{2} h_i^2 . h_{i+1} \ldots h_d.$$

Como $\Delta v = h_1 h_2 \ldots h_d$, tem-se

$$I_j = f(\mathbf{p}_j)\Delta v + \frac{1}{2}\Delta v \sum_{i=1}^{d} h_i \frac{\partial f(\eta_j)}{\partial x_i}.$$

Agora, o valor da integral I sobre toda a região de integração V vale

$$
\begin{aligned}
I &= \sum_{j=1}^{n} I_j \\
&= \sum_{j=1}^{n} f(\mathbf{p}_j)\Delta v + \sum_{j=1}^{n} \frac{1}{2}\Delta v \sum_{i=1}^{d} h_i \frac{\partial f(\eta_j)}{\partial x_i} \\
&= \Delta v \sum_{j=1}^{n} f(\mathbf{p}_j) + \frac{1}{2}\Delta v \sum_{j=1}^{n}\sum_{i=1}^{d} h_i \frac{\partial f(\eta_j)}{\partial x_i} \\
&= \underbrace{\frac{v}{n}\sum_{j=1}^{n} f(\mathbf{p}_j)}_{Q} + \underbrace{\frac{v}{2n}\sum_{i=1}^{d} h_i \sum_{j=1}^{n} \frac{\partial f(\eta_j)}{\partial x_i}}_{E}
\end{aligned}
$$

O termo Q é uma aproximação para a integral I. E o termo E é o erro de truncamento, que pode ser simplificado, observando que, sendo f de classe C^1, pelo Teorema do Valor Intermediário existe um ponto η em V tal que

$$\frac{\partial f(\eta)}{\partial x_i} = \frac{1}{n}\sum_{j=1}^{n} \frac{\partial f(\eta_j)}{\partial x_i}.$$

Assim

$$E = \frac{v}{2}\sum_{i=1}^{d} h_i \frac{\partial f(\eta)}{\partial x_i},$$

e

$$|E| \leq \frac{v}{2}\sum_{i=1}^{d} h_i \left|\frac{\partial f(\eta)}{\partial x_i}\right| \leq \frac{v}{2}\max_{i=1,\ldots,d}\left|\frac{\partial f(\eta)}{\partial x_i}\right| \sum_{i=1}^{d} h_i = c\sum_{i=1}^{d} h_i.$$

onde c é uma constante positiva. O valor da última expressão, sujeito à restrição de ser $h_1 h_2 ... h_d = \Delta v$ constante, é mínimo quando $h_1 = h_2 = ... = h_d =: h$. De outro modo, para uma dada sub-região de V de volume Δv fixo, a forma hipercúbica de ΔV_j minimiza o erro. Nesse caso,

$$\sum_{i=1}^{d} h_i = dh = d(\Delta v)^{1/d} = d\left(\frac{v}{n}\right)^{1/d},$$

e, então,

$$|E| \leq cd\left(\frac{v}{n}\right)^{1/d} = [cdv^{1/d}]n^{-1/d},$$

o que expressa que a ordem de convergência desse método é, na melhor hipótese, $O(n^{-1/d})$. □

Desse modo, comparando-se o método PR de convergência $O(n^{-1/d})$ com o método MC de convergência $O(n^{-1/2})$, conclui-se que o primeiro é superior ao segundo para $d = 1$,

equivalente para $d = 2$, e inferior para $d \geq 3$.

O algoritmo seguinte determina uma estimativa para o valor da quadratura Q, dados a função f, os vértices $\mathbf{a} := (a_1, a_2, ...a_d)$ e $\mathbf{b} := (b_1, b_2, ...b_d)$ da região de integração, a dimensão d e o vetor $\mathbf{n} := (n_1, n_2, ..., n_d)$ com o número de pontos de amostragem em cada dimensão. De acordo com o Teorema 5.2, convém usar $n_1, n_2, ..., n_d$ de modo que $h_1 \approx h_2 \approx ... \approx h_d$.

Algoritmo 5.1: Quadratura d-dimensional pelo método de Partição Regular (QPR).

$Q \leftarrow QPR(f, \mathbf{a}, \mathbf{b}, d, \mathbf{n})$,
{inicialização}
para $i \leftarrow 1 : d$,
    $L_i \leftarrow B_i - A_i$ {arestas da região de integração}
    $H_i \leftarrow L_i/n_i$ {tamanho da partição em dada dimensão}
    $X_{i1} \leftarrow A_i + H_i/2$ {primeira coordenada de cada dimensão }
fim
$v \leftarrow L_1 \times L_2 \times ... \times L_d$ {volume da região de integração}
$n \leftarrow n_1 \times n_2 \times ... \times n_d$ {número de pontos da partição}
para $i \leftarrow 1 : d$,
    para $j \leftarrow 2 : n_i$,
        $X_{ij} \leftarrow X_{i,j-1} + H_i$ {coordenadas restantes}
    fim
fim
{amostragem}
$s \leftarrow 0$ {acumulador para $< f >$}
para $j \leftarrow 0 : n - 1$,
    para $i \leftarrow 1 : d$, {índices para coordenadas}
        $k \leftarrow j \bmod n_i$
        $j \leftarrow (j - k)/n_i$
        $x_i \leftarrow X_{i,k+1}$
    fim
    $s \leftarrow s + f(\mathbf{x})$
fim
{estimativa}
$Q \leftarrow \frac{vs}{n}$

A Figura 5.1 mostra em um gráfico na escala log-log o desempenho do método da Partição Regular. Cada ponto (n, E), onde n é o número de pontos amostrais e E é o erro de truncamento correspondente a avaliação da integral d-dimensional $I_d := \left(\frac{3}{2}\right)^d \int_{V_d} \sqrt{x_1 x_2 ... x_d}\, dx_1 dx_2 ... dx_d = 1$ com $V_d := [0,1] \times [0,1] \times ... \times [0,1]$. A reta (em escala log-log) é o ajustamento pelos mínimos quadrados da curva $E = cn^{-p}$, baseado

no conjunto de dados $\{(n,E)\}$, obtidos pelo método PR. Compare-se a Figura 5.1 com as Figuras 4.2 e 4.3.

Observação: *As Figuras 5.3 e 5.5 mostradas abaixo são semelhantes à Figura 5.1. As descrições da escala log-log e da reta de ajuste são as mesmas. Evita-se, portanto, repeti-las.*

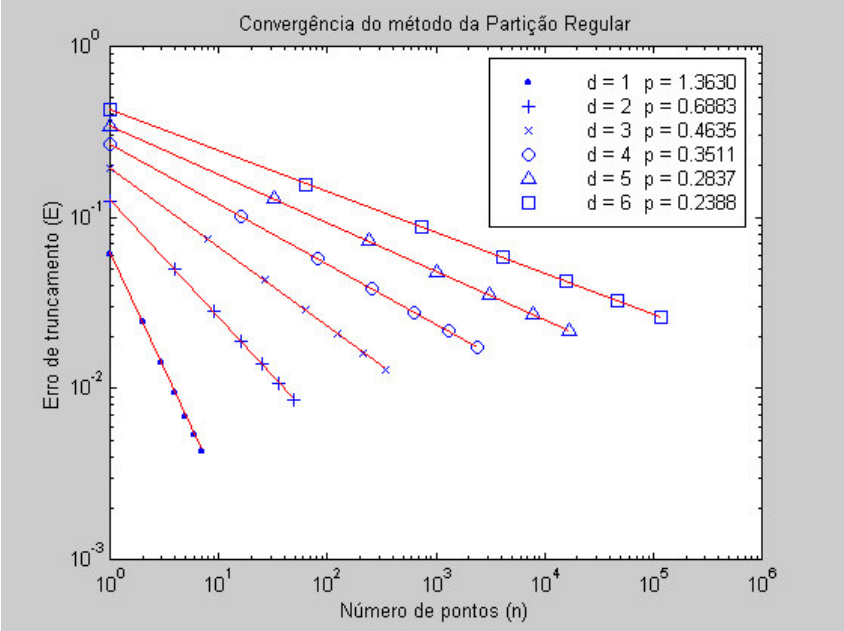

Figura 5.1: A convergência $O(n^{-1/d})$ do método da partição regular.

A Tabela 5.1 mostra os valores de c e p obtidos nos ajustamentos. Pode-se observar que p_d é proporcional a $1/d$. Ou ainda, que, se $p_1 = 1.3630$, então $p_d \approx p_1/d$.

d	c_d	p_d	$\frac{p_1}{d}$
1	0.0619	1.3630	–
2	0.1271	0.6883	0.6815
3	0.1957	0.4634	0.4543
4	0.2677	0.3510	0.3408
5	0.3435	0.2836	0.2726
6	0.4233	0.2387	0.2272

Tabela 5.1: Parâmetros de ordem de convergência do método PR.

O método da partição regular sofre de muitas dificuldades. Primero, para dimensões elevadas, o número n de pontos necessários para obter uma partição adequada cresce exponencialmente com d. Segundo, na seqüência usual de amostragem da partição, em planos $(d-1)$-dimensionais sucessivos, a distribuição dos pontos amostrais não é uniforme, isto é, existem muitos pontos em uma região e nenhum em outra. Assim, o erro de truncamento não decresce de modo suave com o incremento de pontos. Por fim, o único método óbvio de refinar a partição requer 2^d vezes o número atual de pontos (Morokoff & Caflisch, 1995).

No entanto, há uma conexão importante entre o método da partição regular e o método Monte Carlo com amostragem estratificada, visto na subseção 4.3.3. A idéia da estratificação para reduzir a variância de uma amostra pode ser estendida: à medida que o número de estratos aumenta, o tamanho da amostra em cada estrato diminui. No limite, a aleatoriedade dos pontos amostrais desaparece.

Haber (1970) mostrou que, se apenas um ponto amostral é escolhido aleatoriamente no interior de cada uma das n sub-regiões definidas pela partição, então o erro de truncamento E tem convergência acelerada para $O(n^{-(1/2+1/d)})$ (Davis & Rabinowitz, 1975). Esse resultado indica que, se a distribuição dos pontos amostrais não for tão rígida quanto no método PR, nem tão aleatória quanto no método MC, pode-se obter resultados melhores. O método que utiliza um preenchimento do espaço dessa forma foi desenvolvido a partir desses resultados e denominado método **Quase-Monte Carlo**.

5.2 O método Quase-Monte Carlo.

No método Monte Carlo, as seqüências de pontos pseudo-aleatórios simulam pontos verdadeiramente aleatórios *independentes* e preenchem o espaço do modo *suficientemente* uniforme, porém com flutuações de densidade. A análise do comportamento do valor da integral baseia-se no conceito estatístico de valor esperado $Esp[\bullet]$:

$$Q = Esp\left[\frac{v}{n} \sum_{i=1}^{n} f(\mathbf{x}_i) \right].$$

No método Quase-Monte Carlo (QMC), obtêm-se seqüências de pontos não-independentes, que preenchem o espaço de modo mais uniforme, com mínima flutuação de densidade. A análise do comportamento do valor da integral baseia-se no conceito analítico de limite de uma seqüência

$$Q = \lim_{n\to\infty} \frac{v}{n} \sum_{i=1}^{n} f(\mathbf{x}_i). \qquad (5.1)$$

5.2.1 Seqüências de baixa discrepância.

As seqüências de pontos que preenchem o espaço de forma mais uniforme que pontos aleatórios independentes são chamadas de **seqüências quase-aleatórias** ou **seqüências de baixa discrepância**. Duas dessas seqüências serão vistas abaixo.

A **discrepância** de uma seqüência $(\mathbf{x}_i) := (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$ de n pontos no hipercubo unitário d-dimensional $V := [0;1]\times[0;1]\times...\times[0;1]$ é definida por

$$D(\mathbf{x}_i) := \sup_B \left| \frac{N(B, \mathbf{x}_i)}{n} - v(B) \right|,$$

onde o supremo é tomado sobre todas as sub-regiões B retangulares com arestas paralelas aos eixos contidas no interior de V, o valor $N(B, \mathbf{x}_i)$ é o número de pontos $\mathbf{x}_i$ no interior de B e $v(B)$, o volume de B (Morokoff & Caflish, 1995). Resulta que uma seqüência de pontos de baixa discrepância preenche o espaço de forma mais uniforme que seqüências de alta discrepância.

A aceitação do método QMC decorre do resultado obtido inicialmente por Hlwaka (1961) e generalizado por Hickernell (1998) na análise da seqüência (5.1), que determina uma cota superior para o erro $E := I - Q$ na estimativa pelo método QMC de integrais multidimensionais. Se f é uma função d-dimensional, definida na região $V := [0;1]\times[0;1]\times...\times[0;1]$, então

$$|E| \leq D(\mathbf{x}_i) W(f), \tag{5.2}$$

onde $W(f)$ é a **variação total** de f na região V definida por

$$W(f) := \sup_P \left[\sum_{i,j=1}^{n} |f(\mathbf{x}_i) - f(\mathbf{x}_j)| \right], \tag{5.3}$$

onde o supremo é tomado sobre todas as partições P de V (Hewitt & Stromberg, 1965).

Esse resultado fornece uma estimativa para o erro de integração que depende apenas da natureza da função f e da discrepância da seqüência $\mathbf{x}_i$. Para um dado problema de integração, pode-se tentar reduzir a variação total da função (Paskov, 1996) ou diminuir a discrepância da seqüência utilizada.

A seqüência de discrepâncias associadas às seqüências pseudo-aleatórias $D^*(\mathbf{x}_i)$, usadas no método MC, são majoradas pela seqüência

$$c_1 \frac{\log \log n}{n^{1/2}},$$

enquanto que, segundo Faure (1982), muitas das seqüências quase-aleatórias, usadas no método QMC (entre as quais se situam as de Halton e Sobol, descritas adiante), possuem seqüências de discrepâncias $D^{**}(\mathbf{x}_i)$ majoradas pela seqüência

$$c_2 \frac{(\log n)^d}{n}.$$

Isso sugere que $D^{**}(\mathbf{x}_i) \leq D^{*}(\mathbf{x}_i)$, pelo menos para baixas dimensões (Morokoff & Caflisch, 1995). Roth (1954) mostrou que as seqüências de discrepâncias $D^{**}(\mathbf{x}_i)$ são minoradas pela seqüência

$$c_3 \frac{(\log n)^{(d-1)/2}}{n}.$$

A seqüência de Roth é uma espécie de "cota inferior" para as seqüências de discrepâncias das seqüências quase-aleatórias. Isso significa que a seqüência quase-aleaória de menor discrepância possível teria essa seqüência de discrepâncias. No entanto, as seqüências quase-aleatórias estudadas até o momento ainda estão acima (pelo menos, para dimensão elevada) dessa "cota inferior". Assim, surge a motivação de procurar seqüências quase-aleatórias de discrepância ainda menores.

5.2.2 Seqüências de Halton.

Halton (1960) propôs um método de geração de seqüências de baixa discrepância conceitualmente simples. Nessas, o i-ésimo número u_{ib} é obtido com os seguintes passos:

1. escreve-se i como um número numa base b, sendo b algum número primo; por exemplo, $i = 17 = 122_3$ (base $b = 3$);
2. obtém-se u_{ib} invertendo a ordem dos dígitos obtidos e antepondo um ponto indicador de fração base b; no exemplo, $u_{17,3} = 0.221_3$;
3. converte-se o número u_{ib} na base b para a base decimal; no exemplo, $u_{17,3} = \frac{25}{27} = 0.925926...$; para cada número primo b obtém-se uma seqüência distinta; cada elemento u_{ib} da seqüência pertence ao intervalo $(0;1)$ como no caso dos números pseudo-aleatórios obtidos pelos geradores uniformes.

As seqüências de Halton são uma generalização das seqüências de van der Corput de base 2 (Gentle, 1998). De forma mais precisa, se i é decomposto na base b por

$$i = \sum_{j=0}^{t_i} a_j b^j,$$

então u_{ib} é obtido por

$$u_{ib} = \sum_{j=0}^{t_i} a_j b^{i-t_i-1}. \tag{5.4}$$

A seqüência de Halton, de base $b = 3$, por exemplo, é $(0, \frac{1}{3}, \frac{2}{3}, \frac{1}{9}, \frac{4}{9}, \frac{7}{9}, \frac{2}{9}, \frac{5}{9}, \frac{8}{9}, \frac{1}{27}, \frac{10}{27}, \frac{19}{27}, ...)$. Não é difícil verificar o funcionamento desse método. Cada vez que o número i aumenta de um dígito base b, os valores de u_{ib} passam a cobrir regiões vazias por entre os números anteriores, de uma partição cujos subintervalos têm comprimento b vezes menores. Assim, esse método permite o preenchimento não-seqüencial da partição regular com espaçamento cada vez mais diminuto.

O algoritmo seguinte calcula o i-ésimo valor u_{ib} da seqüência de Halton, base b, no intervalo $[0, 1)$, dados os valores de i e da base b.

Algoritmo 5.2: Gerador de Halton (GH).
$u \leftarrow GH(i, b)$,
$j \leftarrow 0$
enquanto $i > 0$, {decompõe i na base b}
    $j \leftarrow j + 1$
    $a_j \leftarrow i \bmod b$
    $i \leftarrow (i - a_j)/b$
fim
$u \leftarrow 0$
enquanto $j > 0$, {calcula u_{ib}}
    $u \leftarrow (u + a_j)/b$
    $j \leftarrow j - 1$
fim

Para gerar uma seqüência de pontos $(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$ no espaço d-dimensional, com $\mathbf{x}_i = (x_{i1}, x_{i2}, ..., x_{id})$, $1 \le i \le n$, cada coordenada x_{ij}, $1 \le j \le d$, é calculada a partir do i-ésimo termo u_{ib_j} de uma seqüência de Halton (u_{ib_j}) com uma base b_j diferente (b_j varia com j). Tipicamente, são usados os d primeiros primos (Gentle, 1998).

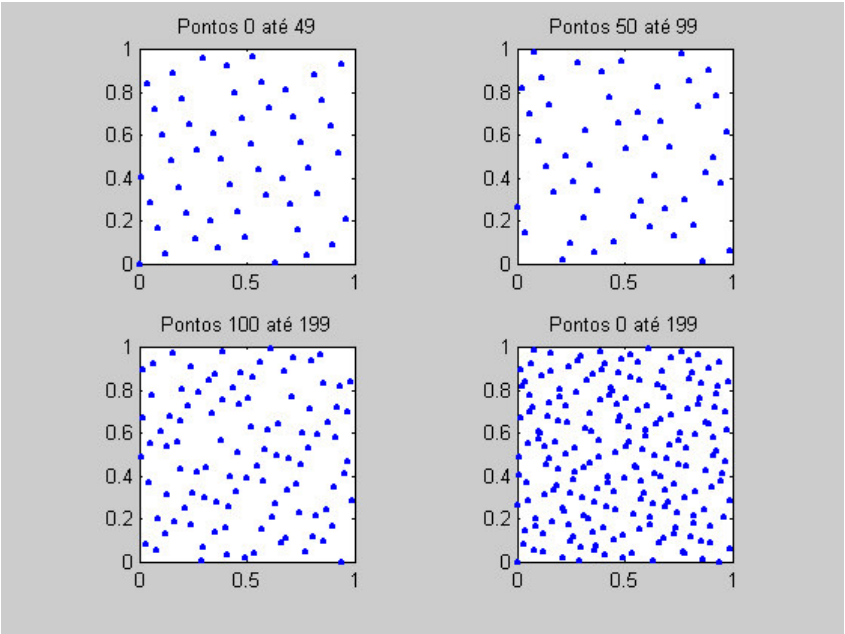

Figura 5.2: Preenchimento do espaço bidimensional pela seqüência de Halton.

A Figura 5.2 mostra os primeiros conjuntos de pontos (u_{i3}, u_{i5}) gerados dentro da região $V = [0,1]\times[0,1]$. Note-se que o preenchimento do espaço é mais uniforme que o da Figura 2.1 das seqüências pseudo-aleatórias.

A seqüência de Halton de pontos $\mathbf{x}_i$ em $\Re^d$ é aceitavelmente uniforme para baixas dimensões. Para dimensões mais altas, entretanto, a qualidade dessas seqüências cai levemente, pois a base b deve ser grande, de forma que o período de varredura do intervalo é grande (Morokoff & Caflisch, 1995). Quando se observa a representação geométrica do conjunto dos pontos $\{(u_{ib_j}, u_{ib_{j+1}}) \mid i = 1,...,n\}$, onde (u_{ib_j}) é uma seqüência de base b_j grande ($b_j > 200$) e $(u_{ib_{j+1}})$ é outra seqüência de base b_{j+1} também grande, seja $b_j = 211$ e $b_{j+1} = 223$, pode-se verificar que os pontos se agrupam sobre retas paralelas, isto é, as mencionadas seqüências se tornam fortemente correlacionadas. Para evitar esse efeito, Kocis & Whiten (1997) sugerem que apenas os l-ésimos termos da seqüência de Halton sejam utilizados, sendo l um número primo distinto de $b_1, b_2, ...b_d$. Seqüências de Halton generalizadas têm sido estudadas, sendo a idéia dominante uma permutação dos valores de a_j em (5.4), como forma de contornar esse problema. Dessa maneira são formadas as seqüências de Faure (1986, 1990).

O algoritmo seguinte determina estimativas para o valor da quadratura Q, dada a função f, a região de integração V definida pelos vértices $\mathbf{a} = (a_1, a_2, ...a_d)$ e $\mathbf{b} = (b_1, b_2, ...b_d)$, a dimensão d, o número n de pontos de amostragem e o vetor $\mathbf{p}$ contendo os primeiros d números primos.

Algoritmo 5.3. Quadratura Quase-Monte Carlo com seqüências de Halton (QH).
$Q \leftarrow QH(f, \mathbf{a}, \mathbf{b}, d, n, \mathbf{p})$,
{inicialização}
$s \leftarrow 0$ {acumulador para $< f >$}
$v \leftarrow 1$ {acumulador para volume da região de integração}
para $i \leftarrow 1 : d$
    $L_i = b_i - a_i$ {arestas da região de integração}
    $v \leftarrow vL_i$ {volume da região de integração}
fim
{amostragem}
para $j = 1 : n$
    para $i \leftarrow 1 : d$
        $x_i \leftarrow a_i + L_i \times GH(j, p_i)$ {ponto de amostragem}
    fim
    $s \leftarrow s + f(\mathbf{x})$
end
$Q \leftarrow v \times s/n$

A Figura 5.3 mostra o desempenho do método Quase-Monte Carlo (QMC) com seqüências de Halton na avaliação da integral d-dimensional $I_d = \left(\frac{3}{2}\right)^d \int_{V_d} \sqrt{x_1 x_2 ... x_d}\, dx_1 dx_2 ... dx_d = 1$ com $V_d = [0,1]\times[0,1]\times ... \times [0,1]$. Note-se que a ordem de convergência desse método ($p > 0.5$) é melhor que a convergência do método MC ($p \approx 0.5$). Compare-se com a Figura 5.1.

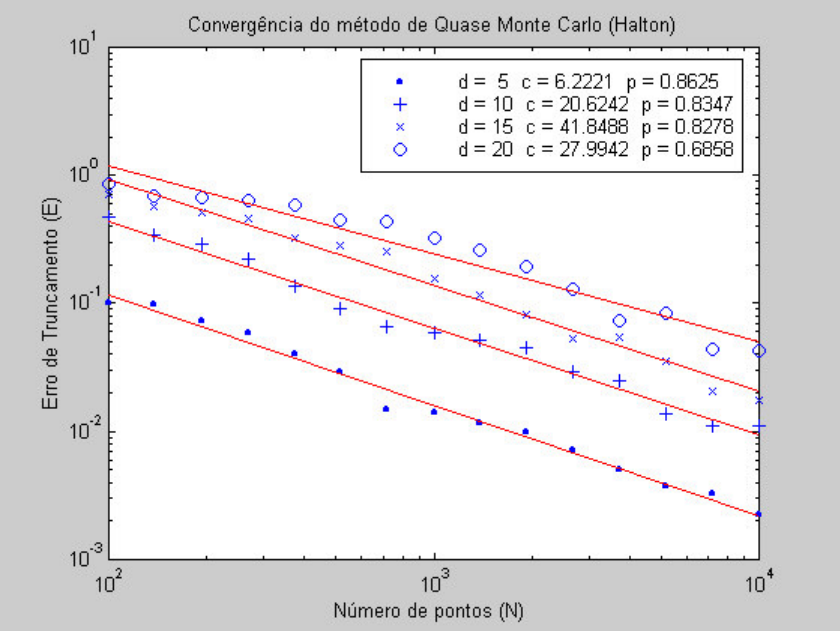

Figura 5.3: Convergência $O(n^{-1})$ do método Quase-Monte Carlo com seqüências de Halton.

5.2.3 Seqüências de Sobol.

Outra seqüência de números quase-aleatórios bastante utilizada foi introduzida por Sobol (1967) e tornada mais eficiente por Antonov & Saleev (1979). De fato, essas seqüências são permutações distintas da seqüência de Halton base $b = 2$ (Morokoff & Caflisch, 1995). Essas seqüências tomam por base um conjunto de números v_i, denominados de **números direcionais**. Os números v_i são

$$v_i = \frac{m_i}{2^i},$$

onde m_i é um número inteiro positivo menor que 2^i, escolhido de modo a satisfazer a relação de recorrência

$$m_i = 2c_1 m_{i-1} \oplus 2^2 c_2 m_{i-2} \oplus ... \oplus 2^q c_q m_{i-q} \oplus m_{i-q},$$

onde c_i e q são, respectivamente, os coeficientes e o grau de algum polinômio primitivo módulo 2 (subseção 2.3.5)

$$P(z) = z^q + c_1 z^{q-1} + ... + c_{q-1} z + 1.$$

O símbolo $\oplus$ denota a adição $(\mathrm{mod}\, 2)$ ou, equivalentemente, operação lógica binária XOR (OU exclusivo). Os valores iniciais $m_1, ..., m_q$ são escolhidos ímpares menores que $2, ..., 2^q$, respectivamente. Bratley & Fox (1988) discutem os critérios de seleção desses valores.

O j-ésimo termo da seqüência original de Sobol é, então, o número S_j calculado por $S_j = b_1 v_1 \oplus b_2 v_2 \oplus ... \oplus b_w v_w$, onde $b_w ... b_2 b_1$ é a representação binária de j. Antonov & Saleev mostraram que uma seqüência equivalente pode ser mais eficientemente calculada por $S_j = S_{j-1} \oplus v_k$, onde k é a posição do bit 0 mais a direita da representação binária de $j - 1$.

Para gerar uma seqüência de pontos $\mathbf{x}_j$ no espaço $\Re^d$, cada coordenada de $\mathbf{x}j$ é calculada a partir de uma seqüência de Sobol $(S_{j1}, S_{j2}, ..., S_{jd})$ com um polinômio diferente.

Os primeiros polinômios primitivos módulo 2 são: $z + 1$, $z^2 + z + 1$, $z^3 + z + 1$, $z^3 + z^2 + 1$, ... Como os coeficientes c_i desses polinômos valem apenas 0 ou 1, é possível representá-los por seu grau p e um número $a = (c_1 c_2 c_3 ...)_2$, cuja representação binária seja idêntica à seqüência dos coeficientes c_i.

Assim, por exemplo, o polinômio $z^3 + z^2 + 1$, cujo grau é $q = 3$, e cujos coeficientes são $c_1 = 1$ e $c_2 = 0$, pode ser representado por $a = 2 = 10_2$.

O algoritmo seguinte determina d (≤ 25) seqüências de Sobol-Antonov-Saleev com n termos cada uma. Os coeficientes foram extraídos de Press *et alii* (1992). $\mathbb{P}$ é um conjunto de índices para a e q. Cada seqüência será gerada a partir de a_{P_i} e q_{P_i}.

Algoritmo 5.4: Gerador Sobol (GS).
$S \leftarrow GS(\mathbb{P}, n)$,
{inicialização}
$w \leftarrow \lceil \log_2 n \rceil$
$q \leftarrow (2,3,3,4,4,5,5,5,5,5,5,6,6,6,6,6,6,7,7,7,7,7,7,7,7)$ {grau do polinômio}
$a \leftarrow (1,1,2,1,4,2,4,7,11,13,14,1,13,16,19,22,25,1,4,7,8,14,19,21,28)$ {coeficientes do pol.}
para $i = 1 : tamanho(\mathbb{P})$
    {cálculo de m}
    para $j \leftarrow 1 : q_{p_i}$
        $m_{i,j} \leftarrow 2^j - 1$
    fim
    para $j \leftarrow q_{p_i} + 1 : w$
        $m_{i,j} \leftarrow 2^{q_{p_i}} \times m_{i,j-q_i} \oplus m_{i,j-q_{p_i}}$
        $t \leftarrow a_{p_i}$
        para $k \leftarrow 1 : q_{p_i} - 1$
            $r \leftarrow t \bmod 2$
            $m_{i,j} \leftarrow m_{i,j} \oplus 2^{q_{p_i}-k} \times r \times m_{i,j-q_{p_i}+k}$
            $t \leftarrow (t - r)/2$
        fim
    fim
    {cálculo de V}
    para $j \leftarrow 1 : w$
        $V_{i,j} \leftarrow \frac{m_{i,j}}{2^j}$
    fim
    {cálculo de S}
    $S_{i,1} \leftarrow 0$
    para $j \leftarrow 1 : n - 1$
        $t \leftarrow j - 1$
        $k \leftarrow 1$
        $r \leftarrow t \bmod 2$
        enquanto $r \neq 0$ {acha bit zero mais à direita}
            $t \leftarrow (t - r)/2$
            $k \leftarrow k + 1$

$r \leftarrow t \bmod 2$

fim

se $k > w$

$k \leftarrow w$

fim

$S_{i,j+1} \leftarrow S_{i,j} \oplus V_{i,k}$

fim

fim

A Figura 5.4 mostra os primeiros conjuntos de pontos gerados dentro da região $V = [0,1]\times[0,1]$. As coordenadas de cada ponto $(S_{1,j}, S_{2,j})$ são obtidas de seqüências de Sobol distintas.

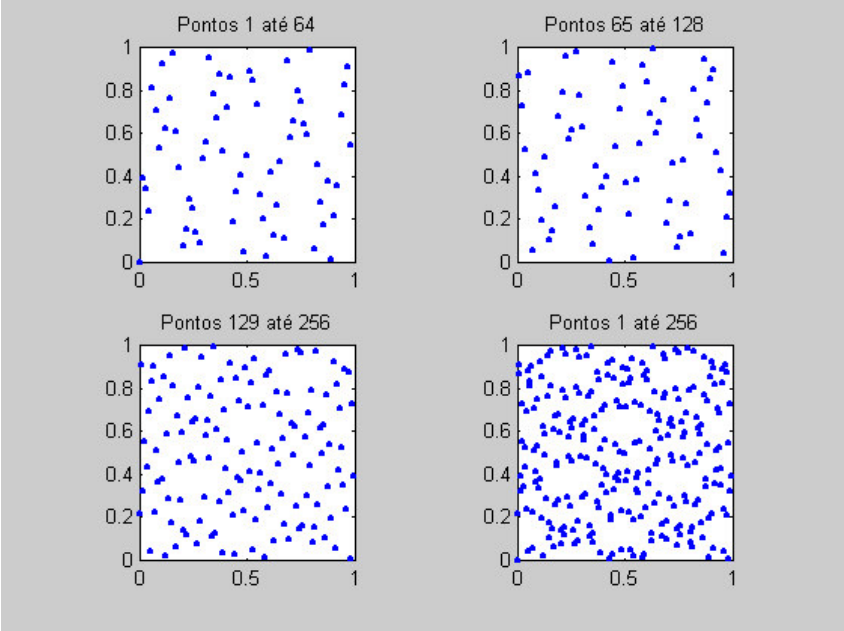

Figura 5.4: Preenchimento do espaço bidimensional pela seqüência de Sobol.

O algoritmo seguinte determina estimativas para o valor da quadratura Q, dadas a função f, e a região de integração V definida pelos vértices $\mathbf{a} = (a_1, a_2, ...a_d)$ e $\mathbf{b} = (b_1, b_2, ...b_d)$. Informa-se ainda a dimensão d, o número n de pontos de amostragem e o vetor $\mathbb{P}$ contendo os índices da seqüência de Sobol.

Algoritmo 5.5. Quadratura Quase-Monte Carlo com seqüências de Sobol (QS).

$Q \leftarrow QS(f, \mathbf{a}, \mathbf{b}, d, n, \mathbb{P})$,

$s \leftarrow 0$ {acumulador para $< f >$}

$v \leftarrow 1$ {acumulador para volume da região de integração}

para $i \leftarrow 1 : d$
    $L_i = b_i - a_i$ {arestas da região de integração}
    $v \leftarrow vL_i$ {volume da região de integração}
fim
$S \leftarrow GS(P, N + N_m)$; {seqüências de Sobol}
{amostragem}
para $j = N_m + 1 : N_m + n$ {despreza primeiros termos}
    para $i \leftarrow 1 : d$
        $x_i \leftarrow a_i + L_i \times S(i,j)$ {ponto de amostragem}
    fim
    $s \leftarrow s + f(\mathbf{x})$
end
$Q \leftarrow v \times s/n$

A Figura 5.5 mostra o desempenho do método Quase-Monte Carlo com seqüência de Sobol na avaliação da integral d-dimensional $I_d = \left(\frac{3}{2}\right)^d \int_{V_d} \sqrt{x_1 x_2 ... x_d}\, dx_1 dx_2 ... dx_d = 1$ com $V_d = [0,1]\times[0,1]\times ... \times [0,1]$. Note-se que, apesar do erro inicial grande, a ordem de convergêcia desse método ($p > 1$) é melhor que a convergência do método QMC com seqüências de Halton ($0.5 < p < 1$).

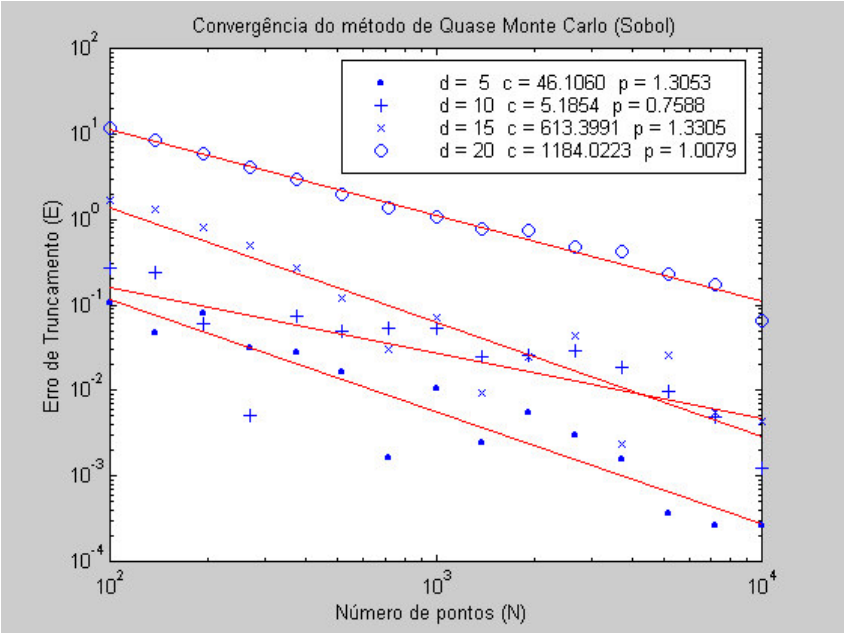

Figura 5.5: Convergência $O(n^{-1})$ do método Quase-Monte Carlo com seqüências de Sobol.

5.2.4 Comparação entre seqüências.

Entre as primeiras seqüências de baixa discrepância estudadas encontram-se as de

Halton e Sobol, descritas anteriormente. Outras seqüências, como as de Faure (1990), Niederreiter (1994) ou Fang & Wang (1994), têm sido propostas, mas são essencialmente variações das seqüências de Halton e Sobol.

Várias comparações empíricas entre as diversas seqüências quase-aleatórias têm sido feitas na tentativa de estabelecer ordem de qualidade. No entanto a literatura é inconclusiva a esse respeito, e parece não haver uma seqüência que seja predominantemente superior a outra (Gentle, 1998). Experimentos realizados por Morokoff & Caflisch (1995) sugerem que as seqüências de Halton produzem melhores resultados que as seqüências de Sobol, enquanto experimentos realizados por Paskov (1996) mostram o contrário.

5.2.5 Estimativa de erro.

A cota dada por (5.2) para o valor máximo do erro de truncamento E na avaliação da integral não é prática por duas razões. Primeiro porque a avaliação da variação total $W(f)$ de f na região V e da discrepância $D_n(\mathbf{x}_i)$ pode ser muito trabalhosa para dimensões elevadas (Kocis & Whiten, 1997). E, em segundo lugar, estudos empíricos sugerem que estimativas obtidas por (5.2) são demasiadamente elevadas (Morokoff & Caflisch, 1995).

Um modo de obter estimativas razoáveis de confiança é repetir k vezes o processo de integração (permutando-se as seqüências) e estimar valor médio e desvio padrão das estimativas $Q_1, Q_2, ..., Q_k$ obtidas.

Para exemplificar, repetiu-se $k = 3$ vezes o cálculo da integral $I_5 = \left(\frac{3}{2}\right)^5 \int_{V5} \sqrt{x_1 x_2 ... x_5}\, dx_1 dx_2 ... dx_5 = 1$ (com seqüências de Sobol diferentes) obtendo-se os valores $Q_1 = 0.99969121439215$, $Q_2 = 0.99994365436921$, $Q_3 = 1.00239750107767$. Então, conforme acima, resultou $Q = 1.00068$ $e = 0.00149$.

6 Exemplos de quadraturas multidimensionais.

Neste capítulo, são resolvidas algumas integrais com dimensão $d > 1$, como exemplo de aplicação das técnicas e dos algoritmos descritos nos Capítulos 4 e 5. Algumas integrais possuem resposta conhecida, outras não a possuem ou são de obtenção não-trivial. Essas integrais foram usadas como exemplo na literatura ou propostas como exercícios.

Os algortitmos utilizados são: Monte Carlo (QMC), Monte Carlo com parada por tolerância (QMC2), Monte Carlo com amostragem de importância (QMCI), Monte Carlo com amostragem antitética (QMCA), Monte Carlo com amostragem estratificada (QMCE), Monte Carlo com amostragem de importância adaptável (QMCIA), Quase-Monte Carlo com seqüências de Halton (QH) e Quase-Monte Carlo com seqüências de Sobol (QS).

Esses cálculos são efetuados com o estabelecimento, arbitrário, dos seguintes critérios de parada:

1. para os algoritmos QMC, QMCI, QMCA, QMCIA, QH e QS, número de pontos amostrais de $n = 50000$; nos algoritmos QMCIA, QH e QS foram efetuadas 5 rodadas de 10000 pontos amostrais;
2. para os algoritmos QMC2 e QMCE, a tolerância e_{max} foi estabelecida como 1% do valor estimado de Q.

A estimativa de erro para os algoritmos QH e QS foi determinada de acordo com o exposto na subseção 5.2.5.

6.1 Porção de toro ($d = 3$).

Esse exemplo é proposto por Press *et alii* (1992) como exemplo de aplicação do método MC. O autor não fornece valores para referência. Quer-se determinar a massa m e a posição $r_{cm} = (x_{cm}, y_{cm}, z_{cm})$ do centro de massa de uma porção do toro centrado na origem, de raio interno 2 e raio externo 4, cuja superfície é dada por

$$z^2 + \left(\sqrt{x^2 + y^2} - 3\right)^2 = 1.$$

Trata-se da porção V do toro contida na região $x \geq 1$ e $y \geq -3$. As coordenadas do centro de massa de V são dadas por

$$x_{cm} = \frac{m_x}{m}, \quad y_{cm} = \frac{m_y}{m}, \quad z_{cm} = \frac{m_z}{m},$$

onde

$$I_1 = m = \int_V \rho dxdydz, \qquad I_2 = m_x = \int_V x\rho dxdydz, \qquad I_3 = m_y = \int_V y\rho dxdydz, \text{ e } I_4 = m_z = \int_V z\rho dxdydz.$$

A densidade do toróide é $\rho = e^{5z}$. A região retangular de amostragem será $\{(x,y,z)|\ 1 \le x \le 4,\ -3 \le y \le 4,\ -1 \le z \le 1\}$.

As Tabelas 6.1, 6.2, 6.3 e 6.4 mostram o valores calculados de Q_1, Q_2, Q_3 e Q_4, seus respectivos desvios padrões e número de pontos utilizados para cada um dos métodos de integração. No método MC com amostragem de importância, foi utilizada a função peso $p(x,y,z) := \frac{5}{21(e^5 - e^{-5})} e^{5z}$.

Método	QMC	QMC2	QMCI	QMCA	QMCE	QMCIA	QH	QS
Q_1 ($\times 10^2$)	2.1473	2.1837	2.1864	2.1978	2.1887	2.1620	2.1718	2.1698
e ($\times 10^2$)	0.0302	0.0200	0.0133	0.0306	0.0108	0.0151	0.0166	0.0174
n ($\times 10^4$)	5.00	11.50	5.00	5.00	6.11	5.00	5.00	5.00

Tabela 6.1: Cálculo de Q_1.

Método	QMC	QMC2	QMCI	QMCA	QMCE	QMCIA	QH	QS
Q_1 ($\times 10^2$)	5.0710	5.1446	5.1616	5.2404	5.1563	5.1347	5.1277	5.1225
e ($\times 10^2$)	0.0732	0.0500	0.0333	0.0746	0.0261	0.0356	0.0379	0.0390
n ($\times 10^4$)	5.00	10.87	5.00	5.00	5.49	5.00	5.00	5.00

Tabela 6.2: Cálculo de Q_2.

Método	QMC	QMC2	QMCI	QMCA	QMCE	QMCIA	QH	QS
Q_3 ($\times 10$)	1.8292	2.1354	2.4797	2.0550	2.1282	1.9577	2.6027	2.4079
e ($\times 10$)	0.5813	0.2000	0.3024	0.5761	0.1152	0.3019	0.1733	0.2377
n ($\times 10^4$)	5.00	43.40	5.00	5.00	23.18	5.00	5.00	5.00

Tabela 6.3: Cálculo de Q_3.

Método	QMC	QMC2	QMCI	QMCA	QMCE	QMCIA	QH	QS
Q_4 ($\times 10^2$)	1.5754	1.5898	1.5635	1.6571	1.5792	1.5535	1.5668	1.5646
e ($\times 10^2$)	0.0602	0.0100	0.0230	0.0619	0.0053	0.0122	0.0141	0.0155
n ($\times 10^4$)	5.00	35.99	5.00	5.00	18.48	5.00	5.00	5.00

Tabela 6.4: Cálculo de Q_4.

6.2 Função Gama como peso ($d = 3$).

Esse exemplo é proposto por Gentle (1998) como exercício de aplicação da técnica MC. O autor não fornece valores de referência, mas a integral pode ser resolvida analiticamente. O valor exato da integral

$$I_5 = \int_{-\pi}^{\pi}\int_0^4\int_0^{\infty} x^2 e^{-x/2} y^3 \sin(z)(\pi + z)^2(\pi - z)^3 dxdydz,$$

é $-245760\pi + 16384\pi^3 = -2.640709733... \times 10^5$ que queremos comparar com os resultados Q_5 dos diversos métodos.

A Tabela 6.5 mostra os valores calculados de Q_5. No método MC com amostragem de importância, foi utilizada a função peso $p(x,y,z) := \frac{1}{128\pi} x^2 e^{-x/2}$, proposta pelo autor do problema. Nas outras técnicas, impõe-se à variável z a condição $0 \le z \le 40$, resultando o valor aproximado de I_5 igual a $-2.640708530... \times 10^5$.

Método	QMC	QMC2	QMCI	QMCA	QMCE	QMCIA	QH	QS
$\lvert Q_5\rvert$ ($\times 10^5$)	2.7061	2.6234	2.6230	2.6602	2.6246	2.6842	2.6134	2.6477
e ($\times 10^5$)	0.1196	0.0200	0.0596	0.1149	0.0103	0.0514	0.0266	0.0294
n ($\times 10^4$)	5.00	171.42	5.00	5.00	48.93	5.00	5.00	5.00

Tabela 6.5: Cálculo de Q_5.

6.3 Integral de colisão de Boltzmann ($d = 5$).

Esse exemplo é de Morokoff & Caflisch (1995) e ilustra o uso das técnicas QMC. Os autores fornecem a solução exata dessa integral na forma de uma série. Deseja-se determinar o valor da integral

$$\frac{1}{\pi^3}\int_{\Re^3}\int_0^{2\pi}\int_0^{\pi} w \sin\chi e^{h(u,\mathrm{w},\chi,\varepsilon)} e^{-\mathrm{w}^2} d\chi d\varepsilon d\mathrm{w},$$

onde

$$h(u,\mathbf{w},\chi,\varepsilon) = -2u^2 + u(w_1 + w2 + (w_2 - w_1)\cos\chi) + u\sin\chi\left(w_{23}\sin\varepsilon + \frac{ww_3\cos\varepsilon + w_1w_2\sin\varepsilon}{w_{23}}\right),$$

e

$$\mathbf{w} = (w_1, w_2, w_3), \quad \mathbf{w} \bullet \mathbf{w} = w_1^2 + w_2^2 + w_3^2, \quad w = \sqrt{\mathbf{w} \bullet \mathbf{w}}, \quad w_{23} = \sqrt{w_2^2 + w_3^2}.$$

Através da transformação de variáveis,

$$\begin{aligned} w_i &= \mathrm{erf}^{-1}(2x_i - 1), \quad i = 1,2,3, \\ \chi &= \pi x_4, \\ \varepsilon &= 2\pi x_5, \end{aligned}$$

onde $\mathrm{erf}^{-1}(x)$ é a função inversa de

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt,$$

a integral pode ser reescrita na forma

$$I_6 = 2\sqrt{\pi}\int_V w\sin\chi e^{h(u,\mathbf{w},\chi,\varepsilon)}d\mathbf{x},$$

onde $\mathbf{x} = (x_1, x_2, x_3, x_4, x_5)$ percorre o hipercubo unitário $V_5 := [0;1]\times[0;1]\times[0;1]\times[0;1]\times[0;1]$.

Esse exemplo provém da equação de Boltzmann, que descreve a evolução da função distribuição de velocidades para um gás rarefeito. O valor exato desta integral pode ser determinado por

$$I_6 = \frac{4}{\pi}e^{-2u^2}\sum_{k=0}^{\infty} c_k u^{2k}, \quad \text{com} \quad c_k = \frac{3^{k+1} - 1}{1\times 3\times 5\times \ldots \times (2k+1)}.$$

Para o valor de $u = 0.25$, $I_6 = 2.442352953...$. A Tabela 6.6 mostra os valores calculados de Q_6. No método MC com amostragem de importância, foi utilizada a função peso $p(\chi) := \sin\chi$. Observem-se os bons resultados obtidos pelos algoritmos QH e QS.

Método	QMC	QMC2	QMCI	QMCA	QMCE	QMCIA	QH	QS
Q_6	2.4379	2.4545	2.4378	2.4432	2.4440	2.4326	2.4423	2.4428
e	0.0103	0.0200	0.0079	0.0102	0.0184	0.0079	0.0004	0.0009
n ($\times 10^4$)	5.00	1.41	5.00	5.00	1.34	5.00	5.00	5.00

Tabela 6.6: Cálculo de Q_6.

7 Conclusão.

A descrição dos métodos Monte Carlo mostrou que o problema da quadratura multidimensional é, sem dúvida, um problema não-trivial. Parece ser consenso na literatura que, para dimensões moderadamente elevadas ($d > 3$), as técnicas Monte Carlo e Quase-Monte Carlo fornecem uma abordagem mais eficiente que as técnicas de Produto Cartesiano.

Foi possível perceber que a ordem de convergência teórica do método Monte Carlo é $O(n^{-1/2})$, uma convergência lenta, sem dúvida, mas as estimativas de erro calculadas no processo são bastante confiáveis. Certamente, técnicas de redução de variância permitem aprimorar os resultados. O estudo de geradores de números pseudo-aleatórios, por sua vez, parece ter ultrapassado uma fase crítica, uma vez que existe uma boa coleção de algoritmos "com qualidade mínima assegurada".

A ordem de convergência teórica do método Quase-Monte Carlo é $O(n^{-1})$. É certo que esse método é superior ao método Monte Carlo para dimensões baixas e para funções integrandas suaves. No entanto, para dimensões mais elevadas, os resultados empíricos parecem ainda inconclusivos. A pesquisa sobre geradores de seqüências quase-aleatórias tem um papel importante nessa técnica, já que ainda não há acordo sobre qual seqüência apresenta qualidade superior.

O presente trabalho pretendeu apresentar uma visão geral dos métodos Monte Carlo, assunto de fervilhante atividade de pesquisa atual, cujos avanços são publicados em variedade e quantidade surpreendentes. Há, ainda, muito a ser abordado, apontando direções para estudos futuros. Entre esses, destacamos:

- outros algoritmos para geração de números pseudo-aleatórios, como o **Mersene twister** de Matsumoto & Nishimura (1998), cujo periodo é $2^{19937} - 1$;
- técnicas utilizadas para a validação das seqüências pseudo-aleatórias, geradas por algoritmos denominados **testes de aleatoriedade;** entre eles, o teste espectral e o teste de Kolmogorov-Smirnoff descritos em Knuth (1998) e os diversos testes estatísticos descritos em Marsaglia (1995);
- outros algoritmos para geração de seqüências de baixa discrepância, como a **seqüência de Faure generalizada,** citada em Papageorgiou & Traub (1997);
- técnicas de **redução da variação total** de f, citadas brevemente por Paskov (1996), que procuram aumentar o desempenho do método Quase-Monte Carlo.

Referências Bibliográficas

ABRAMOWITZ, M., STEGUN, I. A., *Handbook of mathematical functions*, Dover, New York, 1970.

ALLEN, M. P., TILDESLEY, D. J., *Computer simulations of liquids*, Clarendon Press, Oxford, 1987.

ANDERSON, H. L., *Metropolis, Monte Carlo and the MANIAC*, Los Alamos Science, Fall issue, pp. 96-107, 1986.

ANTONOV, I. A., SALEEV, V. M., *A economic method for computing LP_T-sequences*, USSR Computational Mathematics and Mathematical Physics, vol. 19, n. 1, pp. 252-256, 1979.

ÁVILA, G. S. S., *Introdução à análise matemática*, Edgar Blücher, São Paulo, 1993.

BABAI, L., *Monte Carlo algorithms in graph isomorphism techniques*, Research Report DMS n. 79-10, Départament de mathématiques et de statistique, Université de Montréal, (1979).

BECKMANN, P., *A history of π (pi)*, St. Martin's Griffin, New York, 1974.

BOX, G. E. P., MULLER, M. E., MARSAGLIA, G., *A note on the generation of random normal deviates*, Annals of Mathematical Statistics, vol. 29, pp. 610-611, 1958.

BRASSARD, G., BRATLEY, P., *Fundamentals of algoritmics*, Prentice Hall, Englewood Cliffs, 1996.

BRATLEY, P., FOX, B. L., *Algorithm 659: Implementing Sobol's quasirandom sequence generator*, ACM Transactions on Mathematical Software, vol. 14, pp. 88 -100, 1988.

BRAVO, E., CASTRO, A., CLAYEYSSEN, J. R., CUNHA, R. D., FACHIN, M. P. G., *Introdução ao MATLAB para Windows*, Universidade Federal do Rio Grande do Sul, Instituto de Matemática, Porto Alegre, 1995.

BRENT, R. P., *On the periods of generalizaed Fibonacci recurrences*, Mathematics of Computation, v. 63, pp. 389-401, 1994.

BUFFON, G. Comte, *Essai d'arithmétique morale*, in *Supplément à l'Histoire Naturalle*, vol. 4, L'Imprimerie Royale, Paris, 1777.

CAMARGO, M. A. O., TREVISAN, V., CLÁUDIO, D. M., *Um algoritmo algébrico para isolar zeros polinomiais complexos*, Revista de Informática Teórica e Aplicada, vol. 2, n. 2, pp. 93-113, 1995.

CLÁUDIO, D. M., MARINS, J. M., *Cálculo Numérico Computacional: Teoria e Prática*, Ed. Atlas, São Paulo, 1994.

CORBIT, D., *Numerical integration: from trapezoids to RMS*, Dr. Dobb's Journal, ed. october, pp. 117-120, 1996.

DAVIS, P. J., RABINOWITZ, P., *Methods of numerical integration*, Academic Press, New York, 1975.

DOTTO, J. O., *A useful note on Newton-Cotes Quadratures*, Revista do Centro de Ciências Exatas e Tecnologia - Universidade de Caxias do Sul, vol. 1, n. 1, pp. 61-66, 1998.

EDELMAN, A., *Eigenvalues and condition numbers of random matrices*, SIAM Journal on Matrix Analysis and Applications, vol. 9, n. 4, pp. 543-560, 1988.

EICHENAUER, J., LEHN, J., *A nonlinear congruential pseudorandom number generator,* Statistics Papers, v. 27, pp. 315-326, 1986.

FANG, K., WANG, Y., *Number theoretic methods in statistics*, Chapman & Hall, New York, 1994.

FAURE, H., *Discrépance de suites associées à un système de numération (en dimension s)*, Acta Arithmetica, vol. 41, n. 4, pp. 337-351, 1982.

FAURE, H., *On the star discrepancy of generalised Hammersley sequences in two dimensions*, Monatshefte für Mathematik, vol. 101, pp.291-300, 1986.

FAURE, H., *Using permutations to reduce discrepancy*, Journal of Computational and Applied Mathematics, vol. 31, pp. 97-103, 1990.

FERRENBERG, A. M., LANDAU, D. P., WONG, Y. J., *Monte Carlo simulations: Hidden errors from "good" random numbers generators*, Physical Review Letters, v. 69, n. 23, pp. 3382-3384, 1992.

FRÖBERG, C. E., *Introduction to numerical analysis*, Addison-Wesley, Reading, Mass., 1966.

GENTLE, J. E., *Random number generation and Monte Carlo methods*, Springer, New York, 1998.

GORI, L., MICHELLI, C. A., *On weight functions which admit explicit Gauss-Tarán quadrature formulas*, Mathematics of Computation, vol. 65, n. 216, pp. 1567-1581, 1996.

GRAHAM, R. L., KNUTH, D. E., PATASHNIK, O., *Concrete Mathematics*, Addison-Wesley, Reading, Mass., 1994.

GRASSBERGER, P., *On correlations in "good" random numbers generators,* Physics Letters A, vol. 181, n. 1, pp. 43-46, 1993.

HABER, S., *Numerical evaluation of multiple integrals*, SIAM Review, vol. 12, pp. 481-526, 1970.

HALLIDAY, D., RESNICK, R., MERRIL, J., *Fundamentos de física*, vol. 3: Eletromagnetismo, LTC Livros Técnicos e Científicos, São Paulo, 1994.

HALTON, J. H., *On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals,* Numerische Mathematik, vol. 2, n. 5, pp. 84-90, 1960. [Errata: *ibid*, vol. 2, pp. 190, 1960]

HAMMERSLEY, J. M., HANDSCOMB, D. C., *Monte Carlo Methods*, Methuen & Co, London, 1964.

HEWITT, E., STROMBERG, K, *Real and abstract analysis*, Springer-Verlag,

New York, 1965.

HICKERNELL, F. J., *A generalized discrepancy and quadrature error bound,* Mathematics of Computation, vol. 67, n. 221, pp. 299-322, 1998.

HIGHAM, N. J., *Accuracy and stability of numerical algorithms*, SIAM, Philadelphia, 1996.

HLWAKA, E., *Funktionen von beschränkter variation in der theorie der gleichverteiling*, Annali di Matematica Pura et Applicata, vol. 54, pp. 325-333, 1961.

HOARE, C. A. R., *Quicksort*, Computer Journal, vol. 5, n. 1, pp. 10-15, 1962.

HOEL, P. G., PORT, S. C., STONE, C. J., *Introduction to probability theory*, Houghton Mifflin, Boston, 1971.

IEEE *Standard for binary floating-point arithmetic*, ANSI/IEEE Standard 754-1985, Institute of Eletrical and Electronics Engineers, New York, (1985).

KNUTH, D. E., *The art of computer programming, vol. 2: Seminumerical algorithms*, Addison-Wesley, Reading, Mass., 1998.

KOCIS, L., WHITEN, W. J., *Computational investigations of low-discrepancy sequences*, ACM Transactions on Mathematical Software, vol. 23, n. 2, pp. 266-294, 1997.

LAPLACE, Marquis P. S., *Theorie analytique des probabilités*, in *Oeuvres Complétes de Laplace*, vol. 7, Part 2, pp. 365-366, L'Académie des Sciences, Paris, 1886.

L'ECUYER, P., BLOUIN, F., COUTURE, R., *A search for good multiple recursive random number generators*, ACM transactions on Modeling and Computer Simulation, vol. 3, n. ?, pp. 87-98, 1993.

L'ECUYER, P., *Maximally equidistributed combined Tausworthe generators*, Mathematics of Computation, vol. 65, n. 213, pp. 203-213, 1996.

L'ECUYER, P., *Tables of linear congruential generators of different sizes and good lattice structure,* Mathematics of Computation, vol. 68, pp. 249-260, 1999.

LEHMER, D. H., *Mathematical methods in large-scale computing units*, Proceedings of the Second Symposium on Large Scale Digital Computing Machinery, pp. 141-146, Harvard University Press, Cambridge, 1951.

LEWIS, T. G., PAYNE, W. H., *Generalized feedback shift register pseudorandom number algoritms*, Journal of ACM, vol. 20, pp. 456-468, 1973.

LICÍNIO, P., LEROTIC, M., DANTAS, M. S. S., *Difração por redes desordenadas e o efeito de Debye-Waller*, Revista Brasileira de Ensino de Física, vol. 20, n. 3, pp. 206-213, 1998.

MARSAGLIA G., *Random numbers fall mainly in planes,* Procedings of the National Academy of Sciences, vol. 61, pp. 25-28, 1968.

MARSAGLIA G., *A current view of random numbers generator,* Procedings of

the XVIth Symposium on the Interface, pp. 3-10, 1985.

MARSAGLIA G., *The Marsaglia random number CDROM, including DIEHARD battery of test of randomness*, Department of Statistics, Florida State University, Tallahassee, Florida, 1995. [Disponível em: http://stat.fsu.edu/pub/diehard/cdrom]

MASCAGNI, M., CUCCARO, S. A., PRYOR, D. V., ROBINSON, M. L., *A fast, high quality, and reproducible parallel lagged-Fibonacci pseudorandom number generator*, Journal of Computational Physics, v. 119, pp. 211-219, 1995.

MATHEWS, J. H., *Numerical methods for mathematics, science and engineering*, Prentice Hall, Englewood Cliffs, 1992.

MATSUMOTO, M., NISHIMURA, T., *Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator*, ACM Transactions on Modeling and Computer Simulation, vol. 8, n. 1, pp. 3-30, 1998.

METROPOLIS, N., ULAM, S., *The Monte Carlo method*, Journal of American Statistical Association, vol. 44, n. 247, pp. 335-341, 1949.

METROPOLIS, N., ROSENBLUTH, A. W., ROSENBLUTH, M. N., TELLER, A. H., TELLER, E., *Equation of state calculations by fast computing machines*, Journal of Chemical Physics, vol. 21, pp. 1087-1092, 1953.

METROPOLIS, N., *The beginning of the Monte Carlo Method*, Los Alamos Science, Special issue, pp. 125-130, 1987.

MOHAZZABI. P., *Monte Carlo estimations of e*, American Journal of Physics, vol. 66, n. 2, pp. 138-140, 1998.

MOROKOFF, W. J., CAFLISCH, R. E., *Quasi-Monte Carlo Integration*, Journal of Computational Physics, vol. 122, pp. 218-230, 1995.

MOTWANI, R., RAGHAVAN, P., *Randomized algorithms*, Cambridge University Press, London, 1995.

NIEDERREITER, H., *On a new class of pseudorandom numbers for simulation methods*, Journal of Computational and Applied Mathematics, v. 56, pp. 159-167, 1994.

OLSON, D., BROZOVICH, C., CARR, J., HATTON, H., MILES Jr., G., ZWICKE, G., *Monte Carlo computer simulation of a rainbow,* The Physics Teacher, v. 28, n. 4, pp.226-227, 1990.

PAPAGEORGIOU, A., TRAUB, J. F., *Faster evaluation of multidimensional integrals*, Computer in Physics, vol. 11, n. 6, pp. 574-578, 1997.

PARK, S. K., MILLER, K. W., *Random numbers generators: Good ones are hard to find*, Communications of the ACM, vol. 31, pp. 1192-1201, 1988.

PASKOV, S. H., *New methodologies for valuing derivatives*, in PLISKA, S., DEMPSTER, M. (ed.), *Mathematics of derivative securitics*, Cambridge University Pres, Cambridge, 1996.

PLLANA, S., *History of Monte Carlo Method*, Disponível em http://stud2.tuwien.ac.at/~e9527412/ history. html, 28/09/1999.

PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., FLANNERY, B., P., *Numerical recipes in Fortran 77*, vol. 1, Cambridge University Press, London, 1992.

RABIN, M. O., *Probabilistic algorithm for primality testing*, Journal of Number Theory, vol. 12, pp.128-138, 1980.

RAND CORPORATION, *A million random digits with 100,000 normal deviates*, The Free Press, Glencoe, 1955.

RIVEST, R. L., SHAMIR, A., ADLEMAN, L. M., *A method for obtaining digital signatures and public-key cryptosystems*, Communications of the ACM, vol. 21, n. 2, pp. 120 -126, 1973.

ROTH, K. F., *On irregularities of distributions*, Mathematika, vol. 1, pp. 73-79, 1954.

RUCKDESCHEL, F. R., *Basic scientific subroutines*, vol. 1, Byte / McGraww-Hill, Peterborough, 1981.

SCHEID, F., *Análise Numérica*, 2. ed., McGraw-Hill, Lisboa, 1991.

SCHILDT, H., *C completo e total*, 3. ed., Makron Books, São Paulo, 1996.

SCHLESINGER, M. F., WEST, B. J., (ed.), *Random walks and their applications in the physical and biological sciences*, American Institute of Physics, 1984.

SMITH, J., *Design and analysis of algorithms*, PWS-Kent, Boston, 1989.

SOBOL, I. M., *On the distribution of points in a cube and the approximate evaluation of integrals*, USSR Computational Mathematics and Mathematical Physics (english translation), vol. 7, n. 4, pp. 86-112, 1967.

SOBOL, I. M., *O método de Monte Carlo*, Mir, Moscou, 1983.

STAHNKE, W., *Primitive binary polynomials*, Mathematics of Computation, vol. 27, n. 124, pp. 977-980, 1973.

STUDENT, *On the probable error of a mean*, Biometrika, vol. 6, pp. 1-25, 1908 (a).

STUDENT, *Probable error of a correlation coefficient*, Biometrika, vol. 6, pp. 302-310, 1908 (b).

TAUSWORTHE, R. C., *Random numbers generated by linear recurrence modulo two*, Mathematics of Computation, vol. 19, pp. 201-209, 1965.

THISTED, R., *Elements of statistical computing*, Chapman & Hall, London, 1996.

TRIOLA, M. F., *Introdução à estatística*, Livros Técnicos e Científicos, Rio de Janeiro, 1999.

ULAM, S., RICHTMAYER, R. D., von NEUMANN, J., *Statistical methods in neutron diffusion*, Los Alamos Scientific Laboratory Report LAMS-551, 1947.

VUOLO, J. H., *Fundamentos da teoria de erros*, 2. ed, Edgard Blücher, São Paulo, 1996.

WAGNER, M., CEPERLEY, D. M., *Path integral Monte Carlo simulations of the melting of molecular Hydrogen surfaces*, Journal of Low Temperature Physics, vol. 102, n. 274, 1996.

WICHMANN, B., HILL, D., *Building a random-number generator*, Byte, pp. 127-128, march, 1987.

Apêndice A

O método Monte Carlo em simulação.

Neste Apêndice, descrevem-se alguns exemplos do uso do método Monte Carlo na resolução de problemas de simulação práticos. Esses exemplos são descritos de forma simplificada, porém servem para proporcionar uma idéia mais clara do tipo de análise envolvida. Os exemplos mostrados são clássicos, sendo citados por diversos autores, entre eles Sobol (1983) e Schlesinger & West (1984).

A.1 Simulação de sistemas de atendimento.

A.1.1 Descrição do problema.

Considere-se um sistema de atendimento constituído de n canais. Cada canal pode ou não atender a um pedido que ingressa no sistema no tempo T_I, dito **tempo de ingresso**. Se um pedido ingressa no sistema e existe algum canal disponível então esse pedido é atendido durante um certo tempo T_A, dito **tempo de atendimento**. Se um pedido entra no sistema e não há nenhum canal disponível pois os n canais estão atendendo, então o pedido é rejeitado. O problema consiste em determinar quantos pedidos, em média, serão atendidos ou não após um certo tempo T, dito **tempo de amostragem**.

O modelo lógico descrito acima pode representar os mais diversos sistemas reais. Pode representar, por exemplo, uma central de comutação telefônica com a capacidade de processar um número limitado de chamadas. Se uma chamada chega ao sistema e não existem linhas disponíveis, a ligação é perdida. Problemas desse tipo são muito comuns e apenas uma pequena parcela deles pode ser resolvida analiticamente. Portanto a simulação pelo método MC poder ser de grande utilidade.

Na análise desse problema, um fator decisivo é a forma como os valores do tempo de ingresso de pedidos T_I e da duração dos atendimentos T_A distribuem-se. Esses valores são aleatórios, sujeitos a determinadas FDP. Usualmente admite-se que esses valores tenham distribuição Exponencial ou de Poisson. Nesse exemplo consideram-se T_I e T_A com distribuição Exponencial e com tempos médios $T_{IM} := \langle T_I \rangle$ e $T_{AM} := \langle T_A \rangle$.

O algoritmo seguinte simula esse sistema de atendimento: n é o número de canais de atendimento, T_{IM} é o tempo médio de ingresso de pedidos, T_{AM} é o tempo médio de atendimento, T é o tempo total de amostragem, n_A é o número de pedidos atendidos, n_T é o número total de pedidos, t armazena o tempo transcorrido e o vetor $\mathbf{c}$ de n

componentes armazena o instante em que cada canal estará desocupado. Inicialmente os instantes $c_1, c_2, ..., c_n$ são todos inicializados com 0, indicando que todos os canais estão livres. Para $0 \leq t \leq T$, em um laço de repetição, um pedido é criado gerando-se aleatoriamente um tempo T_I e fazendo $t := t + T_I$. Se algum canal i está livre ($t > c_i, 1 \leq i \leq n$), então um tempo de atendimento T_A é gerado deixando o canal ocupado até o instante $c_i := t + T_A$. Incrementa-se o contador de pedidos atendidos. Caso contrário o pedido não é atendido e perde-se. Repete-se o laço temporal.

Algoritmo A.1: Simulador de Sistema de Atendimento (SSA).

```
(n_A, n_T) ← SSA(n, T_IM, T_AM, T)
{inicializa contadores}
n_T ← 0                          {número total de pedidos}
n_A ← 0                          {número de pedidos atendidos}
para i ← 1 : n
    c_i ← 0                      {instante em que o canal será liberado}
fim
t ← 0                            {tempo}
enquanto t < T
    t ← t + DExp(T_IM)           {instante do pedido. distribuição Exponencial}
    n_T ← n_T + 1                {incrementa número de pedidos}
    para i ← 1 : n               {para todos os canais...}
    se t > c_i                   {se o canal está livre...}
        c_i ← t + DExp(T_AM)     {instante de liberação do canal}
        n_A ← n_A + 1            {conta atendimentos}
        break                    {encerra laço}
    fim
fim
```

A.1.2 Exemplo de simulação.

Seja um sistema de 3 canais, cujo fluxo de pedidos tenha intervalo médio de tempo de 1 min. Supõe-se que cada atendimento demore, em média, 2 min. Quantos pedidos serão feitos, em média, durante um tempo de amostragem de 1000 min? Quantos pedidos, em média, serão atendidos?

Para simular esse sistema com essas condições, executa-se o algoritmo acima com os parâmetros $n = 3$, $T_{IM} = 1$, $T_{AM} = 2$, $T = 1000$, obtendo-se $n_A = 787$ e $n_T = 991$, o que significa que, nessa simulação, foram feitos 991 pedidos e desses, 787 foram atendidos. Se a rotina é executada novamente com semente de simulação diferente obtém-se outro resultado. De fato, pode-se determinar valores médios e desvios para n_A e n_T, executando-se a rotina um certo número de vezes, 20, diga-se. De fato, obtém-se $n_A = 794 \pm 21$ e $n_T = 1010 \pm 35$.

Qual o impacto da diminuição do tempo de atendimento de 2.0 min para 1.5 min sobre a quantidade de pedidos atendidos? -Repetindo o procedimento acima com os parâmetros $n = 3$, $T_{IM} = 1$, $T_{AM} = 1.5$, $T = 1000$, obtém-se $n_A = 869 \pm 20$ e $n_T = 1003 \pm 25$, que significa um aumento médio de 75 atendimentos.

Não seria melhor aumentar o número de canais? - Repetindo-se o procedimento acima com os parâmetros $n = 4$, $T_{IM} = 1$, $T_{AM} = 2$, $T = 1000$, obtém-se $n_A = 907 \pm 22$ e $n_T = 1001 \pm 31$, que significa um aumento médio de 113 atendimentos. De fato, é maior o impacto de aumentar o número de canais.

A.1.3 Considerações sobre a complexidade do problema.

No exemplo acima considerou-se um sistema extremamente simples. Muitas outras considerações podem ser feitas para tornar a simulação mais próxima da realidade. Por exemplo, é usual admitir que os canais não são idênticos, tendo tempos de atendimento diferenciados. Pode-se considerar que os canais sofrem avarias, ficando inoperantes por algum tempo. Também pode-se considerar que o fluxo de pedidos não é regular, mas tenha períodos de maior e menor intensidade de fluxo.

Outra característica importante a se considerar seria a capacidade de espera de pedidos. Em um modelo desse tipo, se todos os canais estivessem ocupados, um pedido não seria rejeitado mas esperaria em uma fila. Seria possível simular o tempo de espera de um pedido, o tamanho da fila, etc.

A.2 Simulação de propriedades de sistemas de muitos elementos.

A.2.1 Descrição do problema.

Seja um sistema S constituído de um número m, eventualmente grande, de elementos s_1, s_2, ..., s_m e G, uma certa propriedade desse sistema, determinada em termos de seus elementos como uma função conhecida

$$G = F(s_1, s_2, ..., s_m)$$

Admita-se que s_1, s_2, ..., s_m são variaveis aleatórias com FDP conhecidas $f_1(s_1)$, $f_2(s_2)$, ..., $f_m(s_m)$. Nessa situação, duas questões apresentam-se:

1. Se os elementos de S apresentam valores médios $\langle s_1 \rangle, \langle s_2 \rangle, ..., \langle s_m \rangle$, o valor médio $\langle G \rangle$ será dado por $\langle G \rangle = F(\langle s_1 \rangle, \langle s_2 \rangle, ..., \langle s_m \rangle)$?
2. Os valores de G terão que distribuição?

A resposta para a primeira pergunta é: não. De um modo geral aquela igualdade não é verificada, a não ser que G seja função linear nos elementos de S. A segunda questão é muito mais complicada. A FDP de G não se expressa de forma analítica, a não ser nos casos mais simples.

Com o intuito de avaliar os limites de G, pode-se tomar valores de $s_1, s_2, ..., s_m$ tais que o valor de G seja maximizado ou minimizado. No entanto, tal procedimento pode não ser simples, dependendo da forma de F ou da quantidade de elementos. Mesmo obtendo esses limites, sua ocorrência seria muito pouco provável, sendo de pouca utilidade na obtenção de limites razoáveis para G.

O sistema aqui exposto pode representar, por exemplo, um equipamento eletrônico (um televisor, por exemplo) produzido em grande quantidade, composto de uma certa quantidade de elementos (transistores, capacitores, resistores, indutores, fontes, diodos, ...). Uma certa propriedade desse equipamento (o consumo de energia, por exemplo) pode ser descrita por uma função dos valores dos elementos do equipamento. Embora cada elemento constituinte desse equipamento tenha sido concebido para ter um valor bem definido, dito **valor nominal**, o valor verdadeiro desse componente pode diferir levemente de equipamento para equipamento. Os valores verdadeiros têm distribuição Normal em torno do valor nominal. O desvio padrão desses valores é denominado **tolerância**. Essa distribuição de valores faz com que a propriedade G seja diferente de um exemplar a outro.

Esse problema de simulação pode ser resolvido pelo método MC, se a FDP de cada elemento do sistema for conhecida. O procedimento de resolução é o seguinte:

1. simula-se um conjunto de valores s_1, s_2, ..., s_m para os elementos de S; esses valores são simulados de acordo com a FDP de cada elemento;
2. a propriedade G é calculada;
3. repetem-se os passos 1 e 2 até obter uma quantidade suficientemente grande de valores de G para proceder-se a sua análise.

Evidentemente o grau de exatidão da análise do comportamento da propriedade G dependerá fortemente do grau de conhecimento do comportamento individual de cada elemento do sistema, expresso por sua FDP e, ainda, da fidedignidade de sua relação expressa pela função F.

Para exemplificar o método, considere-se o circuito elétrico mostrado na Figura A.1. Esse circuito é composto de 7 resistores $R_1, R_2, ..., R_7$ e uma fonte de tensão ε.

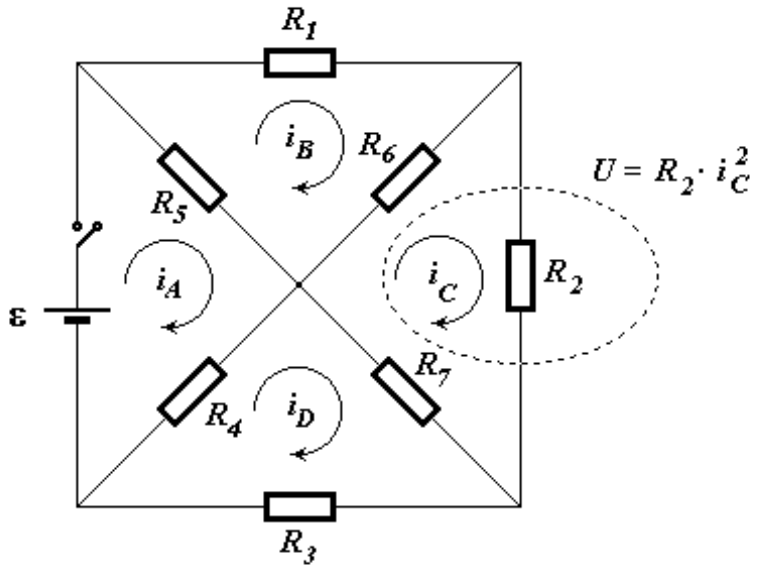

Figura A.1: Um sistema S (circuito elétrico) com 8 elementos (7 resistores e uma fonte).

Os valores nominais e tolerâncias dos componentes do circuito acima são mostrados na Tabela A.1. A tensão é dada em volts (V) e as resistências em ohms(Ω). A tolerância no valor de cada componente do circuito é 20% do seu valor nominal, uma tolerância usual.

Componente	Valor nominal	Tolerância
R_1	100	20
R_2	100	20
R_3	100	20
R_4	50	10
R_5	50	10
R_6	50	10
R_7	50	10
ε	15	3

Tabela A.1: Valor nominal dos componentes do circuito.

A propriedade G em questão será a potência elétrica U dissipada em R_2, dada por

$$U = R_2 i_C^2, \tag{A.1}$$

onde R_2 é o valor da resistência elétrica do componente e i_C o valor da corrente elétrica que o atravessa.

As correntes elétricas i_A, i_B, i_C, I_D utilizadas na resolução do circuito são ditas **correntes de malha**, e não representam necessariamente a corrente em cada componente. O método de resolução desse circuito pode ser visto em livros de eletricidade básica (Halliday *et alii*, 1994). O valor da corrente i_C é obtido através da resolução do seguinte sistema de equações:

$$\begin{bmatrix} R_4 + R_5 & -R_5 & 0 & -R_4 \\ -R_5 & R_1 + R_5 + R_6 & -R_6 & 0 \\ 0 & -R_6 & R_2 + R_6 + R_7 & -R_7 \\ -R_4 & 0 & -R_7 & R_3 + R_4 + R_7 \end{bmatrix} \begin{bmatrix} i_A \\ i_B \\ i_C \\ i_D \end{bmatrix} = \begin{bmatrix} \varepsilon \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{A.2}$$

Usando os valores nominais para os elementos do circuito obtêm-se, resolvendo (A.2) e (A.1), os valores nominais $i_C = 0.03\,\mathrm{A}$ e $U_n = 0.09$ W.

Conhecendo a distribuição dos valores verdadeiros dos componentes do circuito, pode-se querer responder às seguintes perguntas:

1. O valor médio $\langle U \rangle$ da propriedade U, é igual ao seu valor nominal U_n?
2. Os valores dos componentes têm distribuição Normal? U tem distribuição Normal?
3. O desvio padrão dos componentes são todos iguais a 20% do valor nominal; Pode-se dizer o mesmo de U?

O algoritmo seguinte simula o valor para a propriedade global U do circuito elétrico mostrado na Figura A.1. Os vetores $\mathbf{C}$ e $\mathbb{D}$ contém os valores nominais e o desvio padrão dos elementos do circuito. U é a potência dissipada em R_2.

Algoritmo A.2: Simulador de Circuito Elétrico (SCE).
$U \leftarrow SCE(C, D)$
para $i \leftarrow 1 : 7$
    $R_i \leftarrow DNormal(C_i, D_i)$ {distribuição Normal, algoritmo 3.8}
fim
$\varepsilon \leftarrow DNormal(C_8, D_8)$

$$A \leftarrow \begin{bmatrix} R_4+R_5 & -R_5 & 0 & -R_4 \\ -R_5 & R_1+R_5+R_6 & -R_6 & 0 \\ 0 & -R_6 & R_2+R_6+R_7 & -R_7 \\ -R_4 & 0 & -R_7 & R_3+R_4+R_7 \end{bmatrix}$$

$$B \leftarrow \begin{bmatrix} \varepsilon \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$I \leftarrow A^{-1}B$$

$$U \leftarrow R_2 I_3^2$$

A.2.2 Exemplo de simulação.

Em uma típica execução com 5000 simulações do circuito, obtêm-se valores de U (em watts) compreendidos no amplo intervalo $[0.0131; 0.4832]$. Os valores de U distribuem-se conforme mostra o histograma de freqüência da Figura A.2.

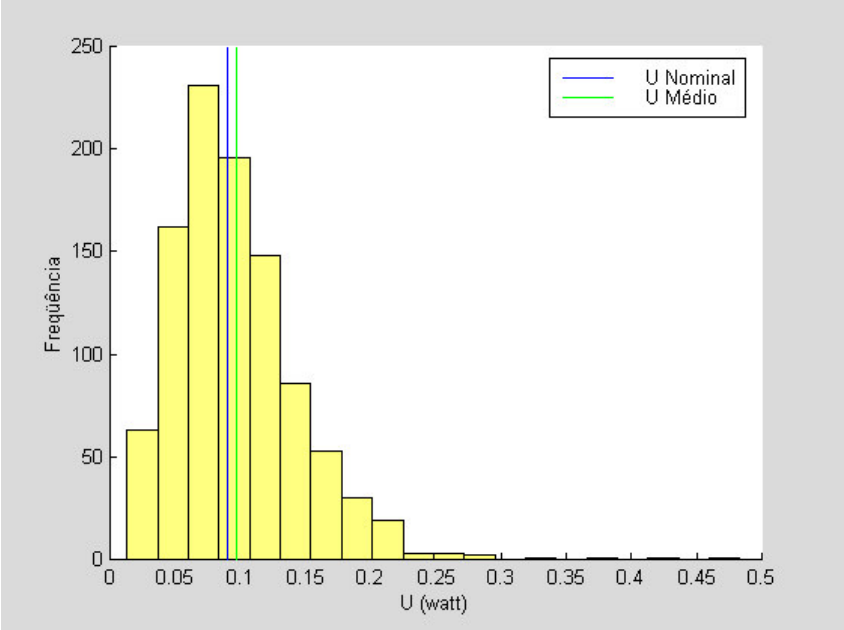

Figura A.2: Distribuição da propriedade global U para 5000 simulações do circuito.

Observando atentamente os dados obtidos e o gráfico, pode-se responder às perguntas levantadas acima. Em primeiro lugar verifica-se que o valor médio dos valores simulados $\langle U \rangle = 0.0968$ é 7.52% maior que o valor nominal U_n.

Em seguida verifica-se que, embora todos os componentes do circuito tivessem

distribuição Normal, os valores de U não são distribuídos dessa forma. A cauda do histograma apresenta claramente um deslocamento à direita, isto é, uma tendência para altos valores (assimetria positiva). Vale notar que esse deslocamento é acentuado pelo fator quadrático i_C^2 de U. Porém, a própria variável i_C apresenta um pequeno deslocamento à esquerda.

Finalmente verifica-se que os valores gerados de U apresentam um desvio padrão de $0,0491$ W, isto é, um alargamento dos valores de U em torno de 54.58% em relação à média. Esse desvio está muito acima dos 20% do desvio dos componentes do circuito.

Também pode-se querer determinar a influência da diminuição da tolerância dos valores de cada componente no valor da propriedade U, isto é, como melhorar o comportamento dessa propriedade substituindo cada tipo de componente por outro de melhor qualidade. A Tabela A.2 mostra o valor de $\langle U \rangle$, o desvio padrão $\Delta\langle U \rangle$ (em watts e em %) desses valor em função do componente substituído, componentes de tolerância 20% por componentes de tolerância de 10%. Observando a tabela, verifica-se que, utilizando resistores R_1, R_2, R_3 de menor tolerância, obtém-se um valor médio $\langle U \rangle$ mais próximo do seu valor nominal $U = 0.0900\,\mathrm{W}$; no entanto uma fonte ε melhor diminui o desvio padrão $\Delta\langle U \rangle$ desses valores.

Componente Substituído	$\langle U \rangle$ (W)	$\Delta\langle U \rangle$ (W)	$\Delta\langle U \rangle$ (%)
Resistores R_1, R_2, R_3	0.0928	0.0395	42.6
Resistores R_4, R_5, R_6, R_7	0.0952	0.0457	48.0
Fonte ε	0.0957	0.0330	34.5

Tabela A.2: A influência da substituição de componentes no desempenho da propriedade U.

A.2.3 Considerações sobre a complexidade do problema.

No exemplo dessa seção considerou-se um sistema extremamente simples. Muitas outras considerações podem ser feitas para melhorar essas simulações. Por exemplo, nesse modelo consideraram-se todos os componentes com distribuição Normal; no entanto outros tipos de distribuições podem ser consideradas.

Nesse exemplo, considerou-se apenas uma propriedade U. No entanto, o refinamento do modelo pode levar à avaliação de muitas outras propriedades do sistema.

Esse tipo de análise pode ser usado para estimar limites de confiabilidade do sistema. Por exemplo, é possível simular a quantidade de circuitos que apresentam propriedades dentro de certos limites desejáveis ou não.

A.3 Passeio Aleatório Discreto.

Uma das aplicações do método Monte Carlo consiste em simular o processo denominado **passeio aleatório**, um dos mais simples exemplos de processo estocástico (Schlesinger & West, 1984). Considere-se uma partícula que ocupe posições a distâncias $0, \pm 1, \pm 2, \pm 3, ...$, a partir de uma origem. Suponha-se que a cada instante a partícula seja movida um passo à direita ou à esquerda com probabilidades p e $q := 1 - p$, respectivamente. A seqüência de posições da partícula é chamada de passeio aleatório.

O passeio aleatório pode ser usado como uma aproximação discreta do movimento errático de partículas microscópicas, denominado **movimento Browniano**. Esse movimento foi notado pela primeira vez em 1827 pelo botânico inglês Robert Brown (1773-1858), que observou uma suspensão de partículas microscópicas de grãos de pólen.

O passeio aleatório também é usado como modelo de situações de risco: muitos modelos econômicos admitem que o mercado de ações tem comportamento estocástico. A Figura A.3 mostra o preço das ações da Texaco inc. na bolsa de valores de Wall Street de agosto de 1998 a janeiro de 1999. As barras vermelhas representam a variação diária. A linha azul, o valor médio dos últimos 50 dias. Observe-se que o valor da ação parece aumentar ou diminuir de modo aleatório. Alguns modelos computacionais usam o método MC para simular esse comportamento.

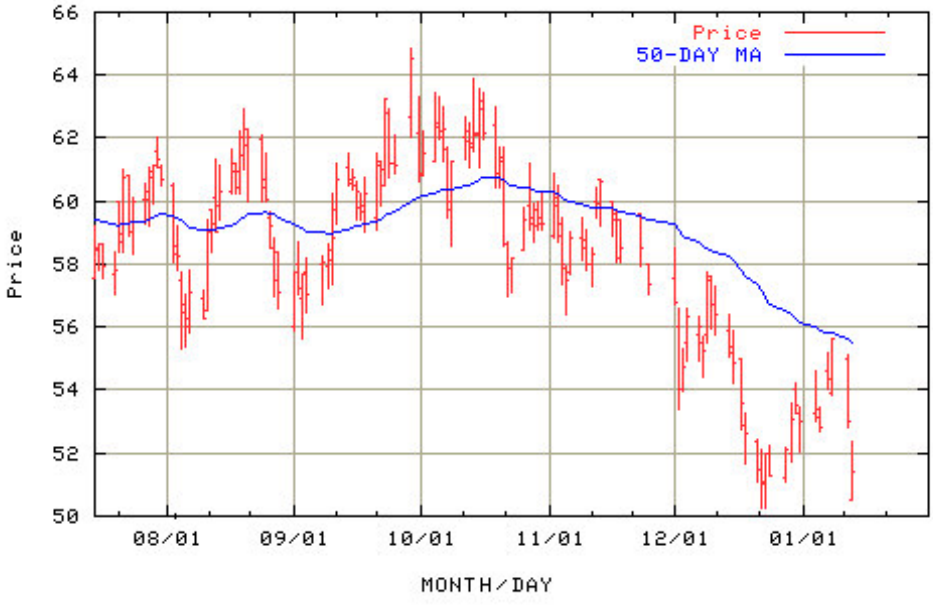

Figura A.3: Preço das ações da Texaco inc. na bolsa de valores

de Wall Street. Fonte: www.wallstreetalert.com.

O passeio aleatório pode ser discreto ou contínuo, unidimensional ou multidimensional. Nessa seção aborda-se o casos discreto e unidimensional, na seção seguinte o caso contínuo e tridimensional.

A.3.1 Descrição do problema.

Um problema clássico que apresenta muitas considerações interessantes é denominado **problema da ruína**. Suponha-se que jogadores iniciem um jogo de apostas, cada um com uma fortuna para aposta. O jogo consiste em uma seqüência de apostas, onde cada jogador participa com uma unidade monetária. Em cada aposta, cada primeiro jogador tem uma probabilidade de vencer. A fortuna de cada jogador é uma variável discreta. O problema consiste em prever a probabilidade com que cada jogador perde toda sua fortuna, a ruína.

O algoritmo seguinte simula um jogo de apostas de muitos jogadores. Os jogadores começam com fortunas iniciais armazenadas no vetor **F**. A cada rodada, os jogadores ganham as apostas com probabilidades constantes, dadas pelo vetor $\mathbb{P}$. O jogo termina após n rodadas ou quando ocorre a ruína de algum jogador. **H** é uma matriz em que cada coluna representa o histórico da fortuna de cada jogador.

Algoritmo A.3: Simulador de Ruína de Jogador (SRJ).

$H \leftarrow SRJ(\mathbf{F}, \mathbb{P}, n)$
$M \leftarrow$ tamanho de P {número de jogadores}
para $j \leftarrow 1 : M$
    $H_{1,j} \leftarrow F_j$ {fortuna inicial}
fim
$i \leftarrow 1$
enquanto $i \leq n$ e $\min(F) > 0$ {enquanto nenhuma ruína...}
    $i \leftarrow i + 1$ {próxima rodada}
    para $j \leftarrow 1 : M$
        $F_j \leftarrow F_j - 1$ {aposta}
    fim
    $j \leftarrow DDG(P)$ {vencedor}
    $F_j \leftarrow F_j + M$ {prêmio}
    para $j \leftarrow 1 : M$
        $H_{1,j} \leftarrow F_j$ {histórico}
    fim
fim

A.3.2 Exemplo de simulação.

A Figura A.4 mostra o resultado da simulação de um jogo de apostas entre 3 jogadores A, B e C. Inicialmente os jogadores possuem as fortunas de $F_A = 100$, $F_B = 150$ e $F_C = 200$ unidades. As probabilidade de vitória dos jogadores são $P_A = 0.4$, $P_B = 0.4$ e $P_C = 0.2$. Como era de esperar, os jogadores A e B incrementam suas fortunas aproximadamente à mesma taxa às custas do jogador C.

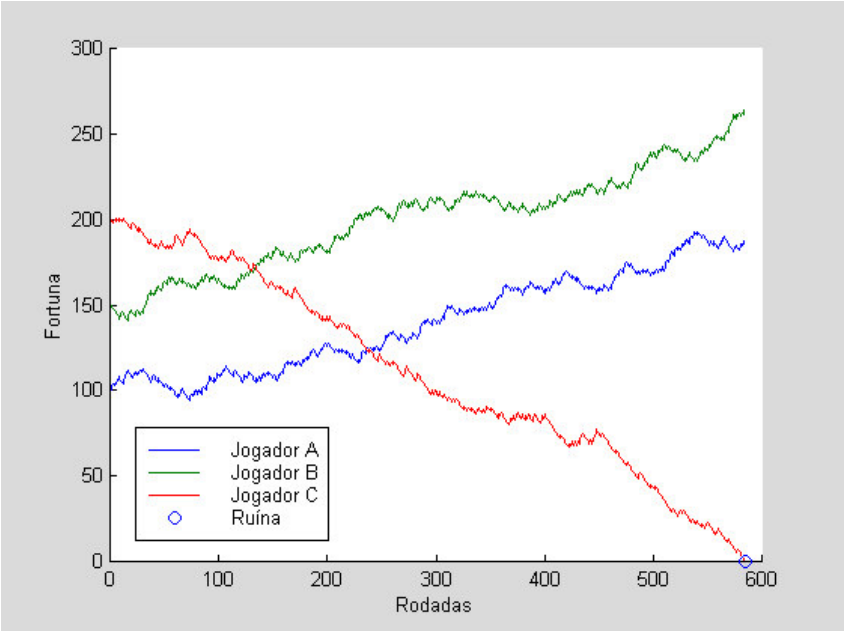

Figura A.4: Evolução da fortuna de três jogadores.

A.3.3 Considerações sobre a complexidade do problema.

No caso de apenas dois jogadores com probabilidade de vitória p_A e p_B, respectivamente, e fortunas iniciais F_A e F_B, respectivamente, a probabilidade de que o primeiro jogador perca toda sua fortuna pode ser calculada. Se $p_A = p_B = \frac{1}{2}$, a probabilidade é $F_A/(F_A + F_B)$. O problema torna-se mais interessante, e mais complicado, se o número de jogadores for maior ou se a probabilidade de vitória de cada jogador não for constante, mas dependente de fatores como a própria fortuna, o tempo de jogo, o número de jogadores, etc. Pode-se, ainda, escolher estratégias de aposta diversificadas como apostas não-unitárias, etc.

A.4 Passeio aleatório contínuo.

Como exemplo de simulação de um passeio aleatório contínuo, simula-se a difusão de nêutrons através de uma parede de contenção. Trata-se de um exemplo clássico do método MC e, historicamente, um dos seus primeiros resultados práticos (Ulam *et alii*, 1947).

A.4.1 Descrição do problema.

Numa sumária colocação do problema, seja um feixe de n nêutrons incidindo perpendicularmente sobre uma parede infinita e homogênea de espessura H. Ao penetrar na parede, cada nêutron percorre uma certa distância λ, antes de sofrer uma colisão com os átomos constituintes da parede. Essa distância tem FDP Exponencial com valor médio L, dito **caminho livre médio**. A cada colisão, o nêutron pode ser absorvido pelo átomo da parede ou desviado em uma nova direção. As possíveis trajetórias de um nêutron são mostradas na Figura A.5. Após um certo número de colisões, o nêutron ser absorvido por um átomo da parede, sair do interior da parede para o lado externo ou interno. Nesses casos diz-se que houve uma **absorção** (A), **fuga** (F) ou **confinamento** (C) do nêutron, respectivamente.

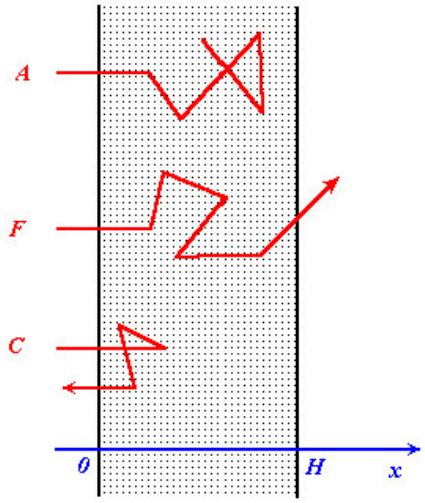

Figura A.5: As possíveis trajetórias de um nêutron. Adaptado de Sobol (1983).

A interação dos nêutrons com os átomos da parede caracteriza-se pela combinação de duas variáveis S_E e S_A, ditas, respectivamente, **seção de desvio** e **seção de absorção**. A soma dessas variáveis,

$$S := S_E + S_A,$$

é dita **seção de choque eficaz**. O caminho livre médio L, percorrido pelo nêutron, está relacionado com S por

$$L = 1/S.$$

Assim, o deslocamento λ é variável aleatória com FDP f, dada por

$$f(t) := \frac{1}{L} e^{-t/L},$$

que, pelo método de inversão (subseção 3.2.1), pode-se simular por

$$\lambda := -L \ln U_1, \tag{A.3}$$

onde U_1 é variável aleatória uniformemente distribuída no intervalo $[0; 1)$.

Em uma colisão, as probabilidades p_A de ocorrer uma absorção e p_E de ocorrer um desvio são dadas por

$$p_A := \frac{S_A}{S} \quad \text{e} \quad p_E := \frac{S_E}{S}.$$

A questão que se propõe resolver é determinar, em função de S_E, S_A e H, que fração dos n nêutrons incidentes é absorvida pela parede, que fração a atravessa e que fração é refletida.

Após uma colisão, a direção de desvio é caracterizada pelo ângulo φ em relação ao eixo dos x e o ângulo θ de rotação em torno do eixo dos x, sendo $0 \leq \varphi \leq \pi$ e $0 \leq \theta \leq 2\pi$. Nesse problema, a parede é infinita e de espessura constante H, portanto somente a componente de l paralela ao eixo dos x é relevante, de modo que pode-se representar as sucessivas posições aleatórias X da partícula por

$$X := \lambda \cos(\varphi), \tag{A.4}$$

onde φ é uma variável aleatória, cuja distribuição determinar-se-á abaixo.

Supondo que a direção do desvio esteja distribuída uniformemente no espaço, a probabilidade $p(\varphi, \theta)$ de que ela esteja contida em um elemento de angulo esférico $d\Omega$ satisfaz

$$p(\varphi, \theta) d\varphi d\theta = \frac{d\Omega}{4\pi}.$$

Mas sendo $d\Omega = \operatorname{sen}(\varphi) d\varphi d\theta$, tem-se

$$p(\varphi,\theta) = \frac{\operatorname{sen}(\varphi)}{4\pi}.$$

Conhecendo a FDP conjunta $p(\varphi,\theta)$, pode-se obter a FDP individual $p(\varphi)$ por

$$p(\varphi) = \int_0^{2\pi} p(\varphi,\theta)d\theta = \frac{\operatorname{sen}(\varphi)}{2}.$$

A simulação de φ pode ser feita pelo método de inversão através da variável U_2 de distribuição uniforme no intervalo $[0;1)$, mediante

$$U_2 = \int_0^{\varphi} \frac{\operatorname{sen}(t)}{2}dt,$$

do que resulta

$$\cos(\varphi) = 1 - 2U_2. \tag{A.5}$$

Desse último resultado pode-se concluir que $\cos(\varphi)$ é uniformemente distribuído no intervalo $[-1;1)$. Assim, combinando (A.3), (A.4) e (A.5), obtém-se a fórmula para simulação de X,

$$X := (-L\ln U_1)(1 - 2U_2)$$

No algoritmo abaixo, S_A e S_E são, respectivamente, as seções de absorção e desvio, H é a espessura da parede e n é o número de nêutrons incidentes. Os valores n_C, n_A n_F são as quantidades de nêutrons que sofreram, respectivamente, confinamento, absorção ou fuga.

Algoritmo A.4: Simulação da passagem de Nêutrons através de uma Parede (SNP).

$(n_C, n_A, n_F) \leftarrow SNP(S_A, S_E, H, n)$
{inicialização}
$n_C \leftarrow 0$ {contadores de partículas}
$n_A \leftarrow 0$
$n_F \leftarrow 0$
$S \leftarrow S_A + S_E$ {seção de choque efetiva}
$P_A \leftarrow S_A/S$ {probabilidade de absorção}
$P_E \leftarrow S_E/S$ {probabilidade de desvio}
$L \leftarrow 1/S$ {caminho livre médio}
{bombardeamento de partículas}
para $i \leftarrow 1 : n$ {para cada partícula...}
    $x = -L\ln(GU)$ {posição inicial. GU: Gerador Uniforme}

```
    {captura ou desvio}
    enquanto x ≥ 0 e x ≤ H              {enquanto dentro da parede...}
        se GU < P_A                     {se for absorvido...}
            n_A ← n_A + 1
            break                       {fim do laço}
        senão
            x ← x + (L ln GU)(2GU − 1)  {nova posição}
        fim
    fim
    {fuga ou confinamento?}
    se x > H
        n_F ← n_F + 1
    senão se x < 0
        n_C ← n_C + 1
    fim
fim
```

A.4.2 Exemplo de simulação.

Em uma simulação do problema, o algoritmo SNP foi executado com $n = 1000$ nêutrons de seção de absorção $S_A = 0.15$ e seção de desvio $S_E = 0.35$, para algumas espessuras de parede H no intervalo $[0;15]$. A Figura A.6 mostra a quantidade de nêutrons que sofreram confinamento, fuga ou absorção em função de H.

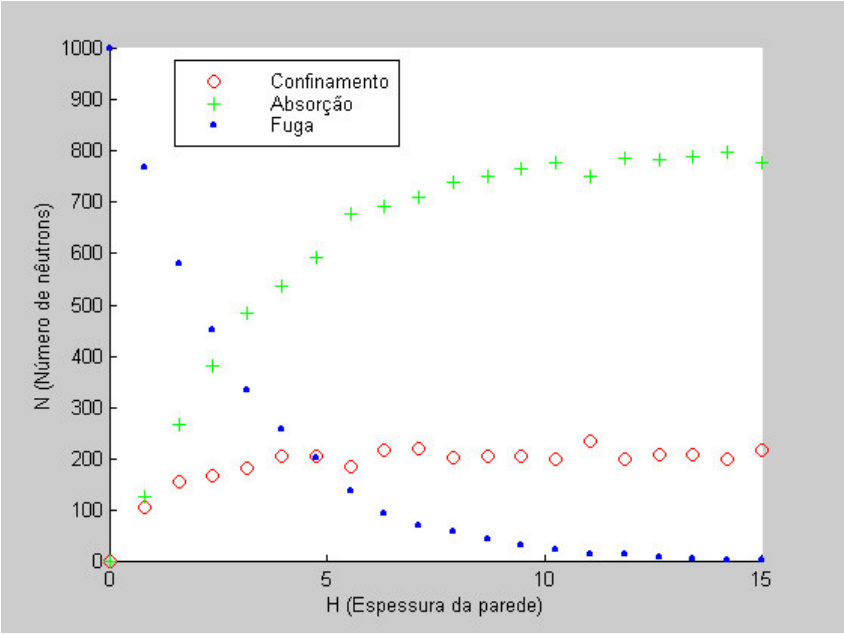

Figura A.6: Confinamento, fuga e absorção de nêutrons em função da espessura da parede.

Observando a figura, verifica-se que, como era de esperar, quanto mais espessa a parede menor a quantidade de nêutrons que a atravessam (fuga). Pode-se observar que essa quantidade decai a aproximadamente à metade a cada duas unidades de espessura. Esse valor é coincidente com o caminho livre médio das partículas

$$L = \frac{1}{S_A + S_E} = \frac{1}{0.15 + 0.35} = 2.$$

Outra observação interessante é que a quantidade de nêutrons confinados é praticamente constante a partir de 4 unidades de espessura.

A.4.3 Considerações sobre a complexidade do problema.

Simulações como essa podem ser úteis, por exemplo, no projeto de reatores nucleares onde, por questões de segurança, a informação de qual deve ser a espessura das paredes é vital.

É claro que o modelo em consideração é demasiado simples, e muitas considerações adicionais poderiam ser incluídas. De fato, as seções de choque podem ser dependentes da energia cinética dos nêutrons; assim após cada colisão esses valores devem ser recalculados. Também consideraram-se uniformemente distribuídas as direções de desvio, o que pode não ser verdade para todos os casos. Ainda é possível que, após uma colisão, o átomo da parede possa, com uma certa probabilidade, fissionar e liberar mais nêutrons.

Apêndice B
Programas para MATLAB

Os programas listados a seguir são implementações para o ambiente matemático MATLAB dos algoritmos descritos ao longo do texto. Seguem a ordem de apresentação no texto. Informações mais detalhadas sobre a linguagem do MATLAB podem ser obtidas no Manual do Usuário ou em Bravo *et alli* (1995).

```
%*********************************************************************
% Rotina: GMQ                                  Última Revisão: 12/02/2000
%*********************************************************************
% Propósito:
%   Gerador uniforme utilizando o método de Meio-de-Quadrado de
%   J. von NEUMANN.
%
% Output:
%   U: número pseudo-aleatório seguinte.
%
% Obs.:
%   1: A variável m pode ser ajustada com valores inteiros até 7.
%   2: A variável X deve ser inicializada com uma semente de
%      simulação.
%*********************************************************************
function U = GMQ,
global X;
m = 7;
X = X * X;                    % quadrado
X = fix(X / 10^fix(m / 2));   % trunca dígitos à direita
X = mod(X, 10^m);             % trunca dígitos à esquerda (nova semente)
U = X / 10^m;                 % conversão inteiro -> decimal
%************************** Fim de Arquivo **************************
```

```
%*********************************************************************
% Rotina: GCL                                  Última Revisão: 12/02/2000
%*********************************************************************
% Propósito:
%   Gerador uniforme utilizando o método de Congruência Linear de
%   D. H. Lehmer.
%
% Output:
%   U: número pseudo-aleatório.
%
% Obs.:
%    1: A variável X deve ser inicializada com uma semente de
%       simulação.
%    2: Os valores de 'a' e 'm' estão de acordo com os critérios de
%       Park & Miller. O valor 'c' é "insubstancial" segundo Knuth.
%*********************************************************************
function U = GCL,
global X;
a = 16807;
c = 1;
m = 2147483647;
```

```
X = mod(a * X + c , m);
U = X / m;
%************************** Fim de Arquivo **************************


%*********************************************************************
% Rotina: GCLC                                  Ultima Revisão: 12/02/2000
%*********************************************************************
% Propósito:
%   Gerador uniforme utilizando o método Wichmann & Hill com a
%   combinação de 3 geradores de congruência linear independentes.
%
% Output:
%   U: número pseudo-aleatório.
%
% Obs.:
%   1: As variáveis X, Y e Z devem ser inicializadas com sementes de
%      simulação.
%*********************************************************************
function U = GCLC,
global X;
global Y;
global Z;
ax = 171;
ay = 172;
az = 170;
mx = 30269;
my = 30307;
mz = 30323;
X = mod(ax * X, mx);
Y = mod(ay * Y, my);
Z = mod(az * Z, mz);
U = (X/mx) + (Y/my) + (Z/mz);
U = U - floor(U);
%************************** Fim de Arquivo **************************


%*********************************************************************
% Rotina: IC                                    Última Revisão: 12/02/2000
%*********************************************************************
% Propósito:
%   Esta rotina calcula o valor de y tal que x*y = 1 (mod m). Usa o
%   algoritmo estendido de Euclides.
%
% Input:
%   x: Número.
%   m: Módulo.
%
% Output:
%   y: Inverso congruente de X (mod m).
%
% Obs.
%   Para que o programa funcione corretamente assume-se que:
%   1. m e x são primos relativos, isto é, mdc(x,m) = 1.
%   2. 1 <= x <= m - 1.
%*********************************************************************
function y = IC(x,m),
u = [1 0 m];
v = [0 1 x];
q = floor(u(3)/v(3));
t = u - q * v;
```

```
while t(3) > 0,
  u = v;
  v = t;
  q = floor(u(3)/v(3));
  t = u - q * v;
end
if v(2) > 0,
  y = v(2);
else
  y = v(2)+ m;
end
%*********************************************************************


%*********************************************************************
% Rotina: GCI                                  Última Revisão: 12/02/2000
%*********************************************************************
% Propósito:
%   Gerador uniforme utilizando o método de Congruência Inversa de
%   Eichenauer & Lehn.
%
% Output:
%   U: número pseudo-aleatório.
%
% Obs.:
%   1: A variável X deve ser inicializada como semente de simulação.
%*********************************************************************
function U = GCI,
global X;
a = 16807;
c = 1;
m = 2147483647;
X = mod(a * IC(X,m) + c , m);
U = X / m;
%************************** Fim de Arquivo ***************************


%*********************************************************************
% Rotina: GF                                   Última Revisão: 12/02/2000
%*********************************************************************
% Propósito:
%   Gerador uniforme utilizando o método de Fibonacci.
%
% Output:
%   U: número pseudo-aleatório.
%
% Obs.:
%   1: O vetor XX com kk elementos deve ser inicializado como semente
%      de simulação.
%   2: As variáveis devem ser globais e inicializadas com kk = 55 e
%      jj = 24 (escolhidos por Knuth)
%*********************************************************************
function U = GF
global XX;
global jj;
global kk;

% Gerador
M = 2147483647;
XX(kk) = mod(XX(jj) + XX(kk),M);
U = XX(kk)/M;
```

```
% Desloca valores
jj = jj - 1;
if jj == 0,
  jj = 55;
end
kk = kk - 1;
if kk == 0,
  kk = 55;
end
%************************** Fim de Arquivo ***************************


%*********************************************************************
% Rotina: GDR                                  Última Revisão: 12/02/2000
%*********************************************************************
% Propósito:
%   Gerador uniforme utilizando o método de Deslocamento de Registro
%   de R. C. Tausworthe.
%
% Output:
%   U: número pseudo-aleatório.
%
% Obs.:
%   1: A variável U deve ser inicializada com uma semente de
%      simulação.
%   2: O parâmetro de combinação 'a' foi escolhido arbitrariamente.
%*********************************************************************
function U = GDR

% inicialização
global U;
a = 0.31415926535898;
TA = a;
TU = U;
bitT = 0;
bitV = 1;

% determinação do próximo bit
while TA > 0 | TU > 0,
  TA = 2*TA;
  if TA >= 1,
    bitA = 1;
    TA = TA - bitA;
  else
    bitA = 0;
  end
  TU = 2*TU;
  if TU >= 1,
    bitU = 1;
    TU = TU - bitU;
  else
    bitU = 0;
  end
  bitT = mod(bitT + bitA * bitU,2);
  bitV = bitV / 2;
end

% deslocamento do bit
U = 2*U;
if U >= 1,
```

```
  U = U - 1;
end
U = U + bitT * bitV;
%************************** Fim de Arquivo **************************


%*********************************************************************
% Rotina: DDG                                  Última Revisão: 12/02/2000
%*********************************************************************
% Propósito:
%   Gera números pseudo-aleatórios com Distribuição Discreta Genérica.
%
% Input:
%   p: vetor de probabilidades.
%
% Output:
%   I: índice simulado.
%
% Obs.:
%   1: O algoritmo se torna mais eficiente (realiza um menor número
%      de comparações) se o vetor de probabilidades for ordenado de-
%      crescentemente. Por exemplo: com  P = [0.4 0.3 0.2 0.1] a ro-
%      tina executa, aproximadamente, a metade das  comparações  ne-
%      cessárias com P = [0.1 0.2 0.3 0.4].
%*********************************************************************
function I = DDG(p),
U = GU;              % gerador uniforme
I = 1;               % contador
q = p(I);            % cumulativo
while U > q,         % enquanto não pertence ao intervalo...
  I = I + 1;         % ...próximo intervalo
  q = q + p(I);
end
%************************** Fim de Arquivo **************************


%*********************************************************************
% Rotina: DDU                                  Última Revisão: 12/02/2000
%*********************************************************************
% Propósito:
%   Gera números pseudo-aleatórios com Distribuição Discreta Uniforme.
%
% Input:
%   n: valor máximo.
%
% Output:
%   X: assume valor {1,2,3,...,n} com igual probabilidade
%
%*********************************************************************
function X = DDU(n),
U = GU;               % gerador uniforme
X = 1 + floor(n*U);   % valor simulado
%************************** Fim de Arquivo **************************


%*********************************************************************
% Rotina: DCI                                  Última Revisão: 12/02/2000
%*********************************************************************
% Propósito:
%   Gera números pseudo-aleatórios com Distribuição Contínua pelo
%   método de Inversão.
```

```
%
% Input:
%   Finv: função inversa de F.
%
% Output:
%   X: variável pseudo-aleatória com FPC F.
%
%*********************************************************************
function X = DCI(Finv),
x = GU;                 % gerador uniforme
X = feval(Finv,x);      % valor simulado
%************************* Fim de Arquivo **************************


%*********************************************************************
% Rotina: DCAR                                  Última Revisão: 12/02/2000
%*********************************************************************
% Propósito:
%   Gera números pseudo-aleatórios com Distribuição Contínua pelo
%   método de Aceitação - Rejeição.
%
% Input:
%   f : função FDP de X.
%   g : função FDP de Z.
%   c : constante de majoração.
%   GZ: gerador da variável Z (DCI, por exemplo).
%
% Output:
%   x: variável pseudo-aleatória.
%
%*********************************************************************
function x = DCAR(f,g,c,GZ),
z = GZ;
while GU * c * feval(g,z) > feval(f,z),
  z = GZ;
end
x = z
%************************* Fim de Arquivo **************************


%*********************************************************************
% Rotina: DdAR                                  Ultima Revisão: 12/02/2000
%*********************************************************************
% Propósito:
%   Gera números pseudo-aleatórios com Distribuição d-dimensional pelo
%   método de Aceitação Rejeição.
%
% Input:
%   f: função FDP de X.
%   g: função FDP de Z.
%   c: constante de majoração.
%   G: gerador d-dimensional da variável Z.
%
% Output:
%   x: variável pseudo-aleatória com FPC F.
%
% Obs.:
%   1: Esta rotina é idêntica a rotina DCAR, a exceção da rotina G de
%      geração da variável z (d-dimensional)
%*********************************************************************
function x = DdAR(f,g,c,G),
```

```
z = G;
while GU * c * feval(g,z) > feval(f,z),
  z = G;
end
x = z
%************************* Fim de Arquivo **************************


%*********************************************************************
% Rotina: DMM                                  Última Revisão: 20/01/2000
%*********************************************************************
% Propósito:
%   Esta rotina gera números pseudo-aleatórios com Distribuição
%   d-dimensional pelo método de Metropolis.
%
% Input:
%   fun: nome de uma função que contém a FDP f.
%   a,b: vetores com os limites da região de amostragem.
%   d  : dimensão da variável aleatória.
%   x  : último valor da variável.
%
% Output:
%   x  : Próximo valor da variável.
%
% Obs.:
%   Antes de usar os valores de 'x' em rotinas de simulação, descartar
%   os 10 - 100 primeiros valores.
%*********************************************************************
function [x] = DMM(f,a,b,d,x),

% inicialização
s = (b - a) / 2;    % passo.

% amostragem
Ok1 = 0;            % Flag
while ~Ok1,

  % escolhe novo ponto
  Ok2 = 0;          % Flag
  while ~Ok2,
    Ok2 = 1;
    for j = 1 : d
      nx(j) = x(j) + s(j)*(2*GUP-1);   % novo ponto
      if nx(j) < a(j) | nx(j) > b(j),  % está dentro de [a;b]
        Ok2 = 0;
        break
      end
    end
  end

  % aceitação - rejeição
  if feval(fun,nx) >= GUP * feval(fun,x), % aceita?
    x = nx;
    Ok1 = 1;
  end

end
%************************* Fim de Arquivo **************************


%*********************************************************************
```

```
% Rotina: DExp                                  Última Revisão: 12/02/2000
%*********************************************************************
%  Propósito: Esta rotina gera um número pseudo-aleatório com FDP
%    Exponencial.
%
%  Input:
%    M: Valor médio (esperança)
%
%  Output:
%    X: variável pseudo-aleatório.
%
%  Obs.:
%    1.  Este algoritmo usa o método de inversão.
%*********************************************************************
function X = DExp(M),
X = - M * log(GUP);
%************************* Fim de Arquivo ***************************


%*********************************************************************
% Rotina: DNormal                              Ultima Revisão: 12/02/2000
%*********************************************************************
%  Propósito: Esta rotina gera um par de número pseudo-aleatório
%    independentes com FDP Normal (de Gauss).
%
%  Input:
%    mi: Valor médio (esperança)
%    sigma: Desvio padrão (sqrt(variância))
%
%  Output:
%    X1,X2: valores pseudo-aleatórios.
%
%*********************************************************************
function [X1,X2] = DNormal(M,D),
U1 = GU;
U2 = GU;
X1 = sqrt(-2*log(U1)) * cos(2*pi*U2);
X2 = sqrt(-2*log(U1)) * sin(2*pi*U2);
X1 = mi + X1 * sigma;
X2 = mi + X2 * sigma;
%************************* Fim de Arquivo ***************************


%*********************************************************************
% Rotina: QMC                                  Última Revisão: 12/02/2000
%*********************************************************************
% Propósito:
%   Esta rotina realiza a Quadratura de Monte Carlo d-dimensional.
%
% Input:
%   fun: nome de uma função que contem o integrando f e a região de
%        integração V.
%   a,b: vetores com os limites da região de integração.
%   d  : dimensão da quadratura.
%   n  : Número de pontos de amostragem.
%
% Output:
%   Q: Estimativa do valor da Quadratura.
%   e: Estimativa do erro (desvio padrão).
%*********************************************************************
function [Q,e] = QMC(fun,a,b,d,n),
```

```
% inicialização
s = 0;              % acumulador para <f>
t = 0;              % acumulador para <f2>
l = b - a;          % arestas da hiper-caixa.
w = prod(l);        % hiper-volume.

% amostragem
for i = 1 : n,
  for j = 1 : d
    x(j) = a(j) + l(j) * GU;    % ponto de amostragem
  end
  gx = feval(fun,x);            % avaliação da função
  s = s + gx;
  t = t + gx * gx;
end

% estimativas
mg  = s / n;                    % <fun>
mg2 = t / n;                    % <fun^2>
Q   = w * mg;
e   = w * sqrt((mg2 - mg^2)/n);
%************************* Fim de Arquivo **************************


%*********************************************************************
% Rotina: QMC2                                  Última Revisão: 12/02/2000
%*********************************************************************
% Propósito:
%   Variação da rotina QMC, onde o critério de parada é a estimativa
%   de erro e.
%
% Input:
%   fun : nome de uma função que contém o integrando f e a região de
%         integracao V.
%   A,B : vetores com os limites da região de integração.
%   D   : dimensão da quadratura.
%   emax: erro máximo admitido.
%
% Output:
%   Q: Estimativa do valor da Quadratura.
%   e: Estimativa do erro.
%   n: Número de pontos amostrados.
%
%*********************************************************************
function [Q,e,n] = QMC2(fun,A,B,D,emax),

% Inicialização
s = 0;              % acumulador para <f>
t = 0;              % acumulador para <f2>
L = B - A;          % arestas da região de integração.
w = prod(L);        % volume da região de integração.
e = inf;            % estimativa de erro.
n = 0;              % pontos de amostragem.
nmin = 100;         % número mínimo de pontos de amostragem

% Amostragem Monte Carlo
while e > emax,

  % Ponto de amostragem
  for i = 1 : D,
```

```
    x(i,1) = A(i) + L(i) * GU;
  end
  n = n + 1;

  % Avaliação da função
  f = feval(fun,x);
  s = s + f;
  t = t + f * f;

  % Estimativa de e
  if n >= nmin,
    nmin = nmin + 100;
    mf  = s / n;
    mf2 = t / n;
    e = w * sqrt((mf2 - mf^2)/n);
    Q = w * mf;
  end
end
%************************* Fim de Arquivo *************************


%*********************************************************************
% Rotina: QMCI                                  Última Revisão: 12/02/2000
%*********************************************************************
% Propósito:
%   Esta rotina realiza a Quadratura de Monte Carlo d-dimensional
%   utilizando a técnica de Amostragem de Importância.
%
% Input:
%   f  : nome de uma função que contém o integrando f na região de
%        integração V.
%   p  : nome de uma função que contém a função peso
%   a,b: vetores com os limites da região de integração.
%   d  : dimensão da quadratura.
%   n  : Número de pontos de amostragem.
%
% Output:
%   Q: Estimativa do valor da Quadratura.
%   e: Estimativa do erro (desvio padrão).
%
%*********************************************************************
function [Q,e] = QMCI(f,p,a,b,d,n),

% inicialização
s1 = 0;              % acumulador para <h>
s2 = 0;              % acumulador para <h^2>

% amostragem
for i = 1 : n,
  x = ...            % x calculado com distribuição p
  h = feval(f,x)/feval(p,x);
  s1 = s1 + h;
  s2 = s2 + h * h;
end
% estimativas
mh  = s1 / n;                  % <h>
mh2 = s2 / n;                  % <h^2>
Q   = mh;
e   = sqrt((mh2 - mh^2) / n);
%************************* Fim de Arquivo *************************
```

```
%*********************************************************************
% Rotina: QMCA                                  Última Revisão: 12/02/2000
%*********************************************************************
% Propósito:
%   Esta rotina realiza a Quadratura de Monte Carlo D-dimensional, com
%   técnica de redução de variância por amostragem antitética.
%
% Input:
%   g  : nome de uma função que contém o integrando f e a região de
%        integração V.
%   A,B: vetores com os limites da região de integração.
%   D  : dimensão da quadratura.
%   N  : Número de pontos de amostragem.
%
% Output:
%   Q: Estimativa do valor da Quadratura.
%   e: Estimativa do erro (desvio padrão).
%*********************************************************************
function [Q,e] = QMCA(g,A,B,D,N),
global seed

% inicialização
if rem(N,2) == 1    % N deve ser par
  N = N + 1;
end
s = 0;              % acumulador para <f>
t = 0;              % acumulador para <f2>
L = B(:) - A(:);    % arestas da hiper-caixa.
w = prod(L);        % hiper-volume.

% amostragem
for i = 1 : N/2,
  for j = 1 : D
    x(j,1) = A(j) + L(j) * GU;      % 1.o ponto de amostragem
  end
  gx = feval(g,x);                  % avaliação da função
  s = s + gx;
  t = t + gx * gx;

  x = A + (B - x);                  % 2.o ponto de amostragem
(antitético)
  gx = feval(g,x);                  % avaliação da função
  s = s + gx;
  t = t + gx * gx;
end

% estimativas
mg  = s / N;                        % <g>
mg2 = t / N;                        % <g^2>
Q   = w * mg;
e   = w * sqrt((mg2 - mg^2)/N);
%************************* Fim de Arquivo ************************


%*********************************************************************
% Rotina: QMCE                                  Última Revisão: 12/02/2000
%*********************************************************************
% Propósito:
%   Esta rotina realiza a Quadratura de Monte Carlo D-dimensional, com
%   técnica de redução de variância por amostragem estratificada.
```

```
%
% Input:
%   FUN : nome de uma função que contém o integrando f e a região de
%         integração V.
%   A,B : vetores com os limites da região de integração.
%   D   : dimensão da quadratura.
%   emax: erro máximo admitido.
%   P   : vetor com o número de divisões em cada dimensão.
%
% Output:
%   Q: Estimativa do valor da Quadratura.
%   e: Estimativa do erro (desvio padrão).
%   n: Número de pontos amostrados.
%*********************************************************************
function [Q,e,n] = QMCE(FUN,A,B,D,emax,P),

% inicialização
NP = prod(P);             % número de sub-regiões.
Diag = (B - A) ./ P;      % Vetor diagonal
Q = 0;
e = 0;
n = 0;

% Amostragem Estratificada.
for i = 0 : NP - 1,          % Para cada sub-região...

  % Combinação
  for j = 1 : D,
    c(j,1) = rem(i,P(j));
    i = (i - c(j,1))/P(j);
  end

  % Vértices e volume da sub-região
  a = A + Diag .* c;
  b = a + Diag;

  % Amostragem Monte Carlo
  [Qsub,esub,nsub]=QMC2(FUN,a,b,D,emax/sqrt(NP));

  % Acumulação de estimativas
  Q = Q + Qsub;
  e = sqrt(e*e + esub*esub);
  n = n + nsub;
end
%************************ Fim de Arquivo ************************


%*********************************************************************
% Rotina: QMCIA                              Última Revisão: 12/02/2000
%*********************************************************************
% Propósito:
%   Esta rotina realiza a Quadratura de Monte Carlo d-dimensional
%   usando um método misto de amostragem estratificada com amostragem
%   de importância através de uma função peso adaptável.
%
% Input:
%   F: nome de uma função que contem o integrando f e a região de
%      integração V.
%   A,B: vetores com os limites da região de integração.
%   S  : Número de subintervalos em cada dimensão.
%   M  : Número de iterações.
```

```
%   N  : Número de pontos de amostragem.
%
% Output:
%   Q: Estimativa do valor da Quadratura.
%   e: Estimativa do erro (desvio padrão).
%   P: Consistência.
%
% Obs.:
%   1: Se P >> 1. o resultado é suspeito.
%**********************************************************************
function [Q,e,P] = QMCIA(F,A,B,S,M,N),

% inicialização
D = length(A);                    % dimensão
Diag = (B - A) / S;               % diagonal
Vol = prod(Diag);                 % volume das sub-regiões
G = 1 ./ (B-A) * ones(1,S);       % FDP inicial
QQ = zeros(1,M);                  % estimativa de I
ee = zeros(1,M);                  % estimativa de E


% Amostragem
for i = 1 : M,                    %Para cada iteração

  % inicialização
  g = zeros(D,S);
  nt = 0;
  sh = 0;
  sh2 = 0;

  for j = 1 : S^D,            % Para cada amostragem na sub-região...

    % Combinação
    temp = j - 1;
    for k = 1 : D,
      c(k,1) = rem(temp,S);
      temp = (temp - c(k,1))/S;
    end

    % Vértices e volume da sub-região
    a = A + Diag .* c;
    b = a + Diag;

    % peso
    c = c + 1;
    p = 1;
    for k = 1 : D,
      p = p * G(k,c(k));
    end

    % amostragem
    ns = floor(N * p * Vol + 0.5);
    nt = nt + ns;
    sf = 0;
    for k = 1 : ns,

      % ponto amostral
      for l = 1 : D
        x(l,1) = a(l) + Diag(l) * GU;
      end
      f = feval(F,x);
```

```
      sf = sf + f;
      h = f/p;
      sh = sh + h;
      sh2 = sh2 + h*h;

    end

    % prepara novo vetor FDP
    if ns ~= 0,
      for k = 1 : D,
        g(k,c(k)) = g(k,c(k)) + abs(sf / ns);
      end
    end


  end            % fim da amostragem na sub-região...

  % Acumulação de estimativas
  mh  = sh / nt;                    % <h>
  mh2 = sh2 / nt;                   % <h^2>
  QQ(i)= mh;
  ee(i) = sqrt((mh2 - mh^2) / nt);

  % novo vetor FDP
  tot = 1 ./ (Diag .* sum(g,2));
  for j = 1 : D,
    G(j,:) = g(j,:) * tot(j);
  end

end            % fim da iteração

% Melhor estimativa
t = sum(1 ./ ee .^ 2);
Q = sum(QQ ./ ee .^ 2) / t;
e = 1 / sqrt(t);
P = sum((QQ - Q).^2 ./ ee.^2) / (M - 1);
%************************* Fim de Arquivo **************************


%*********************************************************************
% Rotina: QPR                                  Última Revisão: 12/02/2000
%*********************************************************************
% Propósito:
%   Esta rotina realiza uma Quadratura D-dimensional usando a técnica
%   da Partição Regular.
%
% Input:
%   fun: nome de uma função que contém o integrando e a região de
%        integração.
%   A,B: vetor com os limites da região de integração.
%   D  : dimensão da quadratura.
%   n  : vetor com número de pontos de amostragem em cada dimensão.
%
% Output:
%   Q: Estimativa do valor da Quadratura.
%
% Obs.:
%   1: Para que os pontos n sejam proporcionais ao comprimento de
%      cada intervalo, fazer n = ceil(L*(N/V)^(1/D))
%*********************************************************************
function [Q] = QPR(fun,A,B,D,n),
```

```
% inicialização
L = B(:) - A(:);        % arestas da hiper-caixa.
v = prod(L);            % hiper-volume.
H = L./n;               % tamanho do intervalo de partição.
N = prod(n);            % numero de pontos da partição.

% coordenadas dos pontos da partição
for i = 1 : D,
  X(i,1) = A(i,1) + H(i)/2;
  for j = 2 : n(i),
    X(i,j) = X(i,j-1) + H(i);
  end
end

% amostragem
s = 0;                        % acumulador para <f>
for j = 0 : N - 1,
  for i = 1 : D,              % índices para coordenadas
    k = rem(j,n(i));
    j = (j - k)/n(i);
    x(i)= X(i,k + 1);
  end
  f = feval(fun,x);           % avaliação da função
  s = s + f;
end

% estimativas
Q   = v * (s / N);
%************************* Fim de Arquivo *************************


%*********************************************************************
% Rotina: GH                                  Última Revisão: 12/02/2000
%*********************************************************************
%  Propósito:
%     Calcula o j-ésimo valor da seqüência de Halton (Hjb), base b, no
%     intervalo [0,1).
%
%  Input:
%    j: número seqüencial.
%    b: base.
%
%  Output:
%    H: número de Halton
%*********************************************************************
function H = GH(j,b),
i = 0;
while j > 0,
  i = i + 1;
  a(i) = rem(j,b);
  j = (j - a(i)) / b;
end
H = 0;
while i > 0,
  H = (H + a(i)) / b;
  i = i - 1;
end
%************************* Fim de Arquivo *************************
```

```
%*********************************************************************
% Rotina: QH                                   Última Revisão: 12/02/2000
%*********************************************************************
% Propósito:
%   Esta rotina realiza a Quadratura Quase Monte Carlo D-dimensional
%   pela seqüência de Halton.
%
% Input:
%   fun: nome de uma função que contém o integrando e a região de
%        integração.
%   a,b: vetores com os limites da região de integração.
%   d  : dimensão da quadratura.
%   n  : Número de pontos de amostragem.
%   P  : Vetor de índices.
%
% Output:
%   Q: Estimativa do valor da Quadratura.
%
%*********************************************************************
function Q = QH(fun,a,b,d,n,P),

% inicialização
s = 0;              % acumulador para <f>
l = b - a;          % arestas da hiper-caixa.
v = prod(l);        % hiper-volume.
p = [2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89
97];
nmin = 100;

% amostragem
for j = nmin + 1 : n + nmin,
  for i = 1 : d
    x(i) = a(i) + l(i) * GH(j,p(P(i)));    % ponto de amostragem
  end
  s = s + feval(fun,x);
end

% Quadratura
Q = v * s / n;
%************************* Fim de Arquivo *************************


%*********************************************************************
% Rotina: GS                                   Última Revisão: 12/02/2000
%*********************************************************************
%  Propósito:
%    Calcula uma matriz S(i,j) contendo o j-ésimo valor da i-ésima
%    seqüência de Sobol no intervalo [0,1).
%
%  Input:
%    P: vetor contendo o número-tipo das seqüências. P.ex.: P=[1 4 7].
%    N: tamanho de cada seqüência.
%
%  Output:
%    S: número de Sobol
%*********************************************************************
function  S = GS(P,N),

% inicialização
w = ceil(log2(N));
q = [2 3 3 4 4 5 5 5  5  5  5 6  6  6  6  6  6 7 7 7 7  7  7  7  7];
```

```
a = [1 1 2 1 4 2 4 7 11 13 14 1 13 16 19 22 25 1 4 7 8 14 19 21 28];

for i = 1 : length(P),
  % cálculo de m
  for j = 1 : q(P(i)),
    m(i,j) = 2^j-1;
  end
  for j = q(P(i)) + 1 : w
    m(i,j) = bitxor(2^q(P(i)) * m(i,j-q(P(i))),m(i,j-q(P(i))));
    t = a(P(i));
    for k = 1 : q(P(i))-1,
      r = rem(t,2);
      m(i,j) = bitxor(m(i,j),2^(q(P(i))-k) * r * m(i,j-q(P(i))+k));
      t = (t-r)/2;
    end
  end

  % cálculo de V
  for j = 1 : w,
    V(i,j) = m(i,j)/2^j;
  end

  %cálculo de S
  S(i,1)= 0;
  for j = 1 : N-1,
    t = j-1;
    k = 1;
    r = rem(t,2);
    while r ~= 0,
      t = (t-r)/2;
      k = k + 1;
      r = rem(t,2);
    end
    if k > w,
      k = w;
    end
    S(i,j+1) = bitxor(S(i,j) * 2^w , V(i,k) * 2^w) / 2^w;
  end
end
%************************** Fim de Arquivo **************************


%*********************************************************************
% Rotina: QS                                   Última Revisão: 12/02/2000
%*********************************************************************
% Propósito:
%   Esta rotina realiza a Quadratura Quase Monte Carlo D-dimensional
%   pela seqüência de Sobol.
%
% Input:
%   fun: nome de uma função que contém o integrando e a região de
%        integração.
%   A,B: vetores com os limites da região de integração.
%   D  : dimensão da quadratura.
%   N  : Número de pontos de amostragem.
%   P  : vetor de permutação de seqüências.
%
% Output:
%   Q: Estimativa do valor da Quadratura.
%*********************************************************************
function Q = QS(fun,A,B,D,N,P),
```

```
% inicialização
s = 0;              % acumulador para <f>
L = B(:) - A(:);    % arestas da hiper-caixa.
v = prod(L);        % hiper-volume.
S = GS(P,N);        % seqüências de Sobol

% amostragem
for j = 1 : N,
  for i = 1 : D
    x(i) = A(i) + L(i) * S(i,j);    % ponto de amostragem
  end
  s = s + feval(fun,x);
end

% Quadratura
Q = v * s / N;
%************************* Fim de Arquivo *************************


%*********************************************************************
% Rotina: SSA                                  Última Revisão: 12/02/2000
%*********************************************************************
% Propósito:
%   Esta rotina simula um SISTEMA DE ATENDIMENTO de pedidos. Determina
%   o número de pedidos atendidos e o número total de pedidos
%   ingressantes. Usa-se a distribuição Exponencial para gerar os
%   intervalos de tempo entre pedidos e também para gerar a duração
%   dos atendimentos.
%
% Input:
%   n  : número de canais de atendimento.
%   TIM: tempo médio de ingresso de pedidos.
%   TAM: tempo médio de atendimento.
%   T  : tempo total de amostragem
%
% Output:
%   NA: número pedidos atendidos.
%   NT: número total de pedidos.
%
%*********************************************************************
function [NA,NT] = SSA(n,TIM,TAM,T),

% inicializa contadores.
NT = 0;                    % número total de pedidos.
NA = 0;                    % número de pedidos atendidos.

% inicializa canais.
for i = 1 : n
  c(i) = 0;                % instante em que o canal será liberado.
end

% laço temporal.
t = 0;                     % tempo.
while t < T,

  % Gera pedido.
  t = t + DExp(TIM);       % instante do pedido.
  NT = NT + 1;             % incrementa número de pedidos.

  % Procura canal livre.
```

```
  for i = 1 : n,              % para todos os canais...
    if t > c(i),              % se canal esta livre...
      c(i) = t + DExp(TAM);   % instante de liberação do canal.
      NA = NA + 1;            % conta atendimento.
      break
    end
  end

end
%************************* Fim de Arquivo *************************


%*********************************************************************
% Rotina: SCE                                  Última Revisão: 12/02/2000
%*********************************************************************
% Propósito:
%   Esta rotina simula o valor para a propriedade global U do circuito
%   elétrico mostrado na Figura A.1. Esta rotina simula o
%   valor dos componentes do circuito como variáveis com distribuição
%   Normal.
%
% Input:
%   C: Valor nominal dos componentes.
%   D: Desvio padrão dos valores nominais dos componentes.
%
% Output:
%   U: Potência dissipada em R2.
%*********************************************************************
function U = SCE(C,D),

% Inicialização dos componentes
for i = 1 : 7,
  R(i) = DNormal(C(i),D(i));
end
ep = DNormal(C(8),D(8));

% montagem do SELA
A(1,1) =  R(4) + R(5);
A(1,2) = -R(5);
A(1,3) =  0;
A(1,4) = -R(4);
A(2,1) = -R(5);
A(2,2) =  R(1) + R(5) + R(6);
A(2,3) = -R(6);
A(2,4) =  0;
A(3,1) =  0;
A(3,2) = -R(6);
A(3,3) =  R(2) + R(6) + R(7);
A(3,4) = -R(7);
A(4,1) = -R(4);
A(4,2) =  0;
A(4,3) = -R(7);
A(4,4) =  R(3) + R(4) + R(7);
B = [ep; 0; 0; 0];

% Cálculo de U
I = A\B;
U = R(2) * I(3)^2;
%************************* Fim de Arquivo *************************
```

```
%*********************************************************************
% Rotina: SRJ                                  Última Revisão: 12/02/2000
%*********************************************************************
% Propósito: Esta rotina simula um jogo de apostas de muitos
%   jogadores.
%
% Input:
%   F: fortuna inicial de cada jogador.
%   P: probabilidade de vitória de cada jogador.
%   N: número máximo de rodadas.
%
% Output:
%   H: matriz em que cada coluna representa o histórico da fortuna  de
%      um jogador.
%*********************************************************************
function [H] = SRJ(F,P,N),

% inicialização
M = length(P);        % número de jogadores
i = 1;                % rodada
H(i,:) = F;           % fortuna inicial

% jogo
while i <= N & min(F) > 0,       % enquanto nenhuma ruína...
   i = i + 1;                    % próxima rodada
   F = F - ones(1,M);            % aposta
   j = DDG(P);                   % vencedor
   F(j) = F(j) + M;              % prêmio
   H(i,:) = F;                   % histórico
end
%************************** Fim de Arquivo **************************


%*********************************************************************
% Rotina: SNP                                  Última Revisão: 12/02/2000
%*********************************************************************
% Propósito: Esta rotina simula a passagem de nêutrons através de uma
%   parede.
%
% Input:
%  SA: seção de choque de absorção.
%  SE: seção de choque de espalhamento.
%  H : espessura da parede.
%  N : número de nêutrons incidentes.
%
% Output:
%  NC: quantidade que sofreu confinamento.
%  NA: quantidade que sofreu absorção.
%  NF: quantidade que atravessou (fuga).
%
%*********************************************************************
function [NC,NA,NF] = SNP(SA,SE,H,N),

% inicialização
NC = 0;              % contadores de partículas
NA = 0;
NF = 0;
S = SA + SE;         % seção de choque efetiva
PA = SA / S;         % probabilidade de absorção
PE = SE / S;         % probabilidade de espalhamento
L = 1/S;             % livre caminho médio
```

```
% bombardeamento de partículas
for i = 1 : N,                    % para todas as partículas...

  x = -L*log(GUP);                % posição inicial

  % absorção ou espalhamento
  while x >= 0 & x <= H,          % enquanto dentro da parede...
    if GUP < PA,                  % se for absorvido...
      NA = NA + 1;                % incrementa contador
      break
    else
      x = x + (L*log(GUP))*(2*GUP-1);       % nova posição
    end
  end

  % fuga ou confinamento?
  if x > H,
    NF = NF + 1;
  elseif x < 0
    NC = NC + 1;
  end

end
%************************** Fim de Arquivo **************************
```