5. APLICAÇÕES

5.1. MÉTODOS DE MONTE CARLO

Define-se método de Monte Carlo como qualquer técnica para a resolução de um modelo, científico ou não, com o uso de números pseudo-randômicos.

Os métodos de Monte Carlo apareceram quando a teoria da probabilidade e processamentos de acaso foram usados para solucionar problemas que não possuíam aspectos estocásticos, isto é, que não podem ser calculados por meio de fórmulas. Um exemplo muito simples dessa situação seria estimar a superfície de um lago irregular.

Tais metodos também foram estudados e pesquisados durante a II Guerra Mundial, a qual proporcionou-os grande popularidade na sociedade científica. Um dos principais pesquisadores que trabalhou com os números pseudo-randômicos para o projeto da bomba atômica foi John von Neumann. O estudo matemático foi necessário durante a fase de pesquisa da bomba, a qual se fez no prédio Gama em Los Alamos nos EUA. Os cientistas precisavam saber a distância que os neutrons atravessavam os diversos tipos de materiais e a solução para essa questão era essencial para o projeto da bomba funcionar; então, von Neumann e S. Ulam sugeriram o uso dos métodos de Monte Carlo, que empregavam números randômicos, e com a técnica foi possível resolver o problema e terminar o projeto com muito mais agilidade.

Estudaremos alguns usos dos métodos de Monte Carlo com exemplos simples na simulação de problemas matemáticos-computacionais. Um exemplo para se usar os métodos de Monte Carlo seria para resolver problemas de áreas: Achar a área de uma circunferncia incrita em um quadrado unitário e a partir desse resultado pode-se também estimar qual deve ser o valor de π. O que se deve observar é que este problema que pode ser resolvido com fórmulas matemáticas, aqui ele está ilustrando que o método pode também ser utilizado em problemas desse tipo, e neste caso, apenas para avaliar o erro e a eficiência do método. É muito importante que o usuário tente resolver o problema sem o uso dos métodos de Monte Carlo, pois os resultados fornecidos por fórmulas matemáticas são muito mais eficientes.

Os números randômicos que foram estudados anteriormente podem ser utilizados para as mais diversas aplicações junto aos métodos de Monte Carlo e eis algumas aplicações: simulação de fenômenos naturais, amostragem, análise matemática, teoria da probabilidade e recreação; cada uma dessas aplicações serão estudadas e exemplificadas. Para usarmos os números pseudo-randômicos usaremos o gerador do Maple ou o nosso gerador para os exemplos.

```
> random:=proc(m::posint,a::nonnegint,c::nonnegint)
> local RANDOMp;
> global _m1,_rsem;
```

```
> RANDOMp:=proc(x,y,z)
> local _modulo,_multip,_increm,aux;
> global _sement,_rsem,_m1;
> _modulo:=x;
> _multip:=y;
> _increm:=z;
> if not(assigned(_sement)) then
>  _sement:=modp(iolib(25),_modulo);
> fi:
>   _rsem:=`false`;
>   _m1:=_modulo;
>   _sement:=(_multip*_sement+_increm) mod (_modulo);
>   RETURN(_sement);
> end:
> if a>=m then ERROR(`O multiplicador fora do intervalo
  [0,módulo[`);
> elif c>=m then ERROR(`O incremento fora do intervalo
  [0,módulo[`);
> elif nargs>3 then ERROR(`Random só aceita 3 argumentos`);
> elif (_rsem=`true`) and (nargs>=0) and (_sement>=m) then
  ERROR(`A semente escolhida está fora do intervalo
  [0,módulo[`);
> else
> RANDOMp(m,a,c);
> fi:
> end:
>
```

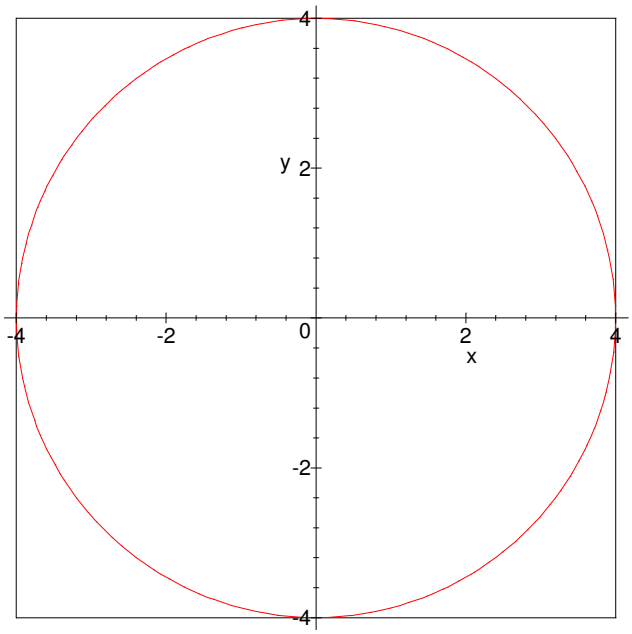
5.2. APLICAÇÕES ELEMENTARES

```
>
```

5.3. SIMULAÇÃO DE RESULTADOS MATEMÁTICOS

Num primeiro exemplo, vamos calcular o valor de π probabilisticamente a partir do seguinte problema:

Suponhamos que uma circunferencia de raio d esteja inscrita dentro de um quadrado de lado 2 d como na figura abaixo e queremos calcular a sua área e a partir dela calcular o valor do número π:



Para calcular a área da circunferência acima, nós podemos lançar pontos aleatórios dentro do quadrado acima e com isso estimar qual será a área, pois a área é a razão entre os pontos que caem dentro da circunferência e o número total de pontos que foram lançados. Para isso, precisamos de um gerador de números pseudo-randômicos que gere números uniformemente distribuídos.

Para simplificar o nosso problema, vamos calcular a área do primeiro quadrante, pois assim o problema é simplificado.

Vamos supor que $d = 1$, então, basta calcular pontos entre zero e um para lançarmos dentro do quadrado, e cada ponto que cair dentro da circunferência será contado

```
> n:=0:
> for i to 10000-1 do
> a:=rand():                Estamos usando o gerador do Maple neste exemplo
> b:=rand();
> if ((a/10^12)^2+(b/10^12)^2)<=1 then
> n:=n+1:
> fi:
> od:
```

```
> n;        Número de pontos que caíram dentro da circunferência
```

7739

```
> i;        Número de pontos lançados
```

10000

```
> evalf(4*n/i);      Valor aproximado de Pi
```

3.095600000

```
> erro:=evalf(Pi-");
```

erro := .045992654

```
>
```

TEOREMA DE CÉSARO

O teorema de Césaro descrito abaixo, se refere ao cálculo da probabilidade entre números primos. Vamos estimar o valor de π utilizando uma técnica de Monte Carlo.

TEOREMA DE CÉSARO: Tomando-se aleatoriamente dois inteiros positivos a probabilidade de que sejam primos entre sí é $\frac{6}{\pi^2}$.

Para verificar o teorema acima, nós obtemos dois números pseudo-randômicos e então verificamos se eles são primos entre sí pelo programa abaixo, e contamos a quantidade de números que são primos entre sí e assim podemos calcular a estatística.

```
> n:=0:                          Contador
> for i to 100000 do
> a:=rand():
> b:=rand():
> if  igcd(a,b)=1 then           Condição
> n:=n+1:
> fi:
> od:
> n;                             Quantidade de números que são primos entre sí
```

60817

```
> evalf(n/100000);               Dá o valor da probabilidade calculada
```

.6081700000

```
> evalf(6/(Pi^2));               Valor real da probabilidade como sugerida por Césaro
```

.6079271016

```
> evalf(abs("-""));              Erro
```

.0002428984

Observe que o erro está na casa dos décimos-milésimos. O que é uma boa aproximação.

```
>
```

Vamos utilizar o programa do teste χ^2 e do teste serial para testar as seqüências geradas a partir daqui.

Teste χ^2

```
> chi:=proc(_v::string,_s::posint,_n::posint)
> local _i,_j,_b,_w,_esp,_h:
> _b:=1/_n:
> if nargs>3 then ERROR(`Você deve digitar no máximo 3
  argumentos`) fi:
> for _i to _n do _w[_i]:=0: od:
> for _j to _n do
> for _i to _s do
> if (_v[_i]>=(_j-1)*_b) and (_v[_i]<(_j)*_b) then
> _w[_j]:=_w[_j]+1: fi:
> od:
>
> od:
> _esp:=_s/_n:
> RETURN(evalf(sum((_w[_h]-_esp)^2/_esp,_h=1..n))):
> end:
```

Teste Serial

```
> serial:=proc(v::string, w::posint,d::posint)
>  local i,j,a,b,z,s,q,r,l;
>  global yyy;
>    yyy:=array(1..w);
>    for i from 1 to w do
>     yyy[i]:=floor(d*v[i])+1;
>    od:
>
>     for j from 0 to d^2 do
>      s[j]:=0;
>     od;
>
>      i:=0;
>       for q from 1 to d do
>        for r from 1 to d do
>         for j from 1 to floor(w/2.) do
>          if yyy[2*j-1]=q and yyy[2*j]=r then
>           s[i]:=s[i]+1;
>          fi:
>         od:
>
>          i:=i+1;
>         od:
>       od:
>        z:=evalf(floor(w/2.)/(d^2));
>        RETURN(sum(((s['i']-z)^2)/z,'i'=0..d^2-1));
> end:
```

[>

Observação importante, o intervalo de números considerado para os testes acima é o intervalo [0, 1[, o teste χ^2 serve para verificar se os números da nossa seqüência estão uniformemente distribuídos em sub-intervalos do intervalo [0,1[.

Gerando e testando as seqüências de números pseudo-randômicos com o teste $\chi 2$ e serial. As seqüências *x* e *y* geradas são formadas por números no intervalo [0, 1 [.

```
> _sement:=44894597459;
```

$$_sement := 44894597459$$

```
> for i to 9999 do
> x[i]:=random(10^12,99999999641,3401)/(10^12):
> od:
> chi(x,9999,10);
```

$$4.993399340$$

```
> serial(x,9999,4);
```

$$16.57911582$$

```
>
> _sement:=6865922;
```

$$_sement := 6865922$$

```
> for i to 9999 do
> y[i]:=random(10^12,99999999981,3267)/(10^12):
> od:
> chi(y,9999,10);
```

$$9.379837984$$

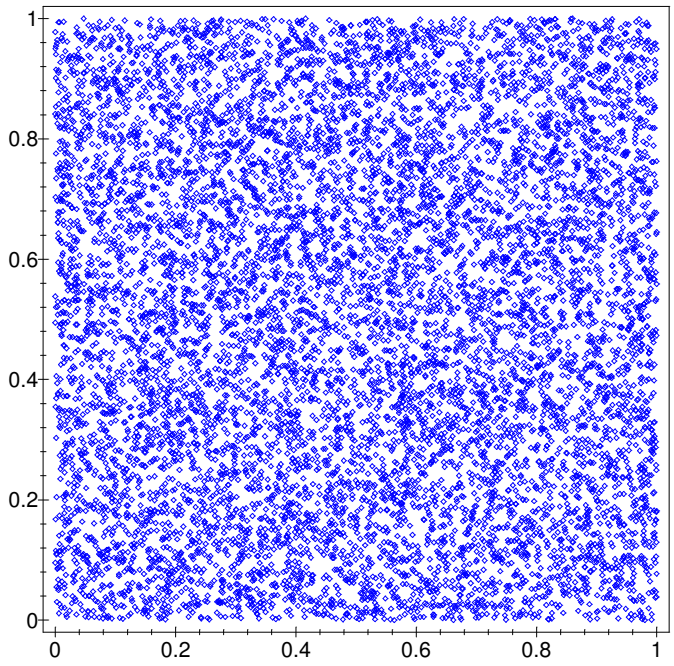
```
> serial(y,9999,4);
```

$$11.54770954$$

```
>
>
```

Visualização dos pontos formados por essas duas seqüências.

```
> with(plots):
> for i to 9999 do
> v[i]:=pointplot([evalf(x[i]),evalf(y[i])],symbol=POINT,col
  or=blue,axes=BOXED);
> od:
> display({seq(v[i],i=1..9999)},insequence=false);
```



Observemos que os pontos encontrados estão uniformemente distribuídos no quadrado unidade acima.

>

>

Exercício 13) [Jurg Nievergelt]

Calcular a probabilidade de que três pontos escolhidos aleatoriamente num plano infinito formem um triângulo obtuso. Estime esta probabilidade realizando experimentos com um computador. O valor desta probabilidade é:

$$\frac{3}{8-\frac{6\sqrt{3}}{\pi}}$$,aproximadamente 0,6394.

>

Resposta

```
> ca:=0:
> n:=3333:
> tempoi:=iolib(25):
> w:=1:
> for q to n do
>
> d[1]:=sqrt( (x[w]-x[w+1])^2 + (y[w]-y[w+1])^2 ):
> d[2]:=sqrt( (x[w]-x[w+2])^2 + (y[w]-y[w+2])^2 ):
```

```
> d[3]:=sqrt( (x[w+1]-x[w+2])^2 + (y[w+1]-y[w+2])^2 ):
>
> mx:= -1:
> for i to 3 do
> if evalf(d[i])>mx then
> mx:=evalf(d[i]):
> j:=i:
> fi:
> od:
>
> if j=1 then
>  mx:=(x[w]+x[w+1])/2;
>  my:=(y[w]+y[w+1])/2;
>  a1:=x[w+2];
>  a2:=y[w+2];
> elif j=2 then
>  mx:=(x[w]+x[w+2])/2;
>  my:=(y[w]+y[w+2])/2;
>  a1:=x[w+1];
>  a2:=y[w+1];
> else
>  mx:=(x[w+1]+x[w+2])/2;
>  my:=(y[w+1]+y[w+2])/2;
>  a1:=x[w];
>  a2:=y[w];
> fi:
>
> d[4]:=sqrt( (mx-a1)^2 + (my-a2)^2 ):
>
> if evalf(d[4]) < evalf(d[j]/2) then
> ca:=ca+1:
> fi:
> w:=w+3;
> od:
> print(tempo=iolib(25)-tempoi):
> ca;
>
```

$$tempo = 201$$

$$2437$$

```
> probabilidade_correta := evalf(3/(8-(6*sqrt(3)/Pi)));
> probabilidade_encontrada:= evalf(ca/n);
> erro:=abs(evalf(3/(8-(6*sqrt(3))/Pi))-");
```

$$probabilidade\_correta := .6393825609$$

$$probabilidade\_encontrada := .7311731173$$

$$erro := .0917905564$$

[>

Exercício 14) [Jurg Nievergelt]

Realizando experimentos com o computador, estimar a probabilidade de três pontos escolhidos aleatoriamente sobre os lados de um quadrado unidade formarem um triângulo obtuso. O valor atual é $\frac{19}{36}$ aproximadamente 0.528.

Resposta

```
> tempoi:=iolib(25):
> ca:=0:
> n:=3333:
>
> _sement:=78635920742:
> for i to 3*n do
>  a[i]:=x[i]:
> od:
>
> _sement:=78635920744:
> for i to 3*n do
>  g:=floor(4*y[i])+1:
>
>   if g = 1 then
>    x[i]:=a[i];
>    y[i]:=0;
>   elif g = 2 then
>    x[i]:=a[i];
>    y[i]:=1;
>   elif g = 3 then
>    x[i]:=0;
>    y[i]:=a[i];
>   else
>    x[i]:=1;
>    y[i]:=a[i];
>   fi:
> od:
>
> w:=1:
> for q to n do
>
> d[1]:=sqrt( (x[w]-x[w+1])^2 + (y[w]-y[w+1])^2 ):
> d[2]:=sqrt( (x[w]-x[w+2])^2 + (y[w]-y[w+2])^2 ):
> d[3]:=sqrt( (x[w+1]-x[w+2])^2 + (y[w+1]-y[w+2])^2 ):
>
```

```
> mx:= -1:
> for i to 3 do
> if evalf(d[i])>mx then
> mx:=evalf(d[i]):
> j:=i:
> fi:
> od:
>
> if j=1 then
>  mx:=(x[w]+x[w+1])/2;
>  my:=(y[w]+y[w+1])/2;
>  a1:=x[w+2];
>  a2:=y[w+2];
> elif j=2 then
>  mx:=(x[w]+x[w+2])/2;
>  my:=(y[w]+y[w+2])/2;
>  a1:=x[w+1];
>  a2:=y[w+1];
> else
>  mx:=(x[w+1]+x[w+2])/2;
>  my:=(y[w+1]+y[w+2])/2;
>  a1:=x[w];
>  a2:=y[w];
> fi:
>
> d[4]:=sqrt( (mx-a1)^2 + (my-a2)^2 ):
>
> if evalf(d[4]) < evalf(d[j]/2) then
> ca:=ca+1:
> fi:
> w:=w+3;
> od:
>
> print(tempo=iolib(25)-tempoi):
> ca;
>
>
```

$$tempo = 189$$

$$2558$$

```
> probabilidade := evalf(ca/n);
> erro := abs(19/36 - ");
```

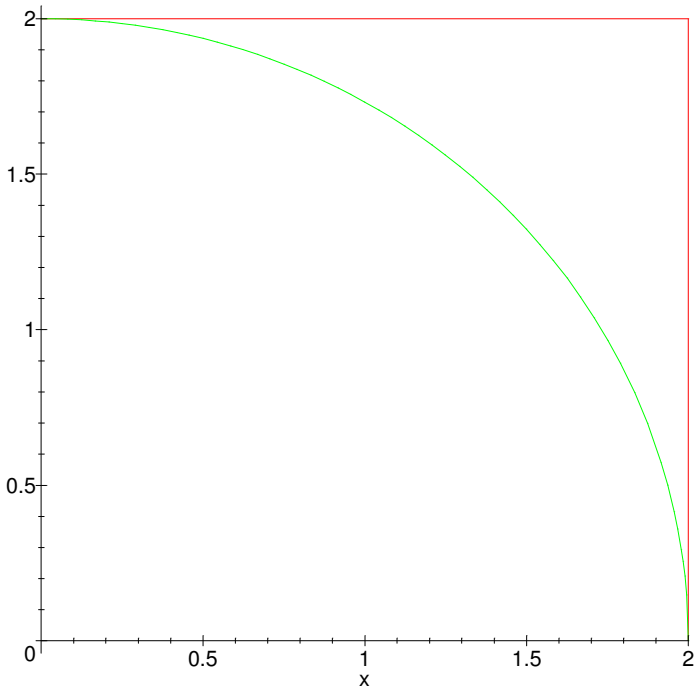
$$probabilidade := .7674767477$$

$$erro := .2396989699$$

Aproximar o valor de π usando uma técnica de Monte Carlo para calcular $\int_0^2 \sqrt{4-x^2}\,dx$.

Resposta

```
> plot({[[2,0],[2,2],[0,2]],sqrt(4-x^2)},x=0..2);
```




```
> area:=evalf(Pi);
```

$$area := 3.141592654$$

```
> int(sqrt(4-x^2),x=0..2);
```

$$\pi$$

O programa abaixo verifica a porcentagem de pontos que caíram dentro da área formada pelo gráfico da figura acima., isto é, o gráfico da equação $f(x) = \sqrt{4-x^2}$.

```
> f:=x->sqrt(4-x^2):
> n:=9999:
> c:=0:
> for i to n do
> if evalf(2*y[i]) <= evalf(f(2*x[i])) then
> c:=c+1:
> fi:
> od:
> print(`pontos internos =`. c):
```

pontos internos =7853

```
> evalf(c/n);
```
Porcentagem dos pontos que cairam dentro da área formada pelo gráfico é:

.7853785379

Para calcular o valor da integral fazemos:

Área = area do quadrado * porcentagem.

```
> AREA:=evalf(4*c/n);
> erro := evalf(abs(Pi - "));
```

AREA := 3.141514151

erro := .000078503

Observe que o valor da integral é π.

```
>
```

Exercício 16) [Jurg Nievergelt]

Considere a área A formada pelas inequações no plano xy :

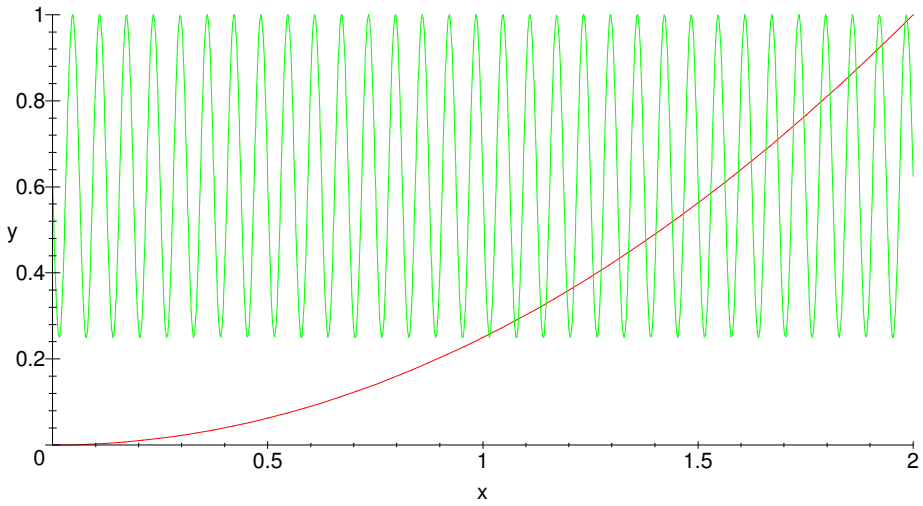
$$0 \le x$$
$$x \le 2$$
$$\frac{x^2}{4} \le y$$
$$y \le \frac{3\sin(32\,\pi\,x + \pi) + 5}{8}$$

Calcular a área A usando métodos de Monte Carlo obtendo pontos dentro do retângulo formado pelas inequações abaixo:

$$0 \le x$$
$$x \le 2$$
$$0 \le y$$
$$y \le 1$$

Resposta

```
> plot([(x^2)/4,(3*sin(32*Pi*x+Pi)+5)/8],x=0..2,y=0..1,sc
  aling=constrained);
```




```
> n:=9999:
> c:=0:
> for i to n do
> if (evalf(y[i])<=evalf((3*sin(32*Pi*(2*x[i])+Pi)+5)/8))
  and (evalf(y[i])>=evalf((2*x[i])^2/4)) then
> c:=c+1;
> fi;
> od:
> probabilidade:=evalf(c/n);
```

$$probabilidade := .3501350135$$

```
> area:=probabilidade*2;
> AREA:=evalf(int((3*sin(32*Pi*x+Pi)+5)/8,x=0..2)-int(x^2
  /4,x=0..2));
> erro:=abs("-"");
```

$$area := .7002700270$$

$$AREA := .5833333333$$

$$erro := .1169366937$$

Vejamos agora dois exemplos aplicados em Simulação:

Observando o gráfico da figura, percebemos que para calcular a área, pelo Método de Monte Carlo acima, é necessário que tenhamos pontos muito bem distruibuídos no quadrado. Isto implica que a nossa seqüência de números pseudo-randômicos deve ser muito bem distribuída para que o erro seja mínimo. O que aconteceu acima, foi que a nossa seqüência de números randômicos é bem distribuída, mas não o suficiente para conseguir um erro mínimo para o valor da área da figura dada.

>

TEOREMA DE BUFFON

(Buffon, 1777). Uma agulha de comprimento l é jogada aleatoriamente sobre um plano reticulado com retas paralelas eqüidistantes. Se d é a distância entre duas retas consecutivas e se $l < d$, a probabilidade da agulha interceptar alguma reta é $\frac{2\,l}{\pi\,d}$.

desenho

Vamos utilizar os métodos de *Monte Carlo* para verificar o teorema de Buffon. Primeiramente, vamos esboçar o problema:

desenho

Como vemos na figura acima, precisamos de apenas duas coordenadas para localizar a agulha, uma é x, a qual é a distância do centro da agulha à reta mais próxima e ϕ pertence a $[\,0\,,\,\frac{\pi}{2}\,]$ que é o ângulo que a agulha faz com a direção das retas paralelas.

Agora vamos definir o valor de d, o valor de l, quantos lançamentos queremos realizar e então gerar duas seqüências de números pseudo-randômicos. A primeira seqüência de números pseudo-randômicos gerados é a seqüência x_i que vai retornar o valor de x. Da mesma forma, a seqüência ϕ_i gerada retorna o ângulo ϕ.

```
> restart;
```

Definindo as constantes de entrada.

```
> d:=20:
> l:=8:
> n:=20000:
```

Gerando as coordenadas aleatoriamente.

```
> for i to n do
> x[i]:=(rand()/999999999988)*(d/2):
> od:
> for i to n do
> phi[i]:=(rand()/999999999988)*(Pi/2):
> od:
```

Contando quantas vezes a agulha intercepta as retas paralelas. Observe, que a agulha intercepta a reta se $x \le \frac{l\,\sin(\phi)}{2}$.

```
> cnt:=0:
> for i to n do
>  if x[i] <= evalf((1/2)*sin(phi[i])) then
>   cnt:=cnt+1:
>  fi:
> od:
```

```
> Valor:=evalf(cnt/n);
```
valor estimado a partir do método de *Monte Carlo*.

Valor := .2531500000

```
> Buffon:=evalf((2*l)/(Pi*d));
```
valor segundo Buffon.

Buffon := .2546479089

```
> erro:=abs(Buffon-Valor);
```

erro := .0014979089

Observe que conseguimos uma boa aproximação do resultado de Buffon.

```
>
```

5.4. EXEMPLO DE SIMULAÇÃO DE FILAS

Exemplos de simulação 1:

Simulação de uma fila de "correio".

Numa agência de correios as pessoas formam uma fila para serem atendidas por um único funcionário. As pessoas chegam num rítmo uniforme. Pretendemos com esta simulação verificar se o funcionário supre a demanda ou se é necessário contratar um novo funcionário.

```
> restart;
```

```
> m:=08:
```
intervalo máximo de tempo entre as chegadas de fregueses;

```
> n:=10:
```
número de fregueses no sistema (escolhemos n pequeno pois no momento estamos interessados apenas em mostrar como usar o Maple nessa situação);

```
> x:=3:
```
tempo máximo de duração do atendimento;

```
> T:=array(1..n+1,1..3):
```
construção da tabela

```
> T[1,1]:=Fregues:
> T[1,2]:=TEnCh:
```
intervalo de tempo entre as chegadas dos fregueses

```
> T[1,3]:=TCh:
```
instante em que o freguês chega

```
> for i from 2 to n+1 do
```
coluna do número do freguês

```
> T[i,1]:=i-1;
> od:
> T[2,2]:=0:
```
intervalo de tempo de chegada do primeiro freguês

```
> T[2,3]:=0:  instante  de chagada do primeiro freguês
> for i from 2 to n do  intervalo randômico de chegada dos outros fregueses
> T[i+1,2]:=rand(1..m)();    usando o gerador  do Maple
> T[i+1,3]:=T[i,3]+";  calculando o instante de chegada
> od:
>
```

Vamos ver a tabela:

```
> evalm(T);
```

Fregues	*TEnCh*	*TCh*
1	0	0
2	2	2
3	7	9
4	2	11
5	4	15
6	1	16
7	3	19
8	6	25
9	1	26
10	2	28

```
>
```

Vamos gerar uma tabela contendo o tempo de atendimento:

```
> T1:=array(1..n+1,1..2):
> T1[1,1]:=Fregues:
> T1[1,2]:= TAt:  coluna do tempo de atendimento
> for i to n do  gerando randômicamente os tempos de atendimento
> T1[i+1,2]:=rand(1..x)();  usando o gerador do Maple
> T1[i+1,1]:=i;    coluna do número do freguês
> od:
>
> evalm(T1);
```

Fregues	*TAt*
1	2
2	2
3	1
4	1
5	3
6	3
7	2
8	1
9	1
10	1

```
>
```

Vamos definir agora a tabela principal:

```
> T2:=array(1..n+1,1..7):
> T2[1,1]:=Fregues:
```

```
> T2[1,2]:=TCh:  coluna do instante  de chegada dos fregueses
> T2[1,3]:=TCAt:  coluna do instante em que começa o atendimento
> T2[1,4]:=TAt:  coluna do tempo que dura o atendimento
> T2[1,5]:=TtAt:  coluna do instante em que o atendimento termina
> T2[1,6]:=TFEsp:  coluna do tempo que o freguês espera na fila
> T2[1,7]:=TFunDes:  coluna do tempo que o funcionário está desocupado
> T2[n+1,7]:=0:  célula não utilizada
> T2[2,3]:=0:  inicializando o tempo que começa o atendimento
> for i to n do   extraindo os dados da tabelas anteriores
> T2[i+1,1]:=i:  número do freguês
> T2[i+1,2]:=T[i+1,3]:  instante de chegada
> T2[i+1,4]:=T1[i+1,2]:  tempo que dura o atendimento
> od:
> for i to n do
> if i>=2 then
> T2[i+1,3]:=max(T2[i+1,2], T2[i,5]); instante em que começa o
  atendimento
> fi:
> T2[i+1,5]:=T2[i+1,3]+T2[i+1,4]: instante em que o atendimento termina
> T2[i+1,6]:=T2[i+1,3]-T2[i+1,2];  tempo de espera na fila
> od:
> for i from 1 to n-1 do
> T2[i+1,7]:=T2[i+2,3]-T2[i+1,5]:  tempo que o funcionário fica
  desocupado
> od:
>
> evalm(T2);
```

Fregues	*TCh*	*TCAt*	*TAt*	*TtAt*	*TFEsp*	*TFunDes*
1	0	0	2	2	0	0
2	2	2	2	4	0	5
3	9	9	1	10	0	1
4	11	11	1	12	0	3
5	15	15	3	18	0	0
6	16	18	3	21	2	0
7	19	21	2	23	2	2
8	25	25	1	26	0	0
9	26	26	1	27	0	1
10	28	28	1	29	0	0

```
> 
```

Resultados da Simulação:

Tempo médio de espera na fila:

```
> evalf(sum(T2['i+1',6],'i'=1..n)/n);
```

.4000000000

Probabilidade do fregues que chega encontrar uma fila (porcentagem):

```
> fe:=0:
> for i to n do
> if T2[i+1,6]<>0 then
> fe:=fe+1;
> fi;
> od:
> evalf(fe/n)*100;
```

20.00000000

Proporção de tempo que o funcionário fica desocupado:

```
> evalf(sum(T2['i+1',7],'i'=1..n-1)/T2[n+1,5]);
```

.4137931034

A proporção de tempo que o funcionário está ocupado é o complemento do resultado anterior, ou seja:

```
> 1-";
```

.5862068966

A média do tempo de atendimento é:

```
> evalf(sum(T2['i+1',4],'i'=1..n)/n);
```

1.700000000

Média do tempo entre as chegadas:

```
> evalf(sum(T['i+1',2],'i'=1..n)/(n-1*));
```

3.111111111

* Subtrai-se um devido a se considerar o primeiro tempo de chegada como sendo zero.

Média do tempo de espera de quem espera na fila:

```
> Tempo:=0:
> fe:=0:
> for i to n do
> if T2[i+1,6]<>0 then
> Tempo:=Tempo+T2[i+1,6];
> fe:=fe+1;
> fi;
> od:
> evalf(Tempo/fe);
```

2.

Média de tempo que o fregues gasta na agência:

```
> evalf((sum(T2['i+1',4],'i'=1..n)+Tempo)/n);
```

2.100000000

```
>
```

CONCLUSÃO

Examinando os resultados da simulação, pode-se concluir se há necessidade de se contratar um novo funcionário para atender a clientela.

Agora vale observar que os resultados acima podem não ser muito confiáveis, pois o número de fregueses no sistema é muito baixo.

Exemplo de simulação 2:

Simulação de um mercado

Num mercado existem dois balconistas para atender a população de uma cidade. Os dados estão mostrados nas 3 tabelas abaixo. O que vamos verificar com esta simulação é se o número de balconistas é suficiente ou não.

O tempo está medido em minutos.

```
> restart;
>
```

Costrução das tabelas

```
> n:=5:                          Tempo máximo entre as chegadas
> T:=array(1..n+1,1..3):
> T[1,1]:=TEnCheg:               Tempo entre as chegadas de pessoas
> T[1,2]:=Prob:                  Probabilidade
> T[1,3]:=DigRand:               Dígito randômico atribuido
> T[2,2]:=0.10: T[3,2]:=0.15: T[4,2]:=0.40: T[5,2]:=0.20:
> T[6,2]:=0.15:
> T[2,3]:=[0..10]: T[3,3]:=[11..25]: T[4,3]:=[26..65]:
> T[5,3]:=[66..85]: T[6,3]:=[86..100]:
> for i to n do
> T[i+1,1]:=i;
> od:
>
>
```

Tabela de Serviço do balconista 1:

```
> n:=7:
> T2:=array(1..n+1,1..2):
```

```
> T2[1,1]:=TAtend:          Tempo que demora o atendimento
> T2[1,2]:=Prob:            Probabilidade
> T2[1,3]:=DRand:           Dígito randômico atribuído
> for i from 2 to n+1 do
> T2[i,1]:=i:
> od:
> T2[2,2]:=0.05: T2[3,2]:=0.10: T2[4,2]:=0.20:
  T2[5,2]:=0.25:
> T2[6,2]:=0.21: T2[7,2]:=0.10: T2[8,2]:=0.9:
> T2[2,3]:=[0..5]: T2[3,3]:=[6..15]: T2[4,3]:=[16..35]:
  T2[5,3]:=[36..60]:T2[6,3]:=[61..81]: T2[7,3]:=[82..91]:
> T2[8,3]:=[92..100]:
>
>
```

Tabela de serviço do balconista 2:

```
> n:=7:
> T3:=array(1..n+1,1..3):
> T3[1,1]:=TAtend:          Tempo que demora o atendimento
> T3[1,2]:=Prob:            Probabilidade
> T3[1,3]:=DRand:           Dígito randômico atribuído
> for i from 4 to n+3 do
> T3[i-2,1]:=i:
> od:
> T3[2,2]:=0.10: T3[3,2]:=0.10: T3[4,2]:=0.20:
  T3[5,2]:=0.20:
> T3[6,2]:=0.21: T3[7,2]:=0.10: T3[8,2]:=0.09:
> T3[2,3]:=[0..10]: T3[3,3]:=[11..20]: T3[4,3]:=[21..40]:
  T3[5,3]:=[41..60]:T3[6,3]:=[61..81]: T3[7,3]:=[82..91]:
> T3[8,3]:=[92..100]:
>
> evalm(T);
```

Tabela da freqüência com que os fregueses chegam

TEnCheg	*Prob*	*DigRand*
1	.10	[0 .. 10]
2	.15	[11 .. 25]
3	.40	[26 .. 65]
4	.20	[66 .. 85]
5	.15	[86 .. 100]

```
> evalm(T2);
```

Tabela de probabilidades que o primeiro balconista atente um freguês

TAtend	Prob	DRand
2	.05	[0 .. 5]
3	.10	[6 .. 15]
4	.20	[16 .. 35]
5	.25	[36 .. 60]
6	.21	[61 .. 81]
7	.10	[82 .. 91]
8	.9	[92 .. 100]

```
> evalm(T3);
```

Tabela de probabilidades que o segundo balconista atende um freguês

TAtend	Prob	DRand
4	.10	[0 .. 10]
5	.10	[11 .. 20]
6	.20	[21 .. 40]
7	.20	[41 .. 60]
8	.21	[61 .. 81]
9	.10	[82 .. 91]
10	.09	[92 .. 100]

Baseado nos dados acima, vamos simular esse sistema de fila e verificar se a quantidade de balconistas é suficiente.

É considerado que o primeiro freguês chegou no tempo 0.

```
>
> n:=10:                Número de fregueses no sistema.  ( Pode ser mudado )
> T4:=array(1..n+2,1..12):
> for i to n+2 do
>  for j to 12 do
>   T4[i,j]:=``:        Inicializando a tabela com células vazias
>  od:
> od:
> T4[1,1]:=Fr:          Freguês
> T4[1,2]:=DRa:         Dígito Randômico para o tempo de chegada
> T4[1,3]:=TEch:        Tempo entre as chegadas de fregueses
> T4[1,4]:=TCh:         Tempo de chegada
> T4[1,5]:=DRs:         Dígito randômico para o tempo de serviço
> T4[1,6]:=TCS1:        Tempo que o serviço começa no 1º balcão
> T4[1,7]:=TS1:         Tempo de Serviço no 1º balcão
> T4[1,8]:=TSA1:        Tempo que o serviço acaba no 1º balcão
> T4[1,9]:=TCS2:        Tempo que o serviço começa no 2º balcão
> T4[1,10]:=TS2:        Tempo de Serviço no 2º balcão
> T4[1,11]:=TSA2:       Tempo que o serviço acaba no 2º balcão
> T4[1,12]:=TEFil:      Tempo de espera na fila
> T4[2,1]:=1: T4[2,2]:=0: T4[2,3]:=0: T4[2,4]:=0:Inicial. o Sistema
> _seed:=1:                                     Semente fixa.
```

```
> for i to n do
> T4[i+1,5]:=rand(0..100)():      Usando o gerador do Maple
> od:
> _seed:=8348783838:              Semente fixa.
> for i to n do
> T4[i+1,1]:=i:
> T4[i+1,2]:=rand(0..100)():      Usando o gerador do Maple
>                    Tempo entre as chegadas
> if (T4[i+1,2]>=00)and(T4[i+1,2]<=10) then T4[i+1,3]:=1:fi:
> if (T4[i+1,2]>=11)and(T4[i+1,2]<=25) then T4[i+1,3]:=2:fi:
> if (T4[i+1,2]>=26)and(T4[i+1,2]<=65) then T4[i+1,3]:=3:fi:
> if (T4[i+1,2]>=66)and(T4[i+1,2]<=85) then T4[i+1,3]:=4:fi:
> if (T4[i+1,2]>=86)and(T4[i+1,2]<=100)then T4[i+1,3]:=5:fi:
>
> od:
> for i from 2 to n do
> T4[i+1,4]:=T4[i,4]+T4[i+1,3]:   Calculando o tempo de chegada
> od:
> a:=0: b:=0:
> for i to n do
>                    Primeiro balconista
> if (a<=T4[i+1,4]) then
>
> if (T4[i+1,5]>=00)and(T4[i+1,5]<=05)  then t:=2:fi:
> if (T4[i+1,5]>=06)and(T4[i+1,5]<=15)  then t:=3:fi:
> if (T4[i+1,5]>=16)and(T4[i+1,5]<=35)  then t:=4:fi:
> if (T4[i+1,5]>=36)and(T4[i+1,5]<=60)  then t:=5:fi:
> if (T4[i+1,5]>=61)and(T4[i+1,5]<=81)  then t:=6:fi:
> if (T4[i+1,5]>=82)and(T4[i+1,5]<=91)  then t:=7:fi:
> if (T4[i+1,5]>=92)and(T4[i+1,5]<=100) then t:=8:fi:
>
> T4[i+1,6]:=T4[i+1,4]:           Tempo que começa o serviço
> a:=T4[i+1,4]+t:
> T4[i+1,7]:=t:                   Tempo que dura o atendimento
> T4[i+1,8]:=T4[i+1,6]+T4[i+1,7]: Tempo que acaba o atendimento
> T4[i+1,12]:=T4[i+1,6]-T4[i+1,4]: Tempo que o freguês espera na fila
> else
>                    Segundo balconista
>  if (b<=T4[i+1,4]) then
>
>  if (T4[i+1,5]>=00)and(T4[i+1,5]<=10)  then t2:=2:fi:
>  if (T4[i+1,5]>=11)and(T4[i+1,5]<=20)  then t2:=3:fi:
>  if (T4[i+1,5]>=21)and(T4[i+1,5]<=40)  then t2:=4:fi:
>  if (T4[i+1,5]>=41)and(T4[i+1,5]<=60)  then t2:=5:fi:
>  if (T4[i+1,5]>=61)and(T4[i+1,5]<=81)  then t2:=6:fi:
```

```
>  if (T4[i+1,5]>=82)and(T4[i+1,5]<=91)  then t2:=7:fi:
>  if (T4[i+1,5]>=92)and(T4[i+1,5]<=100) then t2:=8:fi:
>
>  T4[i+1,9]:=T4[i+1,4]:                    Tempo que começa o atendimento
>  b:=T4[i+1,4]+t2:
>  T4[i+1,10]:=t2:                          Tempo que dura o atendimento
>  T4[i+1,11]:=T4[i+1,9]+T4[i+1,10]:  Tempo que acaba o atendimento
>  T4[i+1,12]:=T4[i+1,9]-T4[i+1,4]:   Tempo que o freguês espera na fila
>                          Os dois balconistas estão ocupados
>  else
>     if a=min(a,b) then
>      m:=min(a,b):          O atendimento começa com o balconista que acabar
  primeiro
>      T4[i+1,6]:=m:         Tempo que começa o atendimento
>      T4[i+1,7]:=t:         Tempo que dura o atendimento
>      T4[i+1,8]:=T4[i+1,6]+T4[i+1,7]:  Tempo que acaba o atendimento
>
>      a:=a+t:
>     else
>      m:=min(a,b):
>      T4[i+1,9]:=m:      Tempo que começa o atendimento
>      T4[i+1,10]:=t2:    Tempo que dura o atendimento
>      T4[i+1,11]:=T4[i+1,9]+T4[i+1,10]:  Tempo que acaba o
  atendimento
>      b:=b+t2:
>     fi:
>     T4[i+1,12]:=m-T4[i+1,4]:    Tempo que o freguês espera na fila
>  fi:
> fi:
> od:
> for j from 7 to 10 by 3 do
> for i to n do
>  if T4[i+1,j]=`` then
>   T5[i+1,j]:=0:           Repassando os valores para uma tabela auxiliar
>  else
>   T5[i+1,j]:=T4[i+1,j]:
>  fi:
> od:
> T4[n+2,j]:=sum('T5[i+1,j]','i'=1..n):      Soma dos tempos de
  serviço
> od:
> T4[n+2,12]:=sum('T4[i+1,12]','i'=1..n): Soma do tempo de espera
> for i to n do
> if (T4[i+1,8]<>``) then T4[n+2,8]:=T4[i+1,8] fi:
> if (T4[i+1,11]<>``) then T4[n+2,11]:=T4[i+1,11] fi:
```

```
> od:
> evalm(T4);
>
```

Fr	*DRa*	*TEch*	*TCh*	*DRs*	*TCS1*	*TS1*	*TSA1*	*TCS2*	*TS2*	*TSA2*	*TEFil*
1	43	3	0	70	0	6	6	,	,	,	0
2	27	3	3	76	,	,	,	3	6	9	0
3	38	3	6	37	6	5	11	,	,	,	0
4	92	5	11	82	11	7	18	,	,	,	0
5	91	5	16	29	,	,	,	16	4	20	0
6	3	1	17	56	18	7	25	,	,	,	1
7	84	4	21	42	,	,	,	21	5	26	0
8	15	2	23	47	25	7	32	,	,	,	2
9	40	3	26	21	,	,	,	26	4	30	0
10	47	3	29	41	,	,	,	30	4	34	1
,	,	,	,	,	,	32	32	,	23	34	4

```
>
```

Calculando as estatísticas

Probabilidade que o primeiro balconista esteja ocupado (porcentagem):

```
> evalf(T4[n+2,7]/T4[n+2,8])*100;
>
```

100.

Probabilidade que o segundo balconista esteja ocupado (porcentagem):

```
> evalf(T4[n+2,10]/T4[n+2,11])*100;
```

67.64705882

Média de tempo que os fregueses esperam na fila

```
> evalf(T4[n+2,12]/n);
```

.4000000000

Média de tempo dos fregueses que esperaram na fila

```
> j:=0:
> for i to n do
> if T4[i+1,12]<>0 then
> j:=j+1:
> fi:
> od:
> evalf(T4[n+2,12]/j);
```

1.333333333

Resultados:

A partir dos dados acima é possível verificar se há a necessidade de se contratar um novo funcionário. Para se fazer uma simulação mais acurada, é necessário pesquisar os dados de entrada e também é necessário um mínimo de fregueses que o funcionario possa atender.

>