Gustavo Estrela de Matos

MAC 110 – EP2

Números pseudo-aleatórios

IME-USP

São Paulo

2013

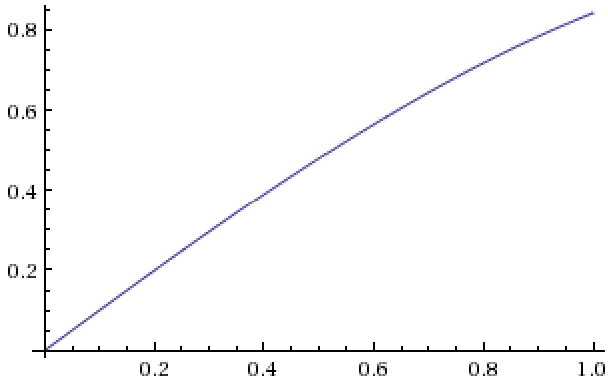
Objetivo

Esse trabalho tem como objetivo discutir e analisar um algoritmo capaz de gerar uma sequência de números pseudo-aleatórios a partir de uma função que inclua alguma aproximação calculada a partir da expansão da série de Taylor.

Formula utilizada

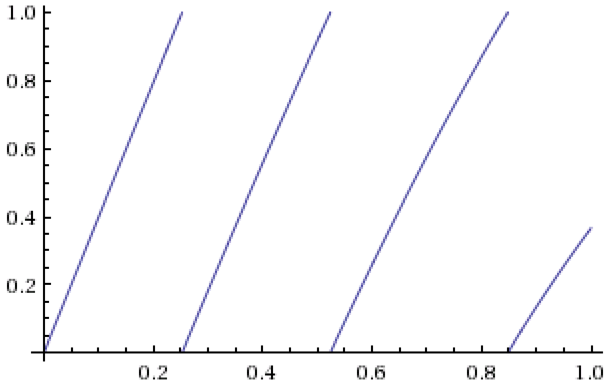
$$\boldsymbol{X_{n+1}} = \boldsymbol{frac}((\mathbf{7^4} * (\mathbf{7} + \boldsymbol{X_n})) * \boldsymbol{modulo}(\boldsymbol{sen}(\boldsymbol{X_n})))$$

A formula acima foi a conclusão após observações da função seno com argumento variando entre 0 e 1:

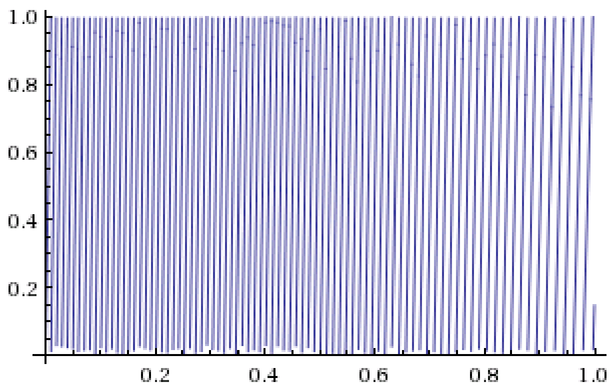


1- Função seno com argumento variando de 0 a 1.

Se multiplicarmos a função por um número qualquer que limite a sua parte fracionária temos o seguinte:

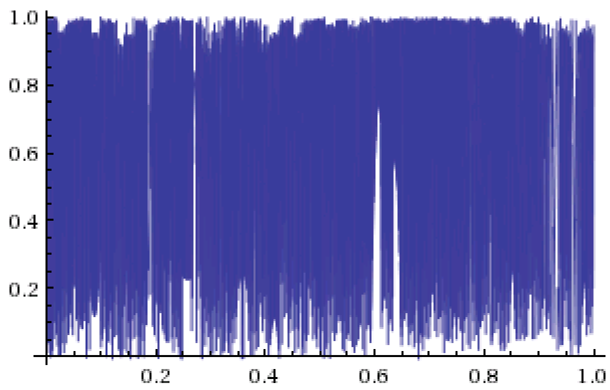


2 - Parte fracionária de 2*(sen(x)) com x variando de 0 a 1



3 – Parte fracionária de 100*(sen(x)) com x variando de 0 a 1

Portanto, quanto maior o fator multiplicador do sen(x) maior é a variação de f(x) para certa variação de x. Isso faz com que, se multiplicarmos sen(x) por um valor muito alto, f(x) será algo imprevisível para qualquer x. Vejamos graficamente:



4 – Parte fracionária de $7^5 * sen(x)$ com x variando de 0 a 1.

Para aumentar a aleatoriedade da função e, também o número que multiplica o seno, foram adicionadas mais casas decimais a uma das parcelas que multiplica sen(x) somando x a um dos fatores de 7^5 assim:

$$\boldsymbol{(7^4 * (7 + x)) * sen(x)}$$

Como semente, foi utilizado um número qualquer, 0."meucpf", pois, como dito acima, o importante para que a função retornasse valores pseudo-aleatórios era uma constante alta e com várias casas, e a semente em si não deveria ter muita influência no resultado.

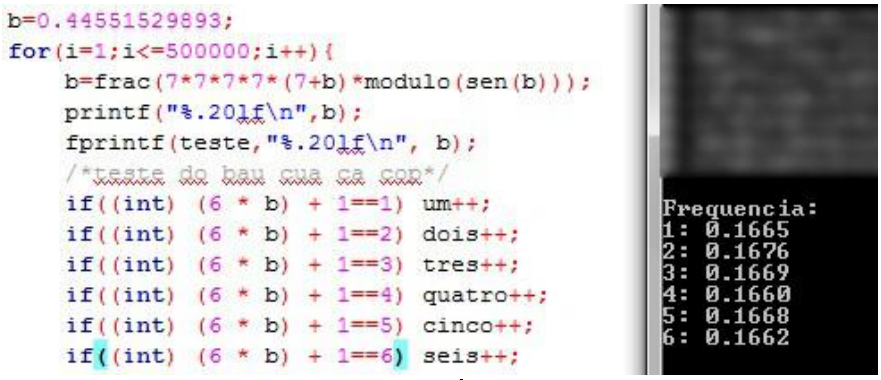
1. Testes Realizados

2.1 Distribuição uniforme

A sequência produzida por um gerador de números pseudo-aleatórios deve possuir uma distribuição uniforme de valores gerados. Para testar o código foram realizados os seguintes testes:

- **Bau cua ca cop**

  Neste teste são gerados *n* números que geram outro número de 1 a 6 seguindo a fórmula: *numero = (int) (6 * X) + 1.* Na qual, com uma boa distribuição, cada valor de 1 à 6 deve possuir frequência de 1/*n*.

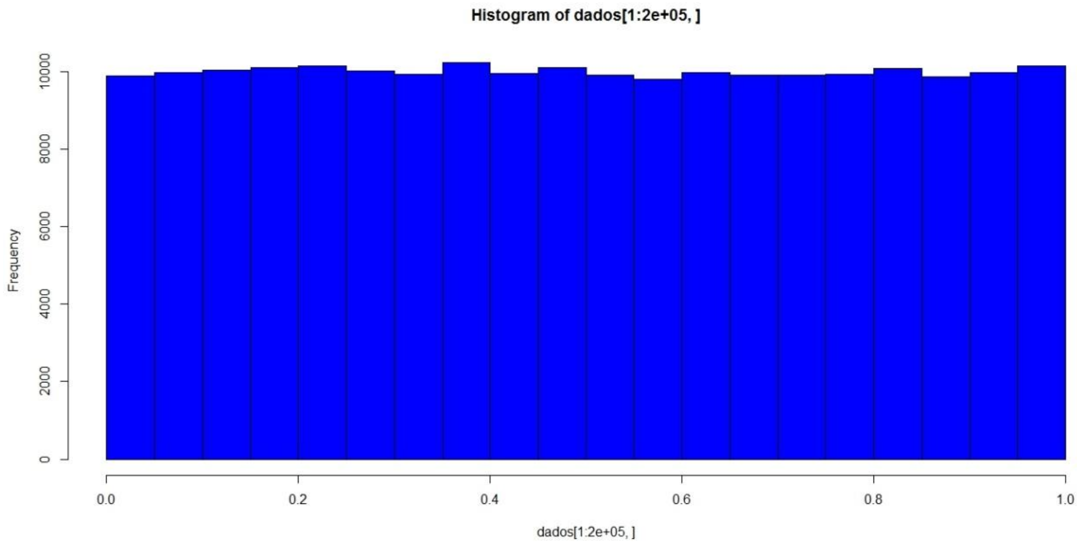


```
b=0.44551529893;
for(i=1;i<=500000;i++){
    b=frac(7*7*7*7*(7+b)*modulo(sen(b)));
    printf("%.20lf\n",b);
    fprintf(teste,"%.20lf\n", b);
    /*teste do bau cua ca cop*/
    if((int) (6 * b) + 1==1) um++;
    if((int) (6 * b) + 1==2) dois++;
    if((int) (6 * b) + 1==3) tres++;
    if((int) (6 * b) + 1==4) quatro++;
    if((int) (6 * b) + 1==5) cinco++;
    if((int) (6 * b) + 1==6) seis++;
```

```
Frequencia:
1: 0.1665
2: 0.1676
3: 0.1669
4: 0.1660
5: 0.1668
6: 0.1662
```

5 - Código e resultado do teste bau na cua cop

  Como podemos observar na imagem 1, a sequência de números gerados a partir da semente “0.44551529893”, que no caso é simplesmente um número no formato 0.“cpf qualquer”, teve uma distribuição boa porque a frequência relativa de números entre 1 e 6 foi de 16%.



6- Histograma com a frequência dos valores gerados

- **Montecarlo**

Montecarlo é um método para calcular áreas de gráficos. Podemos usar esse método para saber se a sequência gerada tem distribuição uniforme, pois esse método consiste em distribuir pontos aleatórios e calcular quantos estão ou não dentro da área determinada por uma função.

```
b=0.44551529893;
for(i=1;i<=500000;i++){
    b=frac(7*7*7*7*(7+b)*modulo(sen(b)));
    printf("%.20lf\n",b);
    fprintf(teste,"%.20lf\n", b);
    if(i%2==0){
        fprintf(yps,"%.20lf\n", b);
        y=b;
        abaixo+=montecarlo(x,y);
    }else{
        fprintf(xis,"%.20lf\n", b);
        x=b;
 int montecarlo(double x, double y){
}    if(y<=sqrt(2*x-x*x)){
         return 1;
     }
     return 0;
 }
```

tecarlo

7 - Função montecarlo()

Como podemos observar na imagem 2 e 3, a cada dois números gerados pelo algoritmo, um ponto é formado, e somando a quantidade de pontos abaixo do gráfico "$f(x)=2x-x^2$" e dividindo esse valor pela quantidade de pontos, o valor resultante deve ser próximo a área do gráfico de f(x) que no caso é π/4:

```
0.44631946503795916000
0.27590042973315576000
0.90911239854722226000
0.15079165736096911000
0.14946580525293029000
0.16785848168456141000
0.30205620446668036000
0.55905809828436759000

Frequencia:
1: 0.1665
2: 0.1676
3: 0.1669
4: 0.1660
5: 0.1668
6: 0.1662

--------------------------------------

Area do grafico: 0.7851
```

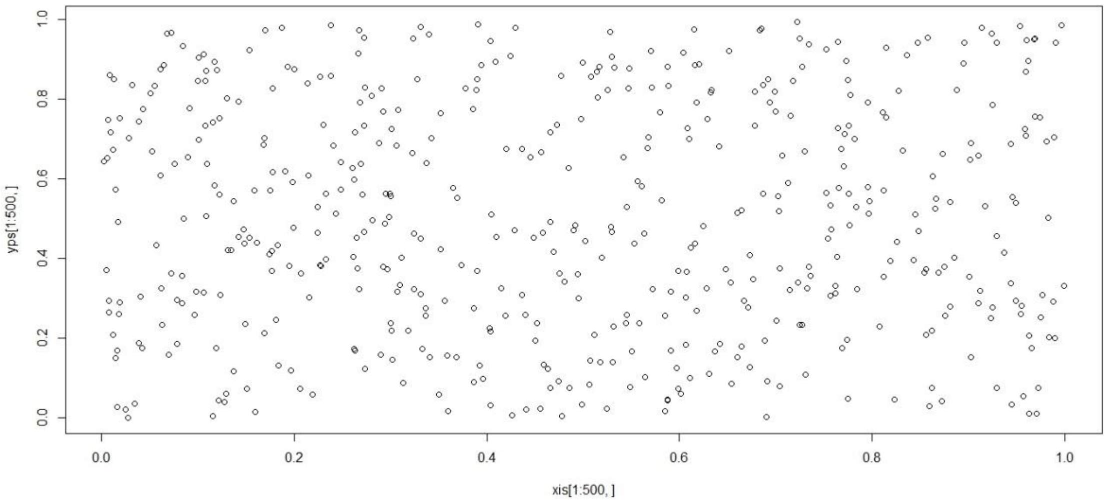
8 - Area aproximada do grafico de $2x-x^2$ calculada pelo algoritmo usando o metodo de Montecarlo

2.2 Independência

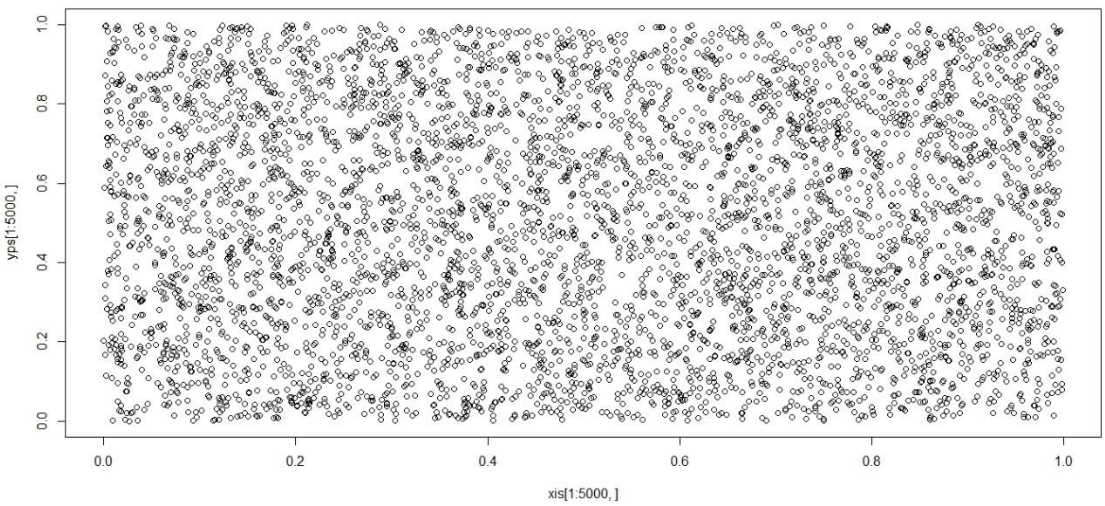
Os testes passados como “Bau na cua cop” e “Monte Carlo” são capazes de dizer se a sequência de números gerados é distribuída uniformemente, mas isso não garante que os números sejam realmente pseudo-aleatórios. Por exemplo, se fosse utilizada uma sequência como 0.1, 0.2, 0.3 ... 0.9, 0.1,... a distribuição seria boa, porém obviamente não aleatória. Por isso são necessários outros testes.

Esses testes devem garantir a independência dos números gerados, ou seja, se olharmos para os números antecessores e sucessores, não deve haver uma sequência lógica que nos permita, a partir de qualquer dos números gerados, saber qual será seu sucessor ou antecessor.

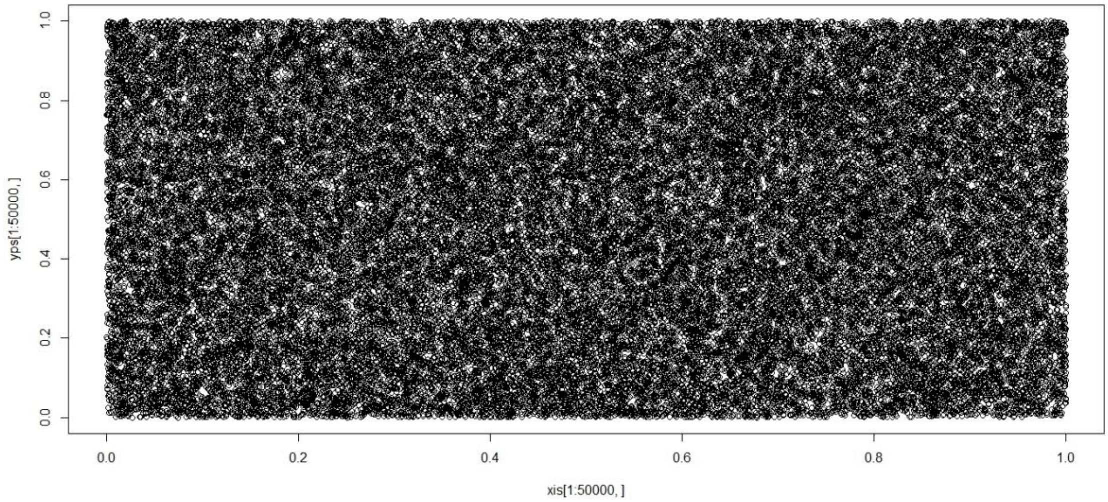
Através de softwares de estatística é possível representar graficamente os pontos formados no algoritmo, e podemos observar que a dispersão dos pontos não tem nenhum tipo de sequência lógica, pois quanto mais pontos gerados, mais “cheio” fica a representação e também não há “buracos”.



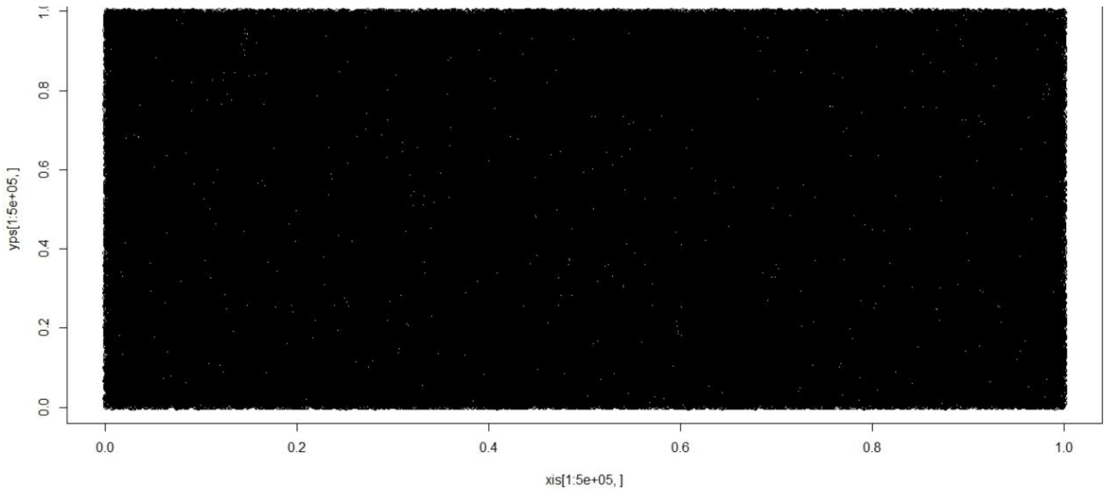
9 - Representação dos primeiros 500 pontos



10 - Representação dos primeiros 5000 pontos



11 – Representação dos primeiros 50000 pontos



12 – Representação gráfica dos primeiros 500000 pontos

1.3 Período

Para que seja aleatória, a sequência de números gerados também não deve se repetir após um número baixo de passos. Na fórmula utilizada, após serem gerados um milhão de números, a sequência não se repetiu em nenhum momento, e isso foi testado usando um arquivo com todos os números gerados e o artifício de procura que é presente quase na maioria dos editores de texto.

Também podemos observar que o período também é maior que 500 mil, pois, como observado na imagem 12, não há espaços em branco, o que significaria que os pontos deixaram de ser diferentes e começaram a se sobrepor.

Referências

VIEIRA, Carlos E. C.; RIBEIRO, Celso C. Geradores de Números Aleatórios. PUC-RIO. Junho, 2004. Disponível em:<ftp://ftp.inf.puc-rio.br/pub/docs/techreports/04_22_vieira.pdf> Acesso em: Abril de 2013.

ROGGIA, Iria B. Modelagem e Simulação. Junho, 2007. Disponível em: <http://www.infovisao.com/elc132/> Acesso em: Abril de 2013.

SOUZA, Críston. Geração de Números Aleatórios. Disponível em: <http://www.univasf.edu.br/~criston.souza/simulacao/arquivos/6-Numeros_aleatorios.pdf> Acesso em: Abril de 2013.

ZUBEN, Fernando V.; CASTRO, Leandro. Geradores de Números Aleatórios(GNA). Disponível em: <ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/ia707_02/topico7_02.pdf> Acesso em: Abril de 2013.

FUENTE, Pablo. Generación de números aleatórios. Disponível em: <http://jair.lab.fi.uva.es/~pablfue/leng_simulacion/materiales/alea_0506_resumen.pdf> Acesso em: Abril de 2013.

Autor Desconhecido. Simulação: geração de números pseudo-aleatórios. Disponível em: <http://web.ist.utl.pt/ist11038/acad/or/simul/GerNumAleat.pdf> Acesso em: Abril de 2013