

Métodos Computacionais em Física I (FIW234)

Turmas IFA e IFB

Métodos de Monte Carlo

Edivaldo M. Santos e João R. T. de Mello Neto

Aula 9

Introdução

- Monte Carlo é a denominação geral dada a uma classe bastante ampla de métodos que lançam mão da amostragem aleatória de um conjunto de variáveis para realizar certos tipos de cálculo.

Introdução

- Monte Carlo é a denominação geral dada a uma classe bastante ampla de métodos que lançam mão da amostragem aleatória de um conjunto de variáveis para realizar certos tipos de cálculo.
- Métodos de Monte são especialmente úteis no trato de sistemas com um número elevado de graus de liberdade. Para tais sistemas, pode-se tornar difícil ou até mesmo impossível, resolver o conjunto acoplado de equações diferenciais ou eqs não-lineares para todas as variáveis de interesse.
- Exemplos de alguns sistemas desse tipo: fluidos, materiais de estrutura cristalina complexa, variáveis de mercados financeiros, etc.

Introdução

- Monte Carlo é a denominação geral dada a uma classe bastante ampla de métodos que lançam mão da amostragem aleatória de um conjunto de variáveis para realizar certos tipos de cálculo.
- Métodos de Monte são especialmente úteis no trato de sistemas com um número elevado de graus de liberdade. Para tais sistemas, pode-se tornar difícil ou até mesmo impossível, resolver o conjunto acoplado de equações diferenciais ou eqs não-lineares para todas as variáveis de interesse.
- Exemplos de alguns sistemas desse tipo: fluidos, materiais de estrutura cristalina complexa, variáveis de mercados financeiros, etc.
- Os sistemas acima e muitos outros são tão complexos que parte da dinâmica pode ser considerada como de natureza aleatória ou estocástica, isto é, não é possível, a partir de um evento, prever o resultado do próximo, mas simplesmente saber a probabilidade com que ele ocorra.

Introdução

- Monte Carlo é a denominação geral dada a uma classe bastante ampla de métodos que lançam mão da amostragem aleatória de um conjunto de variáveis para realizar certos tipos de cálculo.
- Métodos de Monte são especialmente úteis no trato de sistemas com um número elevado de graus de liberdade. Para tais sistemas, pode-se tornar difícil ou até mesmo impossível, resolver o conjunto acoplado de equações diferenciais ou eqs não-lineares para todas as variáveis de interesse.
- Exemplos de alguns sistemas desse tipo: fluidos, materiais de estrutura cristalina complexa, variáveis de mercados financeiros, etc.
- Os sistemas acima e muitos outros são tão complexos que parte da dinâmica pode ser considerada como de natureza aleatória ou estocástica, isto é, não é possível, a partir de um evento, prever o resultado do próximo, mas simplesmente saber a probabilidade com que ele ocorra.
- Outra classe de problemas, que não necessariamente envolve um grande número de graus de liberdade, em que Métodos de Monte Carlo podem ser utilizados é o cálculo de integrais. Dessas, é a na avaliação de integrais multidimensionais que o método mostra seu verdadeiro poder, dado que para integrais em 1 ou 2 dimensões os métodos vistos em aulas anteriores são bastante bem sucedidos.

- Apesar de suas raízes remontarem ao século XVIII com os trabalhos de teoria de probabilidade, métodos de Monte Carlo só vieram a se tornar "populares" a partir de 1945 com o aparecimento dos primeiros computadores.
- A razão para isso é que para aplicar o método, uma grande quantidade de números aleatórios deve ser gerada, o que só se tornou viável com a utilização de máquinas de cálculo rápido.

- Apesar de suas raízes remontarem ao século XVIII com os trabalhos de teoria de probabilidade, métodos de Monte Carlo só vieram a se tornar "populares" a partir de 1945 com o aparecimento dos primeiros computadores.
- A razão para isso é que para aplicar o método, uma grande quantidade de números aleatórios deve ser gerada, o que só se tornou viável com a utilização de máquinas de cálculo rápido.
- O nome foi cunhado pelo matemático John Von Neumann na década de 40 quando trabalhava no projeto Manhattan no Laboratório Nacional de Los Alamos, EUA. A inspiração de Von Neumann, foi o casino Monte Carlo em Mônaco, onde a aleatoriedade das apostas estava sempre presente. Além disso, o nome devia ser apropriado a um projeto secreto.

Sequências pseudo-aleatórias

- Antes de seguir em frente, é preciso que todos estejam plenamente cientes do seguinte fato:
- Um computador só é capaz de realizar cálculos determinísticos. A partir de uma determinada condição inicial, ele é capaz de realizar operações plenamente previsíveis e reprodutíveis. Não importa se são complexas e/ou numerosas, se nós ou outra máquina as repetirmos obteremos o mesmo valor que a máquina original.

Sequências pseudo-aleatórias

- Antes de seguir em frente, é preciso que todos estejam plenamente cientes do seguinte fato:
- Um computador só é capaz de realizar cálculos determinísticos. A partir de uma determinada condição inicial, ele é capaz de realizar operações plenamente previsíveis e reprodutíveis. Não importa se são complexas e/ou numerosas, se nós ou outra máquina as repetirmos obteremos o mesmo valor que a máquina original.
- Isto é, não há qualquer aleatoriedade nas operações realizadas por um computador.
- **Sendo assim, nenhuma sequência de números geradas por um computador é, de fato, aleatória!**

Sequências pseudo-aleatórias

- Antes de seguir em frente, é preciso que todos estejam plenamente cientes do seguinte fato:
- Um computador só é capaz de realizar cálculos determinísticos. A partir de uma determinada condição inicial, ele é capaz de realizar operações plenamente previsíveis e reprodutíveis. Não importa se são complexas e/ou numerosas, se nós ou outra máquina as repetirmos obteremos o mesmo valor que a máquina original.
- Isto é, não há qualquer aleatoriedade nas operações realizadas por um computador.
- **Sendo assim, nenhuma sequência de números geradas por um computador é, de fato, aleatória!**
- Como pode então um computador ser utilizado para implementar métodos de Monte Carlo?
- RESPOSTA: Sequências pseudo-aleatórias.

Sequências pseudo-aleatórias

- Já foi dito que um fenômeno é dito aleatório ou estocástico quando não é possível prever o resultado exato do próximo evento a partir das observações do evento anterior. Tudo o que podemos prever é a probabilidade de ocorrências dos valores de certas variáveis no evento seguinte.

Sequências pseudo-aleatórias

- Já foi dito que um fenômeno é dito aleatório ou estocástico quando não é possível prever o resultado exato do próximo evento a partir das observações do evento anterior. Tudo o que podemos prever é a probabilidade de ocorrências dos valores de certas variáveis no evento seguinte.
- Nesse sentido, numa sequência de números genuinamente aleatória, não há qualquer noção de período. É possível gerar um quantidade arbitrariamente grande de tais números sem que se observe repetição da sequência.

Sequências pseudo-aleatórias

- Já foi dito que um fenômeno é dito aleatório ou estocástico quando não é possível prever o resultado exato do próximo evento a partir das observações do evento anterior. Tudo o que podemos prever é a probabilidade de ocorrências dos valores de certas variáveis no evento seguinte.
- Nesse sentido, numa sequência de números genuinamente aleatória, não há qualquer noção de período. É possível gerar um quantidade arbitrariamente grande de tais números sem que se observe repetição da sequência.
- Sequências pseudo-aleatórias são conjuntos ordenados de números cujo período é grande o suficiente para que numa dada aplicação, não seja possível perceber a natureza determinística da sequência.
- São essas as sequências geradas por qualquer máquina como aproximação para os verdadeiros números aleatórios.

Sequências pseudo-aleatórias

- Já foi dito que um fenômeno é dito aleatório ou estocástico quando não é possível prever o resultado exato do próximo evento a partir das observações do evento anterior. Tudo o que podemos prever é a probabilidade de ocorrências dos valores de certas variáveis no evento seguinte.
- Nesse sentido, numa sequência de números genuinamente aleatória, não há qualquer noção de período. É possível gerar um quantidade arbitrariamente grande de tais números sem que se observe repetição da sequência.
- Sequências pseudo-aleatórias são conjuntos ordenados de números cujo período é grande o suficiente para que numa dada aplicação, não seja possível perceber a natureza determinística da sequência.
- São essas as sequências geradas por qualquer máquina como aproximação para os verdadeiros números aleatórios.
- **Apesar da natureza de tais sequências ser claramente determinística, o termo gerador de números aleatórios é amplamente utilizado. Faremos uso desse termo aqui também, mas o aluno deve ter claro em mente de que esse é um abuso de linguagem.**

Sequências pseudo-aleatórias: comprimento

- O n -ésimo elemento r_n de uma sequência pseudo-aleatória pode ser gerado pela aplicação de uma função f apropriada ao elemento anterior da sequência:

$$r_n = f(r_{n-1}). \quad (1)$$

Sequências pseudo-aleatórias: comprimento

- O n -ésimo elemento r_n de uma sequência pseudo-aleatória pode ser gerado pela aplicação de uma função f apropriada ao elemento anterior da sequência:

$$r_n = f(r_{n-1}). \quad (1)$$

- Já vimos no curso que numa máquina onde a palavra (word) tem N bits, é possível representar a quantidade máxima de 2^N números diferentes, já que a unidade básica de informação são os bits.
- Dessa forma, a sequência da eq. acima irá, inevitavelmente, se repetir após, no máximo, 2^N elementos.
- Na prática, o período de uma sequência pseudo-aleatória será menor que o número máximo de possíveis representações num computador.

Sequências pseudo-aleatórias: exemplos

- Uma função comumente usada por geradores de números aleatórios é uma combinação de uma função linear atuando sobre inteiros, seguida da operação de resto:

$$r_n = (ar_{n-1} + b) \bmod m \quad \text{ou} \quad r_n = (ar_{n-1} + b) \% m \quad (2)$$

Sequências pseudo-aleatórias: exemplos

- Uma função comumente usada por geradores de números aleatórios é uma combinação de uma função linear atuando sobre inteiros, seguida da operação de resto:

$$r_n = (ar_{n-1} + b) \bmod m \quad \text{ou} \quad r_n = (ar_{n-1} + b) \% m \quad (2)$$

- Em C, a biblioteca padrão `stdlib` contém a função `rand()` que usa um desses chamados geradores lineares congruentes.

Sequências pseudo-aleatórias: exemplos

- Uma função comumente usada por geradores de números aleatórios é uma combinação de uma função linear atuando sobre inteiros, seguida da operação de resto:

$$r_n = (ar_{n-1} + b) \bmod m \quad \text{ou} \quad r_n = (ar_{n-1} + b) \% m \quad (2)$$

- Em C, a biblioteca padrão `stdlib` contém a função `rand()` que usa um desses chamados geradores lineares congruentes.
- Note que ao tomar o resto da divisão por um inteiro m , tais métodos possuem um período limite de tamanho m . O tamanho real do período dependerá dos valores de a e b , sendo que para alguns valores desses inteiros, o limite m é de fato atingido.

Sequências pseudo-aleatórias: exemplos

- Uma função comumente usada por geradores de números aleatórios é uma combinação de uma função linear atuando sobre inteiros, seguida da operação de resto:

$$r_n = (ar_{n-1} + b) \bmod m \quad \text{ou} \quad r_n = (ar_{n-1} + b) \% m \quad (2)$$

- Em C, a biblioteca padrão `stdlib` contém a função `rand()` que usa um desses chamados geradores lineares congruentes.
- Note que ao tomar o resto da divisão por um inteiro m , tais métodos possuem um período limite de tamanho m . O tamanho real do período dependerá dos valores de a e b , sendo que para alguns valores desses inteiros, o limite m é de fato atingido.
- O tamanho máximo da sequência pseudo-aleatória gerada por `rand()` em `stdlib` pode ser acessado por meio da macro de nome `RAND_MAX`. Você pode verificar facilmente qual o valor de `RAND_MAX`.

Sequências pseudo-aleatórias: exemplos

- Uma função comumente usada por geradores de números aleatórios é uma combinação de uma função linear atuando sobre inteiros, seguida da operação de resto:

$$r_n = (ar_{n-1} + b) \bmod m \quad \text{ou} \quad r_n = (ar_{n-1} + b) \% m \quad (2)$$

- Em C, a biblioteca padrão `stdlib` contém a função `rand()` que usa um desses chamados geradores lineares congruentes.
- Note que ao tomar o resto da divisão por um inteiro m , tais métodos possuem um período limite de tamanho m . O tamanho real do período dependerá dos valores de a e b , sendo que para alguns valores desses inteiros, o limite m é de fato atingido.
- O tamanho máximo da sequência pseudo-aleatória gerada por `rand()` em `stdlib` pode ser acessado por meio da macro de nome `RAND_MAX`. Você pode verificar facilmente qual o valor de `RAND_MAX`.
- Assim, pode-se gerar uma variável aleatória no intervalo $[a, b]$ qualquer através de: $a + (b-a)*\text{rand()}/\text{RAND_MAX}$.

Sequências pseudo-aleatórias: sementes

- A fórmula (2) acima não define completamente a sequência de números. O aluno atento deve ter percebido que é preciso especificar o primeiro elemento da sequência: **A SEMENTE**.
- A escolha da semente é procedimento importante e se executada corretamente, aumenta o caráter "aleatório" da sequência gerada.
- A semente para a função `rand()` de `stdlib` pode ser fixa usando-se uma outra função da mesma biblioteca chamada `srand`. O protótipo dessa função no seu computador tem muito provavelmente a seguinte cara:

```
void  srand(unsigned);
```

indicando que `srand()` não retorna qualquer valor (é do tipo `void`), nem mesmo por referência, pois seu único argumento é do tipo `unsigned` e é passado por valor.

Sequências pseudo-aleatórias: sementes

- A fórmula (2) acima não define completamente a sequência de números. O aluno atento deve ter percebido que é preciso especificar o primeiro elemento da sequência: **A SEMENTE**.
- A escolha da semente é procedimento importante e se executada corretamente, aumenta o caráter "aleatório" da sequência gerada.
- A semente para a função `rand()` de `stdlib` pode ser fixa usando-se uma outra função da mesma biblioteca chamada `srand`. O protótipo dessa função no seu computador tem muito provavelmente a seguinte cara:

```
void  srand(unsigned);
```

indicando que `srand()` não retorna qualquer valor (é do tipo `void`), nem mesmo por referência, pois seu único argumento é do tipo `unsigned` e é passado por valor.

- Entretanto, uma chamada do tipo:

```
srand(seed);
```

faz com chamadas posteriores de `rand()`, utilizem como semente o valor passado por valor na variável `seed`.

- É comum que você deseje que a sequência de números geradas por `rand()` seja diferente a cada vez que o executável é posto para rodar, sem que o usuário tenha que fazer a escolha a cada vez. Uma maneira de fazer isso é usar a função `time` da biblioteca `time.h` para passar uma semente que depende da hora local indicada pelo relógio do seu computador:

```
srand ( time(NULL) );
```

Conservação de probabilidade e o método da transformação

- Já sabemos então como gerar sequências "aleatórias" uniformes, ou seja, com distribuição de probabilidade uniforme. Dito de outra forma, se x é a variável em questão, em particular, definida no intervalo $(0,1)$, a probabilidade de que ela se encontre entre x e $x + dx$ e denotada $p(x)dx$ é:

$$p(x) = \begin{cases} 1 & \text{se } 0 < x < 1 \\ 0 & \text{do contrário} \end{cases} \quad \text{com } \int_{-\infty}^{\infty} p(x)dx = 1 \text{ (normalização)} \quad (3)$$

Conservação de probabilidade e o método da transformação

- Já sabemos então como gerar sequências "aleatórias" uniformes, ou seja, com distribuição de probabilidade uniforme. Dito de outra forma, se x é a variável em questão, em particular, definida no intervalo $(0,1)$, a probabilidade de que ela se encontre entre x e $x + dx$ e denotada $p(x)dx$ é:

$$p(x) = \begin{cases} 1 & \text{se } 0 < x < 1 \\ 0 & \text{do contrário} \end{cases} \quad \text{com } \int_{-\infty}^{\infty} p(x)dx = 1 \text{ (normalização)} \quad (3)$$

- Agora podemos analisar o problema central envolvido em métodos de Monte Carlo que é o da geração de uma variável y com uma distribuição de probabilidade dada $p(y)$.

Conservação de probabilidade e o método da transformação

- Já sabemos então como gerar sequências "aleatórias" uniformes, ou seja, com distribuição de probabilidade uniforme. Dito de outra forma, se x é a variável em questão, em particular, definida no intervalo $(0,1)$, a probabilidade de que ela se encontre entre x e $x + dx$ e denotada $p(x)dx$ é:

$$p(x) = \begin{cases} 1 & \text{se } 0 < x < 1 \\ 0 & \text{do contrário} \end{cases} \quad \text{com } \int_{-\infty}^{\infty} p(x)dx = 1 \text{ (normalização)} \quad (3)$$

- Agora podemos analisar o problema central envolvido em métodos de Monte Carlo que é o da geração de uma variável y com uma distribuição de probabilidade dada $p(y)$.
- Suponha que x é uma variável com distribuição de probabilidade uniforme e que fazemos agir sobre essa variável uma função $y(x)$. A distribuição de probabilidade $p(y)dy$ da nova variável y fica determinada pela conservação de probabilidades:

$$|p(y)dy| = |p(x)dx| \quad (4)$$

Conservação de probabilidade e o método da transformação

- Já sabemos então como gerar sequências "aleatórias" uniformes, ou seja, com distribuição de probabilidade uniforme. Dito de outra forma, se x é a variável em questão, em particular, definida no intervalo $(0,1)$, a probabilidade de que ela se encontre entre x e $x + dx$ e denotada $p(x)dx$ é:

$$p(x) = \begin{cases} 1 & \text{se } 0 < x < 1 \\ 0 & \text{do contrário} \end{cases} \quad \text{com } \int_{-\infty}^{\infty} p(x)dx = 1 \text{ (normalização)} \quad (3)$$

- Agora podemos analisar o problema central envolvido em métodos de Monte Carlo que é o da geração de uma variável y com uma distribuição de probabilidade dada $p(y)$.
- Suponha que x é uma variável com distribuição de probabilidade uniforme e que fazemos agir sobre essa variável uma função $y(x)$. A distribuição de probabilidade $p(y)dy$ da nova variável y fica determinada pela conservação de probabilidades:

$$|p(y)dy| = |p(x)dx| \quad (4)$$

- Logo

$$p(y) = p(x) \left| \frac{dx}{dy} \right| = \frac{p(x)}{\left| \frac{dy}{dx} \right|} \quad (5)$$

Método da transformação: distribuição exponencial

- Vejamos um exemplo concreto...
- Tome $y(x) = -\ln(x)$ ($\implies x = e^{-y}$). Então

$$p(y) = p(x) \left| \frac{dx}{dy} \right| = \left| \frac{dx}{dy} \right| = e^{-y} \quad (6)$$

- Se gerarmos números aleatórios com distribuição uniforme, o negativo do logaritmo neperiano de tais variáveis seguirá uma distribuição de probabilidades dada por uma exponencial decrescente.

Método da transformação: distribuição exponencial

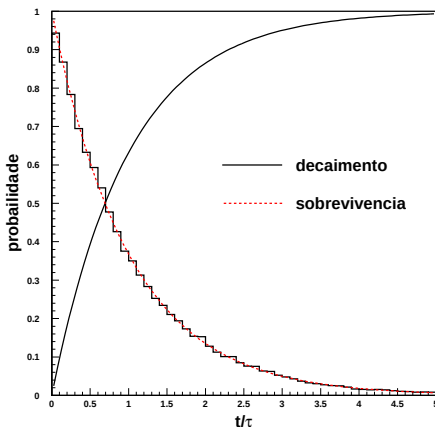
- Vejamos um exemplo concreto...
- Tome $y(x) = -\ln(x)$ ($\implies x = e^{-y}$). Então

$$p(y) = p(x) \left| \frac{dx}{dy} \right| = \left| \frac{dx}{dy} \right| = e^{-y} \quad (6)$$

- Se gerarmos números aleatórios com distribuição uniforme, o negativo do logaritmo neperiano de tais variáveis seguirá uma distribuição de probabilidades dada por uma exponencial decrescente.
- Distribuições desse tipo são comuns envolvendo decaimentos de partículas instáveis e núcleos radioativos em Física.
- Exemplo: A probabilidade de que uma núcleo instável decaia entre t e $t + dt$ é dada por $1 - e^{-\frac{t}{\tau}}$, onde τ é a chamada meia-vida e definida como o tempo necessário para que metade de uma amostra contendo inicialmente N núcleos tenha decaído, no limite de N grande.

Método da transformação: distribuição exponencial

```
float expdev(float tau)
/* Retorna uma variavel aleatoria ditribuida exponencialmente
   com media tau usando a funcao rand() da biblioteca stdlib.
   Adaptado de Numerical Recipes em C.
*/
{
    float dum;
    do
        dum=(1.*rand())/RAND_MAX;
    while (dum == 0.0);
    return -tau*log(dum);
}
```



Método da transformação: distribuição gaussiana (normal)

- O método da transformação pode ser generalizado para mais de uma dimensão. Se $x_1, x_2, \dots, x_n$ são variáveis com distribuição de probabilidade conjunta $p(x_1, x_2, \dots, x_n)dx_1dx_2\dots dx_n$ e $y_1, y_2, \dots, y_n$ são funções dessas variáveis, então a distribuição de probabilidades conjunta $p(y_1, y_2, \dots, y_n)dy_1dy_2\dots dy_n$ dos y 's é dada por

$$p(y_1, y_2, \dots, y_n)dy_1dy_2\dots dy_n = p(x_1, x_2, \dots, x_n) \left| \frac{\partial(x_1, x_2, \dots, x_n)}{\partial(y_1, y_2, \dots, y_n)} \right| dy_1dy_2\dots dy_n, \quad (7)$$

onde o determinante Jacobiano dos x 's com respeito aos y 's é dado por:

$$J(y_1, y_2, \dots, y_n) = \frac{\partial(x_1, x_2, \dots, x_n)}{\partial(y_1, y_2, \dots, y_n)} = \begin{vmatrix} \frac{\partial x_1}{\partial y_1} & \frac{\partial x_1}{\partial y_2} & \dots & \frac{\partial x_1}{\partial y_n} \\ \frac{\partial x_2}{\partial y_1} & \frac{\partial x_2}{\partial y_2} & \dots & \frac{\partial x_2}{\partial y_n} \\ \dots & \dots & \dots & \dots \\ \frac{\partial x_n}{\partial y_1} & \frac{\partial x_n}{\partial y_2} & \dots & \frac{\partial x_n}{\partial y_n} \end{vmatrix} \quad (8)$$

Método da transformação: distribuição gaussiana (normal)

- Uma aplicação do método multi-dimensional acima é o **algoritmo de Box-Muller** para a geração de variáveis com distribuição normal

$$p(y)dy = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{2\sigma^2}} dy \quad (9)$$

Método da transformação: distribuição gaussiana (normal)

- Uma aplicação do método multi-dimensional acima é o **algoritmo de Box-Muller** para a geração de variáveis com distribuição normal

$$p(y)dy = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{2\sigma^2}} dy \quad (9)$$

- Basta tomar 2 variáveis x_1 e x_2 uniformemente distribuídas no intervalo (0,1) e construir 2 outras y_1 e y_2 segundo

$$\begin{aligned} y_1 &= \sqrt{-2 \ln(x_1)} \cos(2\pi x_2) \\ y_2 &= \sqrt{-2 \ln(x_1)} \sin(2\pi x_2), \end{aligned}$$

de modo que é possível expressar x_1 e x_2 como função de y_1 e y_2

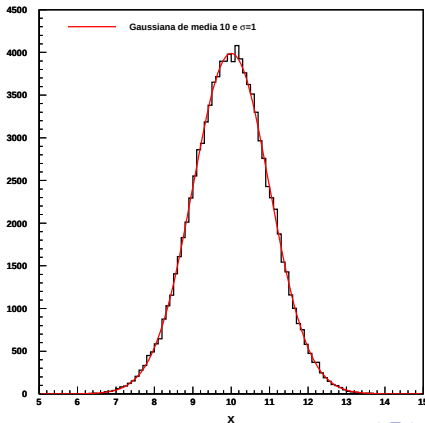
$$\begin{aligned} x_1 &= e^{-\frac{1}{2}(y_1^2 + y_2^2)} \\ x_2 &= \frac{1}{2\pi} \text{atan} \left(\frac{y_2}{y_1} \right). \end{aligned}$$

- O Jacobiano para essa transformação é:

$$J(y_1, y_2) = \frac{\partial(x_1, x_2)}{\partial(y_1, y_2)} = \begin{vmatrix} \frac{\partial x_1}{\partial y_1} & \frac{\partial x_1}{\partial y_2} \\ \frac{\partial x_2}{\partial y_1} & \frac{\partial x_2}{\partial y_2} \end{vmatrix} = -\frac{1}{\sqrt{2\pi}} e^{-\frac{y_1^2}{2}} \frac{1}{\sqrt{2\pi}} e^{-\frac{y_2^2}{2}} \quad (10)$$

Método da transformação: distribuição gaussiana (normal)

- O histograma abaixo mostra a distribuição de 100000 valores de uma variável gaussiana x gerada de acordo com o método de Box-Muller com média 10 e desvio padrão 1. Para gerar esse gráfico, apenas uma das variáveis (y_1 ou y_2) apresentadas na transparência anterior precisa ser usada.
- Para comparação, a distribuição gaussiana correspondente é superposta ao histograma.

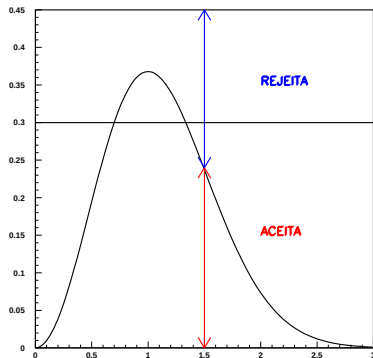


Método de aceitação-rejeição

- Os métodos apresentados anteriormente se baseiam no fato de que a matriz jacobiana da transformação possa ser calculada analiticamente.
- Há inúmeras situações em que isso não é possível, seja pelo fato que o número de variáveis envolvidas na transformação é elevado, seja por que as funções utilizadas na transformação são complicadas demais para serem expressas por uma combinação de funções elementares.

Método de aceitação-rejeição

- Os métodos apresentados anteriormente se baseiam no fato de que a matriz jacobiana da transformação possa ser calculada analiticamente.
- Há inúmeras situações em que isso não é possível, seja pelo fato que o número de variáveis envolvidas na transformação é elevado, seja por que as funções utilizadas na transformação são complicadas demais para serem expressas por uma combinação de funções elementares.
- Recorre-se nesses casos a um método bastante geral, baseado em um argumento geométrico simples, e que deve funcionar para praticamente todos os casos de interesse: o método de aceitação-rejeição.

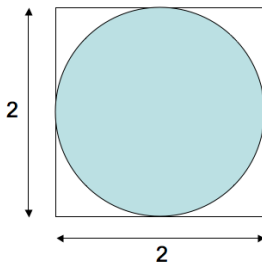


Método de aceitação-rejeição: algoritmo

- 1 Deseja-se gerar uma variável x com distribuição de probabilidade $p(x)dx$ no intervalo (a,b) .
- 2 Gera-se x uniformemente no intervalo de interesse e calcula-se $p(x)$.
- 3 Gera-se uma segunda variável y também uniforme, mas agora no intervalo $(0, y_{max})$, onde y_{max} deve satisfazer $y_{max} \geq p_{max}$, onde p_{max} é o valor máximo de $p(x)$ em (a,b) .
- 4 Se $y \geq p(x)$, rejeita-se o valor de x e gera-se um novo valor, senão, aceita-se o valor de x .
- 5 Os valores de x assim aceitos seguem a distribuição de probabilidade $p(x)$.

PARA FAZER EM SALA DE AULA

- 1 A razão entre a área de um círculo de raio 1 inscrito em um quadrado de lado 2 é $\frac{\pi}{4}$.
Escreva um programa que empregue uma sequência pseuso-aleatória e a idéia básica por trás do método de aceitação-rejeição para calcular π com precisão de 4 casas decimais.
Compare o valor obtido pelo seu programa com a aproximação de π obtida com o auxílio da função `atan()` (arco-tangente) da biblioteca matemática `math.h`.



PARA FAZER EM SALA DE AULA

- 2 As coordenadas polares θ e ϕ , para uma distribuição uniforme de pontos sobre uma esfera de raio R devem ter as seguintes distribuições de probabilidades:

$$p(\theta)d\theta = \sin \theta d\theta$$

$$p(\phi)d\phi = d\phi,$$

com $0 \leq \theta \leq \pi$ e $0 \leq \phi < 2\pi$, dado que a densidade de pontos uniformemente distribuídos sobre uma área infinitesimal em torno da direção (θ, ϕ) sobre a esfera deve ser proporcional ao ângulo sólido $d\Omega = \sin \theta d\theta d\phi$ subentendido por essa área.

- Escreva um programa para gerar pontos uniformemente distribuídos sobre a esfera unitária. No caso da variável θ , utilize o método da transformação para gerá-la. Qual deve ser a função $y(x)$ a ser aplicada nesse caso?
- Seu programa deve escrever as coordenadas cartesianas dos pontos gerados num arquivo de saída:

$$x = \sin \theta \cos \phi$$

$$y = \sin \theta \sin \phi$$

$$z = \cos \theta$$

- Teste o funcionamento do seu programa utilizando o script `scatter.dem`.

PARA FAZER EM SALA DE AULA

