Exercício 01: Gerador de números aleatórios

Francisco Cerdeira*

Neste trabalho foi implementado um gerador de números pseudo-aleatórios com recurso ao método congruente, tendo sido testada, posteriormente, a qualidade deste. Foram também gerados, aleatoriamente e de maneira uniforme, pontos limitados ao interior de um circulo.

Implementação de um gerador linear congruente

- O método linear congruente gera uma sequencia de números pseudo-aleatórios através da equação linear

$$X_{n+1} = (cX_n + a) \mod p \tag{1}$$

onde X_n é a sequencia de números, c é o multiplicador, a é o incremento e m é o numero máximo que X_n pode tomar. Deve ser ainda definida a semente, X_0.

Tendo em conta os parâmetros usados é possível separar o código criado em duas funções distintas: uma primeira na qual foram usados valores pequenos para c e p onde se pretendia fazer notar alguns dos defeitos inerentes ao método; e uma segunda em que foram usados valores semelhantes aqueles propostos na aula teórica e que, em principio, permitem ocultar essas lacunas.

Em baixo encontram-se os parâmetros que distinguem os 2 geradores criados:

  - 1º Gerador (Exercício 1.1 i.)
    - $c = 3$
    - $p = 31$
    - $X_0 = 7$
  - 2º Gerador (Exercício 1.1 iii.)
    - $c = 16807$
    - $p = 2^{31} - 1$
    - $X_0 = 524287$

As semelhanças entre estes são que ambos não apresentam qualquer incremento, $a = 0$, e, que ambos geram um igual número de pontos, $N = 1000$.

- De modo a concluir sobre a qualidade dos números gerados, ou seja, se os números gerados são aleatórios ou se pelo contrário, números consecutivos estão fortemente correlacionados, foi usado o teste do m-cubo para 2 e 3 dimensões. Para além dos 2 geradores criados e descritos em cima, foram também sujeitos a este teste as funções *rand()* e *drand48()*, implementadas em **C++** que irão servir como controlo.

- Os 2 gráficos seguintes apresentam o teste m-cubo, a 3 dimensões, para os dois geradores criados.

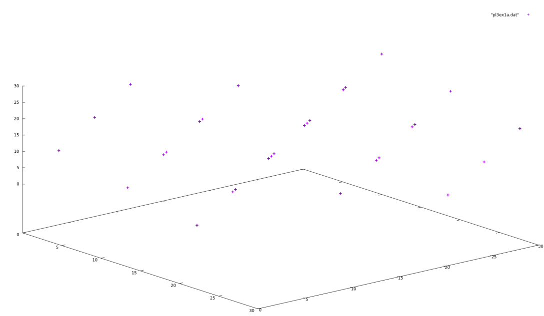

Figura 1. Usando os parâmetros descritos em cima para o Gerador 1 foi possível criar este gráfico. Como é possível observar os diversos pontos formam planos bem definidos o que indica uma fraca qualidade do algoritmo

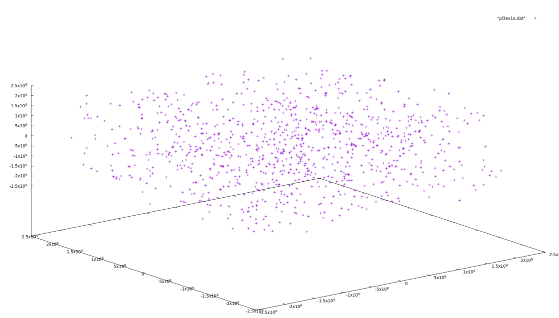

Figura 2. Aqui é apresentado o resultado para o 2º Gerador, ao qual foram alterados diversos parâmetros. Neste caso a maior alteração resulta da diferença de números que podem ser gerados, passando de $p = 31$ para $p = 2^{31} - 1$

Do teste do m-cubo para 3 dimensões observa-se, para o 1º gerador, um padrão bem definido, que indica uma fraca qualidade deste. Isto era expectável pois o número máximo que X_n pode tomar é bastante pequeno, o que aliado à semente baixa, leva a uma repetição dos números gerados, algo facilmente observado no gráfico 1. Quanto ao 2º gerador, figura 2 este apresenta uma grande semelhança aos implementados em C++, sem padrões observáveis, o que leva a crer que este é um bom algoritmo, e visto ter sido implementada a mesma função, 1, a

* francisco.cerdeira@gmail.com

sua qualidade está fortemente dependente dos parâmetros usados.

Gerar pontos uniformemente num círculo

- Nesta secção serão gerados, aleatoriamente e de forma uniforme, pontos limitados pela circunferência de raio R. Apesar de a implementação de uma das funções criadas na secção anterior ser uma possibilidade, optou-se aqui pelo uso da função *rand()*.

  De modo a conter os pontos dentro do circulo foram usadas coordenadas polares (r, θ) o que permitiu definir com facilidade a distancia que um ponto se pode afastar da origem, $R = 1$, bem como o ângulo que este pode fazer com esta, $\theta \in [0; 2\pi]$.

  Menos simples de garantir é a uniformidade da distribuição. Uma distribuição uniforme do número de pontos ao longo do raio, R, levará a um excesso de pontos junto da origem. É necessário compensar o numero de pontos à medida que nos afasta-mos do centro com uma função de densidade de probabilidade, $f(r)$, diretamente proporcional a r.

$$f(r) = C \times r \Leftrightarrow \int_{i=0}^{R} f(r) = \int_{i=0}^{R} C r dr \Leftrightarrow \Leftrightarrow C = \frac{2}{R^2} \tag{2}$$

  De forma a encontrar a função de probabilidade acumulada é necessário integrar $f(r)$

$$\int f(r) = \int \frac{2r}{R^2} dr \Leftrightarrow F(r) = \frac{r^2}{R^2} \tag{3}$$

  Assim as equações usadas para as coordenadas polares dos pontos ficam

$$r = R\sqrt{\frac{rand()}{RAND\_MAX}} \tag{4}$$

$$\theta = 2\pi \frac{rand()}{RAND\_MAX} \tag{5}$$

  sendo depois convertidas para coordenadas cartesianas

- Na figura 3 é apresentado o resultado do código descrito em cima

Teste de χ^2

- Nesta secção irá ser implementado uma função com o objetivo de testar os dois geradores de números

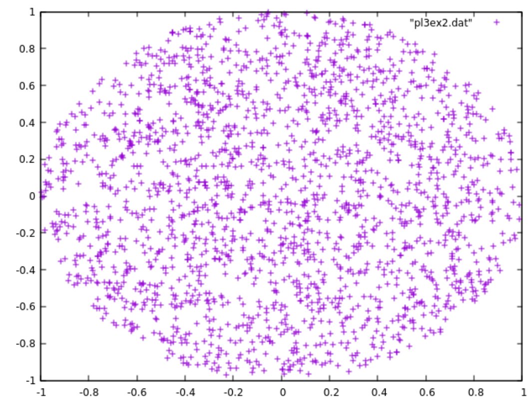

Figura 3. Circulo com uma unidade de raio com pontos gerados uniformenente e sem rejeição pela função *rand()*. Usaram-se coordenadas polares com $R = 1$ e com $\theta \in [0, 2\pi]$

pseudo-aleatórios criados na primeira secção. À semelhança do que acontece para o teste do m-cubo, o teste do χ^2 é usado como forma de testar a independência dos números gerados e deste modo, avaliar a qualidade do algoritmo.

De modo a implementar o teste do χ^2, definido pela equação 6, criou-se um histograma que abrangia o intervalo de números gerados, $[0; 1]$, definindo-se o tamanho de cada coluna como $\frac{1}{k}$, com $k = 20$. O histograma em si não é importante e deste modo não é mostrado, apenas foi usado como forma de encontrar N_i para os diferentes intervalos, i, e assim implementar o teste do χ^2. Foi então necessário contar quantos números caíam nos diferentes intervalos. Daqui facilmente se implementa o teste.

- Parâmetros para o Teste de χ^2
  - $k = 20$
  - $n = 200$
  - $p = 0.05$
  - $np_i = 10$

$$\chi^2 = \sum_{i=0}^{k} \frac{(N_i - np_i)^2}{np_i} \tag{6}$$

- Os valores obtidos são de $\chi^2_{obs_1} = 7$ e $\chi^2_{obs_1} = 62$ para o gerador 1 e 2, respetivamente. Deste modo, e com recurso a uma tabela, compara-se o resultado obtido para $\nu = k - 1 = 19$, concluindo-se que para para um nível de significância, $p{-}value = 0.1$ temos que $\chi^2_{\nu, p-value} = 27.2036$, e deste modo, como $\chi^2_{obs_1} < \chi^2_{\nu, p-value}$, os números obtidos com o Gerador 1 não se podem considerar aleatórios. No caso do gerador 2, visto que $\chi^2_{obs_2} > \chi^2_{\nu, p-value}$, os seus valores são aleatórios com um nível de significância $p{-}value = 0.1$.