MPEI - Métodos Probabilísticos para Engenharia Informática (2015/2016) **PL 03**

Palavras-chave: geração de números aleatórios, geração de variáveis aleatórias, simulação de experiências aleatórias

1 Geração de números aleatórios

Consulte a informação das aulas Teórico-Práticas e responda às questões seguintes através de pequenos programas em Octave/Matlab:

1. Implemente uma função que gere um vector de números aleatórios com base no método da Congruência.

   A função deve permitir controlar o seu comportamento através dos parâmetros necessários à definição da fórmula usada neste método e também permitir definir o comprimento do vector a gerar. Sugestão: `function y = lcg (X0,a,c,m,N).`

2. Utilizando a função anterior:

   (a) gere um vector de comprimento 1000 usando parâmetros à sua escolha e visualize o seu histograma. Quantos números diferentes obteve? Sugestão: experimente a função `unique()`.

   (b) gere um conjunto de números aleatórios entre 0 e 1 com base nos da alínea a) . Visualize o histograma e conte novamente os números diferentes.

   (c) repita as alíneas anteriores com os parâmetros utilizados na implementação incluída na biblioteca NAG.

3. Por questões de eficiência passemos a usar a função Matlab/Octave `rand()` , que permite de uma forma simples obter os números aleatórios que obtivemos nos pontos anteriores. Com base nesta função:

   (a) Simule uma sequência de 10 experiências de Bernoulli com probabilidade de sucesso p. Represente sucesso por 1 ;

   (b) Simule 15 lançamentos de 1 dado (honesto);

   (c) Obtenha um conjunto de 20 números reais igualmente distribuídos entre -4 e 10;

4. Crie uma função que gere um vector de números com uma distribuição de Bernoulli com parâmetro p.

   Utilize o histograma (função `hist()`) e a estimativa das probabilidades com base num vector gerado para testar o seu funcionamento.

   Exemplo de teste: hist(Bernoulli(.3, 10000),(0:1)')

5. Crie uma função que gere um vector de números com uma distribuição Binomial, dados os parâmetros deste tipo de distribuição(n e p) e N (tamanho do vector a gerar).

   Utilize o histograma e a estimativa das probabilidades com base num vector gerado para testar o seu funcionamento. Tente comparar as probabilidades obtidas com base no vector gerado com os valores teóricos.

6. Crie uma função que gere um vector de números com uma distribuição discreta genérica definida pela sua pmf.

   A sua função deve receber como parâmetros de entrada dois vectores definindo a pmf, xi e pX, assim como o número de valores a gerar.

   Utilize o histograma e a estimativa das probabilidades com base num vector gerado para testar o seu funcionamento.

7. Crie uma função que gere um vector de números com a distribuição normal utilizando o Método de Box-Müller.

   Usando esta função como base, gere as classificações de uma turma de 30 alunos por forma a terem média 14 e variância 2.

   Confirme os resultados com a função Matlab `randn()` . Consulte a informação sobre esta função.

8. Repita o exercício anterior criando uma nova versão da função geradora baseada no método da rejeição. Assuma que os valores de $f_X(x)$ da função Gaussiana normalizada é próxima de zero fora do intervalo (-5,5).

   Use o histograma para verificar as características principais (forma, média e dispersão em torno da média).

9. Utilize o método da transformação para implementar uma função geradora de números com uma distribuição exponencial. Use exponencial para o nome dessa função.

   Teste a função, usando, por exemplo `hist(exponencial (10000,.5),20).`

10. Crie uma função que permita simular a retirada aleatória de um subconjunto de bolas de uma urna, sem reposição.

    Aplique a função ao caso do Totoloto.

2 Exemplo de Aplicação - Simulação do comportamento de uma Chaining Hash Table

O objectivo deste exercício é avaliar o funcionamento de uma estrutura de dados importante, a *Chaining Hash Table*, e um dos conceitos que a suportam, *Hash Functions* (funções de dispersão). Em Programação II foi apresentada alguma informação sobre as funções de dispersão sem entrar em detalhes, nomeadamente que "o desempenho da tabela de dispersão depende da capacidade da função de hash para distribuir uniformemente as chaves pelos índices do array" e que "uma análise estatística da distribuição das chaves pode ser considerada". Neste exercício, usando geração de chaves de forma aleatória iremos efectuar a análise da distribuição das chaves.

Começaremos por criar os componentes essenciais para efectuar uma primeira simulação bastante simplificada. De seguida tornaremos um pouco mais realista.

- Primeira versão:

  1. Crie uma função que gere uma chave (string) com comprimento aleatório entre 3 e 20 assumindo uma distribuição uniforme (discreta) e em que as letras (apenas minúsculas e maiúsculas) são equiprováveis.
  2. Implemente uma função de hash simples. Sugestão: Adopte a função de dispersão hashstring() adoptada em Programação II.
  3. Simule a inserção de 1000 das strings criadas pela função que desenvolveu numa Chaining Hash Table (que como se deve recordar usa internamente um array e listas ligadas). Como para a nossa simulação não necessitamos guardar as chaves e valores a elas associados, apenas necessitamos do array. Adopte para tamanho do array um valor que implique um factor de carga elevado (ex:0.8) .

     Guarde informação sobre o número de chaves que foram mapeadas numa determinada posição do array até esse momento.

     Durante a simulação, visualize, em gráficos separados:

     - o número de strings que foram mapeadas pela hash function para cada posição;

     - o histograma desses números
  4. Usando a informação sobre o número de chaves que foram mapeadas para cada posição do array após a inserção de todas as chaves, estime e represente graficamente a função de distribuição para a variável aleatória X definida como o número de chaves mapeadas para uma posição. Qual seria o valor médio do comprimento das listas ligadas neste caso ? Qual a variância ?

- Segunda versão:

    1. Crie uma nova versão da função de geração das chaves assumindo que o tamanho das chaves segue uma distribuição normal (com média 10 e variância 5) e as letras seguem a distribuição das letras em Português.

       Nota: Utilize o script disponibilizado no material de apoio à UC para efectuar a estimativa dessa distribuição.
    2. Repita a simulação com 1000 chaves e os cálculos da última alínea da versão 1.

- Terceira versão (opcional):

    1. Altere a função de hash na segunda versão e repita a simulação e cálculos.

- Questões finais:

  A função de hash conseguiu o objectivo de espalhar as chaves/strings pelo array ? Temos de facto uma distribuição uniforme das chaves pelos índices do array ?